PLOS Computational Biology

Towards a comprehensive visualization of structure in large scale data sets

--Manuscript Draft--

Manuscript Number:	PCOMPBIOL-D-22-00121
Full Title:	Towards a comprehensive visualization of structure in large scale data sets
Short Title:	Towards a comprehensive visualization of structure in large scale data sets
Article Type:	Research Article
Keywords:	big data, large scale datasets, visualization, dimensional reduction, embedding, parallelization
Abstract:	Dimensional reduction methods are fundamental to exploring and visualizing large data sets. Basic requirements for unsupervised data exploration are simplicity, flexibility and scalability. However, current methodologies show complex parameterizations and computational limitations when exploring large data structures across scales. We focus on the t-SNE algorithm and show that a simplified parameter setup with a single control parameter, namely the perplexity, can effectively balance local and global data structure visualization. We also describe a chunk&mix protocol to parallelize t-SNE, and we use it to build pt-SNE, a parallel version of the Barnes-Hut t-SNE. The pt-SNE converges to good global embedding, comparable to state-of-the-art, and allows us to explore data structure much beyond current limitations. The downside is a little noise at the local scale that simple post-processing can efficiently restore. We expect the same approach to apply to faster embedding algorithms other than BHt-SNE, like FIt-SNE or UMAP, thus, extending the state-of-the-art and leading to more comprehensive data structure visualization and analysis.
Additional Information:	
Question	**Response**
Financial Disclosure Enter a financial disclosure statement that describes the sources of funding for the work included in this submission. Review the submission guidelines for detailed requirements. View published research articles from *PLOS Computational Biology* for specific examples. This statement is required for submission and **will appear in the published article** if the submission is accepted. Please make sure it is accurate.	The authors received no specific funding for this work.

Unfunded studies Enter: *The author(s) received no specific funding for this work.* **Funded studies** Enter a statement with the following details: • Initials of the authors who received each award • Grant numbers awarded to each author • The full name of each funder • URL of each funder website • Did the sponsors or funders play any role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript? • **NO** - Include this sentence at the end of your statement: *The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.* • **YES** - Specify the role(s) played. * typeset	
Competing Interests Use the instructions below to enter a competing interest statement for this submission. On behalf of all authors, disclose any competing interests that could be perceived to bias this work—acknowledging all financial support and any other relevant financial or non-financial competing interests. This statement **will appear in the published article** if the submission is accepted. Please make sure it is accurate. View published research articles from *PLOS Computational Biology* for specific examples.	The authors have declared that no competing interests exist.

NO authors have competing interests Enter: *The authors have declared that no competing interests exist.* **Authors with competing interests** Enter competing interest details beginning with this statement: *I have read the journal's policy and the authors of this manuscript have the following competing interests: [insert competing interests here]* * typeset This statement is required for submission and **will appear in the published article** if the submission is accepted. Please make sure it is accurate and that any funding sources listed in your Funding Information later in the submission form are also declared in your Financial Disclosure statement.	
Data and Code Availability From the time of publication, Authors are required to make fully available and without restriction all data and computational code underlying their findings. Please see our PLOS Data Policy page for detailed policy information, and our Code Sharing page for specific information on code sharing. A **Data Availability Statement**, detailing where the data (and code, if applicable) can be accessed, is required at first submission. Insert your Data Availability Statement in the box below. The statement you provide **will be published in the article**, if accepted. Please see the Data Reporting section of our submission guidelines for instructions on what you need to include in your Data Availability Statement.	The data sets used in this work and the necessary code to reproduce the results are available at https://doi.org/10.5281/zenodo.5772287. The code for the examples is also available at https://rpubs.com/bigMap/. The latter provides immediate access to individual HTML documents explaining each example. The pt-SNE algorithm is part of the bigMap R-package. We used version bigMap 4.6.0, available at https://doi.org/10.5281/zenodo.5779186. We run all our analysis on the HPC platform at the Computational Biology Lab (CEAB-CSIC) (Table 1).

If the data and code are all contained in your submission files, please state: *All relevant data are within the manuscript and its Supporting Information files.* **PLOS allows rare exemptions to address legal and ethical concerns.** If you have legal or ethical restrictions, please detail these in your Data Availability Statement below for the Journal team to consider.	

Towards a comprehensive visualization of structure in large scale data sets

Joan Garriga[1,*], Frederic Bartumeus[1,2,3]

1 Theoretical and Computational Ecology Group,
Centre d'Estudis Avançats de Blanes (CEAB-CSIC),
Cala Sant Francesc, 14, 17300, Blanes, Spain
2 Centre de Recerca Ecològica i Aplicacions Forestals (CREAF),
Cerdanyola del Vallès, 08193, Barcelona, Spain
3 Institució Catalana de Recerca i Estudis Avançats, ICREA,
Passeig Lluís Companys, 23, 08010, Barcelona, Spain
E-mail: jgarriga@ceab.csic.es
URL: http://theelab.net/

Abstract

Dimensional reduction methods are fundamental to exploring and visualizing large data sets. Basic requirements for unsupervised data exploration are simplicity, flexibility and scalability. However, current methodologies show complex parameterizations and computational limitations when exploring large data structures across scales. We focus on the t-SNE algorithm and show that a simplified parameter setup with a single control parameter, namely the perplexity, can effectively balance local and global data structure visualization. We also describe a chunk&mix protocol to parallelize t-SNE, and we use it to build pt-SNE, a parallel version of the Barnes-Hut t-SNE. The pt-SNE converges to good global embedding, comparable to state-of-the-art, and allows us to explore data structure much beyond current limitations. The downside is a little noise at the local scale that simple post-processing can efficiently restore. We expect the same approach to apply to faster embedding algorithms other than BHt-SNE, like FIt-SNE or UMAP, thus, extending the state-of-the-art and leading to more comprehensive data structure visualization and analysis.

1 Author summary

A growing need in many research fields is the development of tools to process and visualize large-scale data sets. Neurosciences and quantitative behaviour analyses use experimental big data sets from model organisms (e.g., nematodes, fruit flies, zebrafish larvae, social insects, mice) to infer organizational principles and generative mechanisms of behaviour with minimal or no prior assumptions. Molecular biology or single-cell genomics and transcriptomics generate gene

expression data from tens to hundreds of thousands of cells using improved experimental techniques. Dimensional reduction methods are fundamental tools to explore and visualize such large amounts of data. However, current methods show complex parameterizations and significant computational limitations as data size increases. We propose a big data approach to increase the flexibility and scalability of these methods. We developed a generic protocol to efficiently parallelize dimensional reduction methods accounting for a comprehensive data exploration across scales.

Introduction

High dimensional data sets are prevalent in many areas of research. The computational analysis of these data often entails unsupervised exploratory steps where low dimensional visualization of data becomes crucial. Dimensional reduction techniques [1] are mainly divided into linear projections, focused on preserving the global structure of the data, *e.g. Principal Component Analysis* [2], *multidimensional scaling* [3] and non-linear embedding, focused on preserving the local structure of the data, *e.g. Sammon mapping* [4], *Isomap* [5], *Laplacian eigenmaps* [6]. Two main ideas support the use of non-linear techniques for the visualization of high dimensional data: (i) high dimensional data is likely to be organized in a non-linear manifold of much lower dimension where the visualization of the global structure could make little sense, and (ii) linear projections of high dimensional data to a human-readable scale (i.e. two or three first components) might drop off essential local information. However, non-linear methodologies share serious drawbacks regarding the computational complexity of the algorithms and the interpretation of the output when the algorithm is not well-understood [7].

Two excellent methodologies stand out from the flurry of non-linear dimensional reduction techniques: (i) t-SNE (t-Stochastic neighbouring Embedding, [8, 9]) with many faster variants (e.g. Barnes-Hut t-SNE (BHt-SNE) [10], MultiCore t-SNE (opt-SNE) [11, 12], Fast-Fourier Interpolation-based t-SNE (FIt-SNE) [13]) widely used in flow and mass cytometry and single-cell RNA sequencing data analysis ([14, 12]), and (ii) UMAP (Uniform Manifold Approximation and Projection, [15]), increasingly prevalent in many areas of scientific research ([16]). t-SNE and UMAP are both gradient descent algorithms operating as a set of attractive and repulsive forces

that make similar points attract each other and push dissimilar ones far away. Also, comparing UMAP and FIt-SNE, both algorithms are similar in terms of memory resources and computation times ($\mathcal{O}(n)$), both scaling well with large data sets.

Although t-SNE and UMAP are similar algorithms, there is an open discussion among practitioners about which algorithm is better at capturing the structure in the data [16, 17]. In t-SNE and UMAP, a neighbourhood-size parameter modulates how much the algorithm focus on capturing local or global structure, namely *perplexity* (ppx) for t-SNE and *nearest-neighbours* for UMAP. Based on this parameter, the first step in both algorithms is to compute a matrix of affinities, and the fact is that this step does not scale so well for large values of the neighbourhood-size. Therefore, an important assumption in both frameworks is that the input data will usually reside in a manifold (a topological space where the neighbourhood around every point is only locally Euclidean). While this assumption supports the use of low-ranged ppxs, it tells nothing about how large a locally Euclidean neighbourhood can be. The effect of increased ppx is not yet thoroughly explored (just up to 0.001% for a 1M data set [12]), though the general assumption is that using low-ranged perplexities may constitute a potential limitation to unveiling higher levels of data structuring. Common techniques used to overcome this limitation are: (i) using informative initialization methods to inject the global structuring into the embedding from the very first stage of the algorithm (e.g. principal component analysis initialization, [8]), and (ii) performing multi-scale analysis by mixing different scales ([18], [19]).

Another potentially limiting aspect of non-linear embedding techniques is the non-trivial parametric setup. The huge parameter space and the difficulty in understanding its influence on the output can be daunting to non-experts. In particular, for t-SNE, the main parameters are initialization, learning-rate, momentum, early/late exaggeration and ppx. In addition, a few more parameters control a contrived internal trickery used in the computation of the gradient descent to force a quick convergence and a neat visualization of the structure (e.g. momentum switch, early-exaggeration stop, step size update [20]). Such parametric complexity has motivated specific work giving some interesting clues though not compelling enough conclusions about optimal t-SNE parameterization[18, 7, 21, 22, 19, 12]. For the recently appeared UMAP, the lack of information about optimal tuning is even more critical.

We focus here on t-SNE with a double aim: first, to provide rationale on simplifying t-SNE

parameterization, and second, to unleash the limits of t-SNE far beyond the range of small neighbourhoods usually considered. For the latter objective, we assume that random subsets of data suffice in representing the whole data structuring. More generally, this would imply that large scale data sets convey large amounts of redundant evidence. Therefore, we propose to break down t-SNE into partial/parallel t-SNE runs on subsets of data, and we describe a chunk&mix protocol that promotes the convergence of the partial solutions into a global one.

Results

As a proof-of-concept, we provide a parallelized BHt-SNE, namely pt-SNE, where we apply the chunk&mix protocol and a simplified parametric setup (see section Parametric configuration of the cost gradient in Methods). Nevertheless, we want to highlight the generality of the concept, possibly suitable also to FIt-SNE or UMAP. We show that pt-SNE achieves good approximations to FIt-SNE, which we take here as our ground truth, and significantly reduces the computational complexity that penalizes t-SNE for increasing ppxs. Herein, we can explore data structuring at scales that otherwise remained prohibitive. The downside is a small cost in local accuracy that, if required, we can improve with simple post-processing later on.

We assess the goodness of pt-SNE visualizations by means of the *k-ary neighbourhood preservation* (kNP, [18]). The kNP depicts the matching between neighbourhoods in the high and the low dimensional spaces (HD, LD) across a wide range of neighbourhood sizes. A further summarization of kNP is the area under the curve (AUC, [18]). The kNP is commonly depicted against a logarithmic transformation of ppx to enhance the results of the low range of neighbourhood sizes. As our interest extends now to higher values of ppx, we include a linear version of the kNP. Consequently, we distinguish linear and logarithmic versions of the AUC (linAUC and logAUC, respectively) to characterize the high-dimensional to the low-dimensional matching of global and local data structures.

Global/Local structure trade-off

It is well established that t-SNE fails to capture the global structure of large data sets [7, 19] and, since the appearance of UMAP, an increasing feeling among practitioners is that UMAP outper-

forms t-SNE both in output quality and running time [16]. This debate has been distorted so far by the inability to work with large neighbourhood affinity matrices. Seemingly, the captured global structure is just the result of an informative initialization and can hardly be credited to any intrinsic advantage of either of the algorithms [17]. We stress that t-SNE simultaneously optimizes both the local and the global structure in the data, and featuring more of the latter is just a matter of using large enough perplexities as the method prescribes. We show this by following the *retrieval information* approach presented in [23], where the authors introduce the *neighbour retriever visualizer* (NeRV). NeRV is a visualization method that identifies the cost of retrieving/missing neighbours from the HD/LD representations of the data with the fundamental trade-off between *precision* and *recall* in information retrieval. By extending their approach (see section S1 of the Supplementary Information), we reach a reformulation of the t-SNE cost function that explicitly states how t-SNE assesses data structure at the local and global scale,

$$KL\left(P\|Q\right) \equiv \mathbb{E}_{p_i}\left[KL\left(p_{\cdot|i}, q_{\cdot|i}\right)\right] + KL\left(p_{i|\cdot}, q_{i|\cdot}\right) \tag{1}$$

The distributions $p_{\cdot|i}$ and $q_{\cdot|i}$ in Eq. 1 are probabilistic models of *smoothed recall*, i.e. describe the probability of picking a data point from the neighbourhood of $\mathbf{x}_i$ in the HD and LD spaces, respectively. Analogously, the distributions $p_{i|\cdot} \equiv p_i$ and $q_{i|\cdot} \equiv q_i$ are probabilistic models of *global prevalence*, i.e. represent the average probability of picking data point $\mathbf{x}_i$ as a neighbour of any other point. $KL\left(\right)$ is the Kullback-Leibler divergence. Thus, the first term in Eq. 1, known as the *expected smooth recall*, is an average measure of the mismatch between the HD and LD representations of the neighbourhood of $\mathbf{x}_i$, weighted by the prevalence of $\mathbf{x}_i$. The second term is a measure of matching between the HD and LD models of prevalence, closely related to an index of global structure preservation. In summary, the t-SNE approach maximizes the *expected smoothed recall* prioritizing areas where local structure is more significant while simultaneously trying to preserve the global structure.

We illustrate this idea using a data set with a known structure (Fig. 1). The Sierpinski-3D data set is a graph representation of the well known Sierpinski 3D-triangle, where each row represents a node of the graph (a vertex of the structure) and each column is the *shortest-path-distance* to the rest of the nodes (https://sparse.tamu.edu/, [24], [25]). The Sierpinski-3D data

set is not challenging in size but represents a challenging fractal structure for a dimensional reduction algorithm. By scanning this data set across a wide range of perplexities (Fig. 1) we observe how the algorithm accurately resolves the fractal structure at each specific scale (Fig. 1 panels a to i, see also section S4.1 of the Supplementary Information). Interestingly, the embedding starts showing the complete structure for ppxs beyond 30% of the data (Fig. 1 f), an indication of the convenience of exploring high-ranged perplexities. Additionally, we include the kNP plots to depict a quantitative assessment of the embedding (Fig. 1, j, k). The kNP curves show how pt-SNE balances the prevalence of local and global structure signatures as ppx increases (Fig. 1, j) while the waving in each curve show how pt-SNE captures the inherent fractality of the object across different scales.

Speed/Accuracy trade-off

A fundamental parameter in the chunk&mix protocol is the thread-ratio ρ defining the proportion of data running in each partial t-SNE (see section Chunk&Mix parameters in Methods). Using low thread-ratios, that is, splitting the data into more and smaller chunks, is the key to achieving reasonable running times at large perplexities. However, as the data chunks become smaller, the amount of structural information contained in each of them is lower, leading to a quality loss in the visualization of the local structure. This balance between chunk size and visualization quality is the speed/accuracy trade-off in pt-SNE. As an example, we show the visualization of the Sierpinski-3D data set for $ppx = 1948$ (same as in Fig. 1 h) with decreasing thread-ratio $\rho = \{1.0, 0.67, 0.50, 0.40, 0.33, 0.25\}$ (Fig. 2 panels a to f, see also section S4.2 of the Supplementary Information). As no subset of this data set can adequately reflect the overall data structure, a decrease in the *thread-ratio* rapidly translates into a loss of information. In terms of kNP, a degradation of the local structure is clear (Fig. 2 g), but the global structure is preserved (Fig. 2 h). We also depict the speed/accuracy trade-off by plotting the running times and the logAUC versus the thread-ratio ranging from 1.0 down to 0.25 (Fig. 2 i). In general, the local accuracy loss will not be as noteworthy as in the Sierpinski-3D data set. However, if required, simple post-processing can efficiently restore the local scale structure, preserving the precision achieved at the global scales (section S3 of the Supplementary Information).

Parametric setup

Our parallel implementation of t-SNE demands a smooth parameter optimization avoiding too early cluster arrangements that may compromise the long run convergence among partial t-SNE solutions. To achieve such a smooth optimization pt-SNE shows a minimal parametric configuration where we drop *exaggeration* and we use auto-adaptive schemes for *momentum* and *learning-rate* (see subsection. Parametric configuration of the cost gradient in Methods). Furthermore, pt-SNE works well with random initialization. Therefore we are basically left with ppx (and the Barnes-Hut parameter θ which works well in general with the default value $\theta = 0.5$, [10]). Additionally, the chunk&mix protocol involves the *thread-ratio* ρ, given by way of two parameters: $threads$, defining the number of chunks of data, and $layers$, defining the degree of overlapping among threads. These two parameters determine the proportion of data running in each partial t-SNE, $\rho = layers/threads$, (see section Chunk&Mix parameters in Methods and section Speed/Accuracy trade-off in Results).

[19] explains how to achieve improved t-SNE visualizations that preserve the global structuring in the data using real UMI-based transcriptomic data, e.g. [26], and [27]. The main settings in [19] included PCA initialization, multi-scale affinities combining perplexities 30 and $n/100$, learning-rate $eta = n/12$ and exaggeration $\alpha = 12$. We run pt-SNE on the same data sets to check out our simple parametric setup involving random initialization, no exaggeration and auto-adaptive schemes for learning rate and momentum. Performing PCA on [26] data (Fig. 3 a) shows three well-separated groups corresponding to excitatory neurons (cold colors), inhibitory neurons (warm colors) and non-neural cells such as astrocytes or microglia (grey/brown colors). Performing separate PCA on these three subsets reveals further (local) structure in each of them ([19] Supplementary Material). [19] showed that using their settings, FIt-SNE preserved much of the global structure in the data, clustering the different classes/sub-classes of cells in a coherent pattern. In the case of [26] data (Fig.3 b), inhibitory neurons are well-separated into two groups, Pvalb/SSt-expressing (red/yellow) and Vip/Lamp5-expressing (purple/salmon)). In the case of [27] data (Fig:3 f), multiple clusters of amacrine cells (green), bipolar cells (dark-blue), and non-neural cells (classes from Mueller glia to microglia) are close together. Using pt-SNE, we achieve similar results by simply switching from high (Fig. 3 c, g) to low (Fig. 3 d, h) ppx values.

Of note, pt-SNE does not mix multi-scale affinities but, instead, shows the different scales by gradually increasing the ppx (see section S4.3 of the Supplementary Information). For instance, panels d and h show a similar level of local structure as reported in [17] (panels b and f) but a lower level of the global one. However, pt-SNE can achieve higher levels of global structure (panels c and g), up to the point of improving the global structure suggested by the PC plots (panels a and e).

Computational scaling

By breaking down the t-SNE into chunks, pt-SNE could potentially scale as much as the number of parallel t-SNE instances allowed by our hardware resources. On top of this, we designed pt-SNE to effortlessly work with high-performance computing (HPC) platforms using both intra and inter-node parallelization. However, two limitations arise: (i) decreasing the *thread-ratio* in excess, that is, reducing too much the length of the data chunks, can result in a loss of data structure information and visualization over-blurring (Fig. 2, see also section Speed/Accuracy trade-off in Results), and (ii) the chunk&mix scheme (alternating short t-SNE runs with mixing of the partial solutions) adds an extra computational time (inter-epoch time) required to pool all partial t-SNE outputs after an epoch has finished, and this inter-epoch time increases with the number of partial t-SNEs (see section Chunk&Mix parameters in Methods).

We run pt-SNE on the [28] data set with ppxs. 130, 653, 1301, 6531, 13061 and 65306, the latter equivalent to 5% of the data set size (Fig.4) to show that, pt-SNE scales computationally for biological large data sets of about this size and up to perplexity values far beyond a few hundreds. The [28] is a large transcriptomic data set ($n = 1306127$ brain cells from E18 mice) commonly used to assess large scale visualization procedures ([14, 19, 12]). The clustering labels for this data were derived by [14] using the Louvain clustering algorithm ([29]) and comprise 39 clusters. However, when visualized through dimensional reduction algorithms, these clusters show a significant overlap. Our analysis, with an augmented perplexity, offers an improved visualization in terms of compactness of the Louvain clusters (section S4.4.1 of the Supplementary Information) and in terms of kNP (both log and linear, Fig.4 c, d), which could bring more accurate insights about the data. We run this analysis on an HPC platform using MPI and

130 cores (i.e. thread-ratio $\rho = 0.015$, data chunks of 20094 observations). On this order of magnitudes, it took about 300 min (Fig.4 f) to complete the whole process (i.e. computing the bandwidths β_i and running the gradient descent). We also run FIt-SNE for values of ppx ranging from 32 to 320 with quite similar results (section S4.4.2 of the Supplementary Information). We could not explore higher ranges because the amount of memory needed to run FIt-SNE with $ppx = 320$ was already above 400GB. Of note, the ppxs in pt-SNE and FIt-SNE are not equivalent: in pt-SNE, the ppx must be higher to get similar results because the neighbourhood size is smaller (see section Chunk&Mix parameters in Methods). In terms of running times, FIt-SNE ($\mathcal{O}(n)$) is obviously faster than pt-SNE (indeed a BHt-SNE $\mathcal{O}(n \log n)$) but FIt-SNE is extremely dependent on ppx while pt-SNE is almost independent (Fig.4 f). FIt-SNE performed extremely well at capturing global structure from the very lowest value of ppx, only surmounted by pt-SNE at much larger values of ppx (Fig.4 e). However, pt-SNE embedding shows higher sensitivity to ppx than FIt-SNE (note the increasing linAUC for increasing ppxs, dashed red line in panel e). Therefore, pt-SNE highlights much better the differences of data organization across scales. The drawback is a loss of local structure (solid red line in panel e) due to the small size of the data chunks, (i.e. low thread-ratio $\rho = 0.015$).

We also compared pt-SNE and FIt-SNE on the *Primes-1M* data set [30]. This data set is a structured representation of the first 10^6 integers based on their prime factor decomposition, conforming to a highly sparse matrix with 78498 dimensions (the set of primes lower than 10^6) where the values are the prime factorization powers. We note some differences: (i) in Williamson's work, any factor is represented as either a zero or a one, just indicating divisibility by that factor, while we use the actual power for each factor to have unique representations for each integer; (ii) Williamson computes cosine-similarity based affinities, while we use euclidean-distance based affinities.

We run pt-SNE on the *Primes-1M* dataset with ppxs 1000, 5000, 10000, 20000 and 50000, the latter equivalent to a neighbourhood size of 5% of the data set size. We run pt-SNE using MPI and 100 cores with 4GB/core (thread-ratio $\rho = 0.02$). We also run FIt-SNE on *Primes-1M* data set with ppxs 50, 100, 200, 300 and 400, the latter equivalent to 0.04% of the data set size. As Fit-SNE in [31] does not have support for sparse matrices, for this dataset we used FIt-SNE from OpentSNE [32], using a single node with 20 cores and 400GB. We give further details of

both experiments in sections S4.5.1 and S4.5.2 of the Supplementary Information.

The *Primes-1M* showed a complex data structure with strong hierarchical relations between its components not equally depicted by both algorithms (Fig. 5). The differences observed between the pt-SNE and the Fit-SNE visualizations are mainly due to a different computation of the bandwidths B_i (see sections S2.2 and S2.3 of the Supplementary Information). Also, the two algorithms showed unequal behaviour as we moved from local to global scales. On the one hand, FIt-SNE returned a quite neat depiction of the structure, capturing much of both the local and the global structure, even from the lowest value of ppx (5 c). Increased ppxs slightly improved the global structure without any loss of information at the local level (Fig. 6 c, d). However, the embedding landscape did not change significantly overall (5 c, d). On the other hand, beyond $ppx = 10000$ pt-SNE starts revealing similar structures as those shown by FI-tSNE, but adding the characteristic blurring due to the chunk&mix protocol (Fig. 5 a, b). Noteworthy, pt-SNE embedding outputs were much more sensitive to ppx than the ones derived from FIt-SNE (compare Fig. 6 a, b) with Fig. 6 c, d), revealing novel details at high perplexities not observed with FI-tSNE. We can also assess the higher sensitivity of pt-SNE landscapes to ppx by looking at the linAUC curves (global structure) in Fig. 6 e (doted-lines). By means of pt-SNE, we *gradually* evolve the landscape from local (Fig. 5 a) to global structure (Fig. 5 b) by simply increasing the ppx (section S4.5.1 of the Supplementary Information), and this is just the aim of pt-SNE. In terms of running times, pt-SNE shows a lower dependence on ppx and lower running times than FIt-SNE as implemented in OpentSNE (Fig. 6 f).

Both examples in this section, the first from the biological domain, the second from the mathematical one, illustrate the main features of pt-SNE: (i) independence of running times to ppx, enabling global data structure analysis beyond the state-of-the-art (ii) high sensitivity of embedded landscapes to ppx, gradually depicting data organization across structuring scales, and (iii) blurring at local scales due to the chunk&mix protocol, highlighting an unavoidable trade-off between computational speed and accuracy. In section S3 of the Supplementary Information, we explain how simple post-processing can efficiently restore the accuracy of the local structure while keeping the global data structure obtained with pt-SNE (Figures S1 and S2).

Discussion

As a first principle, methods for data exploratory analysis should be as simple and flexible as possible. Nonetheless, it is not easy to accommodate and fulfill these principles, hence, key conceptual and operational limits exist in current visualization methods. FIt-SNE [13] and UMAP [15] are both outstanding algorithms that can perform the gradient-descent on large data sets within very reasonable times but show complex parameterizations and computational limitations to explore large scale data structures. Focused on t-SNE, our work aims at breaking through these two limitations.

At the origin of the parametric complexity of t-SNE there is a contrived trickery in the computation of the gradient descent intended to force a quick convergence and a neat depiction of the structure. We noted that the gradient descent equation suggests by itself an auto-adaptive scheme for the learning-rate (Eq. 9) that smoothly drives the embedding to an optimal solution. Based on this observation, we showed that a simple parametric setup, using random initialization, auto-adaptive schemes for learning-rate and momentum and discarding the use of exaggeration can achieve similar results as those reported in the related literature.

The limitation at exploring data across scales might stem from the t-SNE heuristic itself or the algorithmic implementation of the heuristic. We showed that the t-SNE heuristic simultaneously optimizes both local and global structure (Eq. 1) and that the final embedding is a balanced visualization of one or the other only depending on the value of ppx, just as the heuristic prescribes. Thus, the actual limitation to explore data across scales is purely an implementation issue stemming from both data set size and the neighbouring size parameter (i.e. ppx). To overcome this limitation, we introduced a chunk&mix protocol as a parallelized re-implementation of t-SNE. The underlying assumption is that large data sets convey a lot of redundant evidence so that random subsets of data comprehensively reflect the structure in the data. Therefore, we can approximate the solution by running instances of the algorithm on random subsets of the data and combining the partial results. This approach extends the state-of-the-art, overcoming the impossibility of computing the affinity matrix for large values of ppx, and allows scanning data structure across a wide range of scales. The downside is a loss of detail in the definition of the local structure, the loss increasing as the data chunks are shorter or the redundancy in the

data is less.

As a proof of concept, we developed pt-SNE, a parallel version of BHt-SNE, and we showed that the chunk&mix protocol converges to good global embedding. Hence, we expect the same approach to apply to other visualization algorithms (e.g. FIt-SNE or UMAP). pt-SNE runs efficiently on HPC platforms up to 10^6 observations. To scale beyond this order of magnitude, we can either increase the number or the size of the data chunks. Increasing the number of data chunks may result in too small data subsets and excessive loss of information at local scales. Still, we could overcome this accuracy loss by combining a large ppx pt-SNE with a subsequent dimensional reduction algorithm (e.g. FIt-SNE) with lower ppx, using the former to sketch the global structure and the latter to refine the local one. Increasing the size of the data chunks results in a quadratic increase of running time (being pt-SNE based on BHt-SNE), so it is not a practical solution. However, algorithms like FIt-SNE and UMAP around $\mathcal{O}(n)$ could help improve this limitation. Therefore, a promising solution is to apply the chunk&mix approach with these much faster algorithms, thus boosting the computation of the gradient descent and allowing larger thread sizes within reasonable running times.

Methods

The pt-SNE runs multiple instances (independent threads) of the t-SNE on different chunks of data (partial t-SNEs). The algorithm starts by randomly allocating the data points in the low-dimensional space (a 2D half-unit disk, i.e. of radius $r = 0.5$). Afterwards, a cyclic scheme arranged into *epochs* optimizes the embedding (Fig. 7), each epoch involving: shuffling and chunking the data set indexes, exporting a chunk of indexes to each thread, running the partial t-SNEs with a short number of iterations, and pooling the solutions of the partial t-SNEs into a global embedding.

Chunk&Mix parameters

The parameters that define the chunk&mix scheme are the following:

threads The number of threads z is the number of partial t-SNEs that will run. The pt-SNE splits the data set into this number of elementary chunks, so that, the larger the number of threads, the faster the computation of the final solution. Note that the number of threads can be higher than the number of physical cores available (what is known as multi-threading). Using multi-threading can yield further reductions in computational time.

layers In the most simple scheme (Fig. 7), each thread runs a single chunk of data. However, the key to convergence to a common solution is to impose some overlap among data chunks. Instead of running single chunks of data, threads are cyclically chained so that each thread includes at least two chunks of data. Thus, each thread shares one-half of the data points with the previous thread and the other half with the next one. The number of *layers* sets the degree of overlapping. We name this parameter *layers* because the overlapping makes each data-point take part in at least two partial t-SNEs and, after pooling all partial solutions, we will have at least two layers of global solutions. As an example, setting $threads = 5$ and $layers = 3$ (Fig. 8) the pt-SNE pools chunks 1, 2, and 3 into thread 1, chunks 2, 3 and 4 into thread 2, and so on, up to chunks 5, 1 and 2 into thread 5).

thread-ratio and thread-size The *thread-ratio* is defined as $\rho = layers/threads$ and determines the thread-size $\nu = \rho\, n$. Being t-SNE of order quadratic to the size of the data set, making $\nu \ll n$ overcomes the unsuitability of the t-SNE algorithm for large-scale data sets. The thread-ratio represents a trade-off between accuracy and computational time. The closer is the thread-ratio to 1, the more robust and comprehensive is the solution, but the larger the computational cost. However, for large data sets ($n > 10^4$) where the redundancy assumption holds, pt-SNE yields a good global solution even with values of ρ as low as 0.01.

epochs and iterations The number of epochs is set to $2\, log\, n^2$ (where n is the data set size) and the number of iterations per epoch is set to $2\, log\, \nu^2$ (where ν is the *thread-size*). We empirically determined that these settings allow reaching a stable solution. Nevertheless, the user can restart the process and run it for additional epochs if the cost function does not show a flat line (Fig. 9 b, c). The epoch running time depends on the number of iterations performed by the threads, plus an inter-epoch time for the master process to pool the solutions, mixing them

and sending new chunks to the workers.

Partial t-SNEs

The t-SNE algorithm starts by transforming similarities (usually given as pairwise euclidean distances) into a probability distribution known as the *affinity matrix*. Computing the whole affinity matrix is prohibitive for large data sets. The pt-SNE splits the data set into chunks and runs parallel instances of the t-SNE with subsets of the data, each partial t-SNE computing a much smaller affinity matrix. When splitting a data set X into z chunks, we denote the subset of data in each partial t-SNE as X^k, $1 \leq k \leq z$, and we denote the neighbourhood of i in X as N_i and the neighbourhood of i in X^k as N_i^k. Thus, in each partial t-SNE, we compute the similarities in the input and output spaces as follows:

Similarities in the input (high dimensional) space, $\mathcal{X} \in \mathcal{R}^m$ The similarity between observations x_j and x_i, expressed as $\|x_i - x_j\|^2$, is converted into the conditional probability $p_{j|i}$ given by a Gaussian kernel centered at x_i,

$$p_{j|i} = \frac{\exp\left(-\beta_i \left\|x_i - x_j\right\|^2\right)}{\sum_{k \neq i} \exp\left(-\beta_i \left\|x_i - x_k\right\|^2\right)} \,, \quad i \in X^k,\ j \in N_i^k$$

$$p_{j|i} = 0 \,, \quad i \in X^k,\ j \notin N_i^k$$

with precision $\beta_i = 1/\left(2\,\sigma_i^2\right)$. Decreasing values of β_i induce a probability distribution of increasing entropy $H\left(p_{j|i}\right)$ and increasing *perplexity*, defined as,

$$ppx\left(p_{j|i}\right) = 2^{H\left(p_{j|i}\right)} \,, \quad i \in X,\ j \in N_i$$

We stress that β_i are computed globally, i.e. for the whole data set. The maximum perplexity is equal to $n - 1$ (where n is the data set size), corresponding to $\beta_i = 0$ and a uniform affinity distribution. In a preliminary step, t-SNE computes the values β_i that result in a fixed perplexity for all x_i. We describe the procedure to find β_i in section S2 of the Supplementary Information. Computing similarities based on perplexities is a powerful transformation because it allows

defining affinities in terms of spatial proximity without explicitly referring to any actual distance. The usual interpretation of perplexity is the number of neighbours to be picked: low perplexity will unveil the local structure in the data, whereas high perplexity will enhance the emergence of global structuring. Thus, it is fundamental to tune perplexity according to our requirements or, alternatively, explore data structuring across a range of perplexities. Afterwards, each partial t-SNE computes a symmetric joint probability given by,

$$p_{ij} = p_{ji} = \frac{p_{j|i} + p_{i|j}}{2\nu}$$

where ν is the thread-size, thus $\sum_{i,j} p_{ij} = 1$, and $\sum_j p_{ij} > \frac{1}{2\nu}$, $\forall\, x_i$, so that each data point plays its role in the embedding process [8].

To reduce the complexity in the computation of attractive forces, [10] proposed to use a neighbourhood size of $N_i = 3\,ppx$, dramatically decreasing the size of the affinity matrix. To find the nearest neighbours' sets, standard implementations of t-SNE make use of fast approximations like *vantage point trees* [33] or ANNOY [34] and, afterwards, determine the local bandwidths and compute the affinity matrix. In pt-SNE, the partial t-SNEs run a different chunk of data at each new epoch, thus requiring the recomputation of the affinity matrices. We alleviate this process by precomputing the global neighbourhoods N_i and the bandwidths $\beta_i\,(N_i)$, and we pull out the distance of the furthest neighbour $L_i = max\{Lij \mid j \in N_i\}$. The values B_i and L_i are shared to all processes so that each one can use it to determine the neighbourhoods $N_i^k = \{j \in X^k \mid L_{ij} \ll L_i\}$ and compute the partial affinity matrix. This process adds a short inter-epoch computation time. On average, $N_i^k \approx \rho\, 3\, ppx$, thus, for low values of ρ, the number of forces acting on a data point is much lower in pt-SNE than in standard t-SNE. Therefore, as a general rule, we will use higher perplexities in pt-SNE to have equivalent outputs.

Similarities in the output (low-dimensional) space, $\mathcal{Y} \in \mathcal{R}^d$, $d \in \{2, 3\}$ The similarities between mapped data points y_j and y_i, also expressed as $\|y_i - y_j\|^2$, are treated differently. A well-known issue of embedding processes is the so-called *crowding problem* (i.e. a surface at a given distance from a data point in a high-dimensional space can enclose more data points than those that fit in the corresponding low-dimensional area [8]). This problem is alleviated using a

heavy-tailed distribution to represent affinities in the low dimensional space, namely a Cauchy distribution (i.e. a t-Student distribution with one degree of freedom). Therefore, we define the joint probabilities q_{ij} as,

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}, \quad i, j \in X^k \tag{2}$$

Cost function

t-SNE uses a gradient descent method to find a low-dimensional representation of the data that minimizes the mismatch between p_{ij} and q_{ij}. The cost function is defined as the Kullback-Leibler divergence between both distributions ([8]),

$$C = KL\left(P \| Q\right) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3}$$

with a gradient with respect to the low-dimensional mapped positions given as ([8]),

$$\frac{\delta C}{\delta y_i} = 4 \sum_j \left(p_{ij} - q_{ij}\right)\left(y_i - y_j\right)\left(1 + \|y_i - y_j\|^2\right)^{-1} \tag{4}$$

The mode the gradient descent operates is clear by noting $L_{ij} \equiv \frac{(y_i - y_j)}{(1 + \|y_i - y_j\|^2)}$, and writing Eq.4 as,

$$\frac{1}{4}\frac{\delta C}{\delta y_i} = \sum_j p_{ij}\, L_{ij} - \sum_j q_{ij}\, L_{ij} = \mathcal{F}_{attr,i} - \mathcal{F}_{rep,i} \tag{5}$$

L_{ij} depends only on the pair-wise distances in the embedding, and the computation of the gradient represents a balance between attractive and repulsive forces exerted over each data point by all the others. On the one hand, high affinities in the HD space embedded as long distances in the LD space result in strong attractive forces, and low affinities in the HD space embedded as close distances result in weak attractive forces. On the other hand, repulsive forces monotonically decrease as a function of the embedding distance. Thus, the final embedding positions correspond to the point of equilibrium where attractive and repulsive forces are optimally balanced.

In pt-SNE, the gradient descent is performed independently at each partial t-SNE, and the global cost function is computed as an average of the cost functions in each partial t-SNE,

$$C = \frac{1}{z}\sum_k C^k = \frac{1}{z}\sum_k KL\left(P^k \| Q^k\right)$$

Finite affinity mass t-SNE holds an implicit dependence on the size n of the data set. The reason is that the joint affinity distribution has a finite amount of probability mass to be allocated among all pairwise distances, the latter growing with $n\ (n-1)$. Therefore, as n grows, affinities will be lower on average, tending to uniformity for the limiting case $n \to \infty$. This loss in discriminating power generates undesired side effects that we generically call the *finite affinity mass* problem.

Pseudo-normalized cost function A first effect of the *finite affinity mass* is that the cost function (Eq. 3) holds itself an implicit dependence on n, and makes it difficult to objectively interpret its value. Taking into account that p_{ij} and q_{ij} decrease on average with $n\ (n-1)$, we see that this dependence amounts to,

$$\begin{aligned} \langle C \rangle &\propto -\log\langle q_{ij}\rangle \sum_{i,j} p_{ij} \\ &\propto \log\left(n\ (n-1)\right) \end{aligned} \tag{6}$$

where we have dropped the term $\sum_{i,j} p_{ij} \log p_{ij}$, which is constant along the gradient descent.

The expression in Eq. 6 is the cost of a uniform distribution of affinities, i.e. the cost of a uniform embedding of $n\ (n-1)$ pairwise distances, expressing that all data points are equally similar. While it is not feasible to arrange $n\ (n-1)$ uniform pairwise distances in 2D, such a uniform distribution constitutes the worst possible embedding with respect to P, be P what it may. Thus, Eq. 6 is an upper bound for $KL\left(P\|Q\right)$ and it makes sense to define a pseudo-normalized cost function as,

$$C = -\frac{\sum_{i,j} p_{ij} \log q_{ij}}{\log\left(n\ (n-1)\right)} = -\frac{H\left(P,Q\right)}{H\left(P,U\right)} \tag{7}$$

In terms of information theory, this is the *normalized cross-entropy* of distributions P and Q, that is, the average cost of coding P as Q relative to the worst-case cost, which is the cost of a uniform embedding of n $(n-1)$ pairwise distances. Also, we can interpret this normalization factor as an expression of the loss in discriminating power due to the *finite affinity mass* which, intuitively, we could approach as the relative increase of entropy of the affinity distribution, amounting to $\log(n(n-1))$. This pseudo-normalized expression results in a value close to 1 for a random initial mapping, thus helping to make sense of the gradient descent and assess the stability of the final output.

Parametric configuration of the cost gradient

pt-SNE starts with a random embedding in a 2D half-unit disk (i.e. of radius r = 0.5) drawn from an isotropic Gaussian and updates embedding positions y_i using the following expression,

$$\begin{aligned} y_i^t &= y_i^{t-1} + \Delta y_i^t \\ \Delta y_i^{d,t} &= \mu^t \Delta y_i^{d,t-1} - 4\,\eta^t \sum_{j \in nN_i} \left(\alpha^t\, p_{ij} - q_{ij}^t\right) \frac{l_{ij}^{d,t}}{1 + \left(L_{ij}^2\right)^t} \end{aligned} \tag{8}$$

where t, d are indicators of current iteration and embedding dimension respectively, μ is the *momentum*, η is the *learning rate* and α is the *exaggeration fator*. Also, we recall that p_{ij} and q_{ij} are the affinities in the HD/LD spaces, L_{ij} is the pairwise distance in the embedding space and l_{ij}^d is the component of Lij along d.

Learning-rate The learning-rate η has a strong impact on convergence speed and it is not clear from the literature how to set its value: the default in most t-SNE implementations is $\eta = 200$, while [14] recommends increasing it to 1000 or [12] suggests the value $\eta = n/12$. In pt-SNE, we use an auto-adaptive scheme given by,

$$\eta_i^t = 2\left(d^t + \frac{1}{d^t}\right) \log\left(\nu\, N_i^k\right)\, g_i^t \tag{9}$$

where d^t is the diameter of the embedding at iteration t, and where we note the following:

- While the size of the embedding clearly changes along the optimization, Eq. 8 seems to be lacking a reference size for the distance factors L_{ij} and l_{ij}, thus suggesting,

$$\eta^t \propto \frac{1+(d^t)^2}{d^t} = d^t + \frac{1}{d^t}$$

The factor $\left(d^t + \frac{1}{d^t}\right)$ renders Eq.8 independent of the size of the embedding and, interestingly, yields equally increasing learning-rates at both sides of $d = 1$ (Fig. 9 a) i.e. either expanding or compressing embedding, the latter occurring for large ppx. Furthermore, if we specifically set

$$\eta^t \propto 2\,\frac{1+(d^t)^2}{d^t} = \frac{1+(d^t)^2}{r^t} \tag{10}$$

where r is the radius of the embedding, then for $d = 1$ we have $\eta = 4$, so that we make sense out of factor 4 in the gradient expression (Eq. 4) as the learning-rate corresponding to a half-unit disk embedding (i.e. $r = 0.5$). Therefore we can drop the factor 4 from Eq. 8.

- $\log\left(\nu\, N_i^k\right)$ compensates the effect of the *finite affinity mass* problem rendering Eq.8 independent of the size of the affinity matrix operating on each partial t-SNE, actually given by $\nu\, N_i^k = (\rho\, n)\ (\rho\, 3\, ppx)$.

- g_i^t is an acceleration factor given as a point-wise step size update by means of the Jacobs adaptive learning rate scheme [20] which is indeed implemented in BHt-SNE [10]. This factor increases the learning rate in directions in which the gradient is stable, anticipating position updates that are likely to occur in the next steps,

$$
\begin{aligned}
\text{init:}\quad & g\,(i,1) = 1.0 \\
\text{if}\quad & \text{gradient\_direction}\,(i,t) == \text{gradient\_direction}\,(i,t-1) \\
& g\,(i,t)\ += 0.1\,\text{gain} \\
\text{else}\quad & \\
& g\,(i,t)\ *= (1 - 0.1\,\text{gain})
\end{aligned}
$$

with $gain = 2.0$ by default.

In summary, our auto-adaptive scheme for the learning-rate (Eq.9) operates in favor of a stable gradient descent. On the initial stages of the optimization, the mismatch between p_{ij} and q_{ij} will likely be large but the embedding size is small, thus the learning-rate will mitigate the impact of the strong attraction/repulsion forces originated during this critical moment. Along the course of the optimization, the size of the embedding will get larger, increasing the learning rate to compensate the decreasing attraction/repulsion forces.

Momentum Using momentum μ speeds up the gradient descent but might blow up the embedding size on the final steps. In this final stage, the learning rate is high because the embedding has grown large, but the attractive/repulsive forces are almost balanced hence the position updates should be minor. Therefore adding momentum is risky. Because of this, we use a decaying momentum of the form,

$$\mu^t = 0.8\,\left(1 - \frac{e}{\epsilon}\right)^2 \tag{11}$$

where e stands for the current epoch and ϵ is the total number of epochs.

Exaggeration factor Using the exaggeration factor α is likely to be detrimental in pt-SNE. Forcing an early arrangement of clusters in each thread might depict a more fragmented global solution at the early stages of the gradient descent, which is unlikely to be solved in subsequent epochs. Therefore we dismiss using exaggeration in pt-SNE.

We illustrate the effect of the parametric setup described above in the gradient descent (Fig.9 b, c). We note that a plain learning-rate (Eq.10) yields a very smooth decay of the embedding cost (Fig. 9 b, solid lines) with a strong dependence on the size of the data set n. Including the acceleration factor $g(i,t)$ has a massive impact on the gradient descent, dramatically reducing the number of iterations needed to reach a stable embedding (Fig. 9 c, compare the range of the x-axis with panel b). Using a decaying momentum contributes to faster reaching a stable solution, although in general, the final embedding is not so good (Fig.9 c dashed lines).

Data availability

The data sets used in this work and the necessary code to reproduce the results are available at https://doi.org/10.5281/zenodo.5772287. The code for the examples is also available at https://rpubs.com/bigMap/. The latter provides immediate access to individual HTML documents explaining each example.

Code availability

The pt-SNE algorithm is part of the **bigMap** R-package. We used version bigMap_4.6.0, available at https://doi.org/10.5281/zenodo.5779186. We run all our analysis on the HPC platform at the Computational Biology Lab (CEAB-CSIC) (Table 1).

Acknowledgments

We acknowledge members from the Theoretical and Computational Ecology Group to provide comments on previous versions of the MS, and insights as beta users of the bigMap R-package. This work was supported by the Spanish Ministry (MINECO, Grant CGL2016-78156-R), the Max Planck Institute for Ornithology (MPIO, Germany), and the EU Horizon 2020 program Grant agreement VEO No.874735. The high-performance computation cluster at the Computational Biology Lab (CEAB-CSIC) was supported by the Spanish Ministry (MINECO, CSIC13-4E-1999).

nodes	model	CPU	cores	fr.(MHz)	RAM
5	PowerEdge R420	Intel(R) Xeon(R) E5-2450L	16	1800	161G
7	PowerEdge R430	Intel(R) Xeon(R) E5-2650	20	2300	193G
1	PowerEdge R815	AMD Opteron(tm) 6380	64	2500	515G

Table 1: **High-performance computing cluster at the Computational Biology Lab (CEAB-CSIC)**. Technical specifications.

Author information

The authors contributed equally in designing and writing the paper. J.G developed pt-SNE and performed the analysis.

Competing interests

The authors declare no competing interests.

References

[1] Gisbrecht A, Hammer B. Data Visualization by Nonlinear Dimensionality Reduction. Wiley Int Rev Data Min and Knowl Disc. 2015;5(2):51–73. doi:10.1002/widm.1147.

[2] Hotelling H. Analysis of a complex of statistical variables into principal components. J Educ Psych. 1933;24.

[3] Torgerson WS. Multidimensional scaling: I. Theory and method. Psychometrika. 1952;17:401–419.

[4] Sammon JW. A Nonlinear Mapping for Data Structure Analysis. IEEE Trans Comput. 1969;18(5):401–409. doi:10.1109/T-C.1969.222678.

[5] Tenenbaum JB, de Silva V, Langford JC. A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science. 2000;290(5500):2319.

[6] Belkin M, Niyogi P. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. Advances in Neural Information Processing Systems. 2001;14:585–591.

[7] Wattenberg M, Viégas F, Johnson I. How to Use t-SNE Effectively. Distill. 2016;doi:10.23915/distill.00002.

[8] vdMaaten L, Hinton GE. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research. 2008;9:2579–2605.

[9] vdMaaten L, Postma E, Van den Herik J. Dimensionality reduction: a comparative review. J Mach Learn Res. 2009;10:66–71.

[10] vdMaaten L. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research. 2014;15:3221–3245.

[11] Ulyanov D. Muticore-TSNE; 2016. https://github.com/DmitryUlyanov/Muticore-TSNE.

[12] Belkina A, Ciccolella CO, Anno R, Halpert RL, Spidlen J, Snyder-Cappione J. Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. Nature Communications. 2019;10. doi:10.1038/s41467-019-13055-y.

[13] Linderman GC, Rachh M, Hoskins JG, Steinerberger S, Kluger Y. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. Nature Methods. 2019;16(3):243–245. doi:10.1038/s41592-018-0308-4.

[14] Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome biology. 2018;19(1):1–5.

[15] McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv e-prints. 2018;.

[16] Becht E, McInnes L, Healy J, Dutertre C, Kwok I, Ng LG, et al. Dimensionality reduction for visualizing single-cell data using UMAP. Nature Biotechnology. 2018;37:38–44.

[17] Kobak D, Linderman GC. UMAP does not preserve global structure any better than t-SNE when using the same initialization. bioRxiv. 2019;doi:10.1101/2019.12.19.877522.

[18] Lee J, Peluffo D, Verleysen M. Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. Neurocomputing. 2015;169. doi:10.1016/j.neucom.2014.12.095.

[19] Kobak D, Berens P. The art of using t-SNE for single-cell transcriptomics. bioRxiv. 2018;doi:10.1101/453449.

[20] Jacobs RA. Increased rates of convergence through learning rate adaptation. Neural Networks. 1988;1(4):295–307. doi:https://doi.org/10.1016/0893-6080(88)90003-2.

[21] Cao Y, Wang L. Automatic Selection of t-SNE Perplexity. CoRR. 2017;abs/1708.03229.

[22] Im DJ, Verma N, Branson K. Stochastic Neighbor Embedding under f-divergences. CoRR. 2018;abs/1811.01247.

[23] Venna J, Peltonen J, Nybo K, Aidos H, Kaski S. Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization. J Mach Learn Res. 2010;11:451–490.

[24] Hu Y. Efficient and High Quality Force-Directed Graph Drawing. Mathematica Journal. 2005;10:37–71.

[25] Kruiger JF, Rauber PE, Martins RM, Kerren A, Kobourov S, Telea AC. Graph Layouts by t-SNE. Comput Graph Forum. 2017;36(3):283–294. doi:10.1111/cgf.13187.

[26] Tasic B, Yao Z, Graybuck L, Smith K, Nguyen TN, Bertagnolli D, et al. Shared and distinct transcriptomic cell types across neocortical areas. Nature. 2018;563. doi:10.1038/s41586-018-0654-5.

[27] Macosko E, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, et al. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. Cell. 2015;161:1202–1214. doi:10.1016/j.cell.2015.05.002.

[28] 10x Genomics. 1.3 Million Brain Cells from E18 Mice, Single Cell Gene Expression Dataset by Cell Ranger 1.3.0, (2017, February 9); 2017.

[29] Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008;2008(10):P10008. doi:10.1088/1742-5468/2008/10/p10008.

[30] Williamson J. What do numbers look like?; 2019. Available from: `https://johnhw.github.io/umap_primes/index.md.html`.

[31] Linderman G. FFT-accelerated Interpolation-based t-SNE (FIt-SNE); 2019. Available from: `https://github.com/KlugerLab/FIt-SNE`.

[32] Poličar PG, Stražar M, Zupan B. openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding. bioRxiv. 2019;doi:10.1101/731877.

[33] Nielsen F, Piro P, Barlaud M. Bregman Vantage Point Trees for Efficient Nearest Neighbor Queries. In: Proceedings of the 2009 IEEE International Conference on Multimedia and Expo. ICME'09. IEEE Press; 2009. p. 878–881.

[34] Bernhardsson E. Annoy: Approximate Nearest Neighbors in C++/Python; 2018. Available from: `https://pypi.org/project/annoy/`.

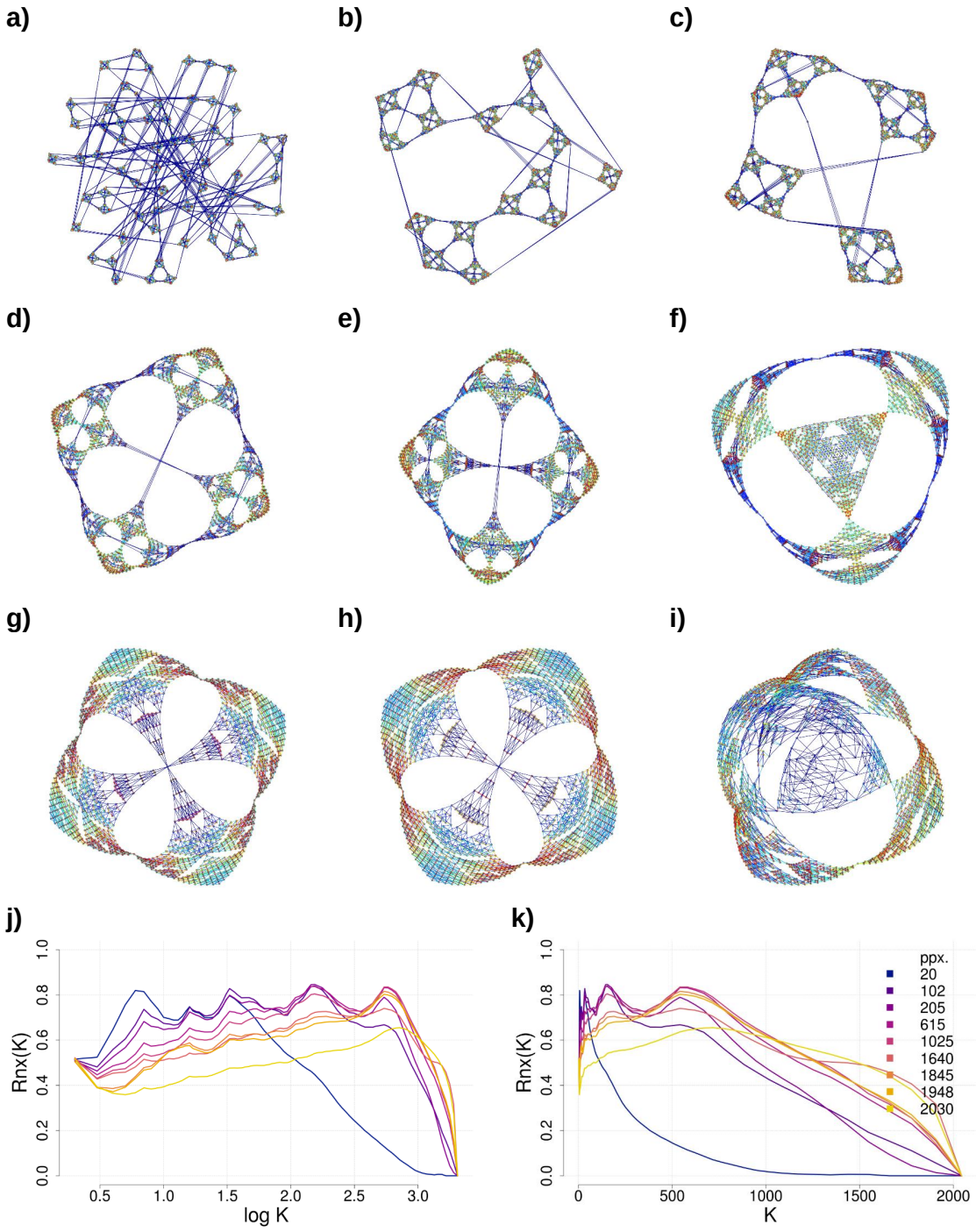

Figure 1: **Global/Local trade-off** panels a) to i) pt-SNE visualization with ppx values corresponding to 0.01, 0.05, 0.10, 0.30, 0.50, 0.80, 0.90, 0.95 and 0.99 of the data set size. Colors depict relative pair-wise distances in original space (red:closer, blue:farther); j) log and k) linear kNP.

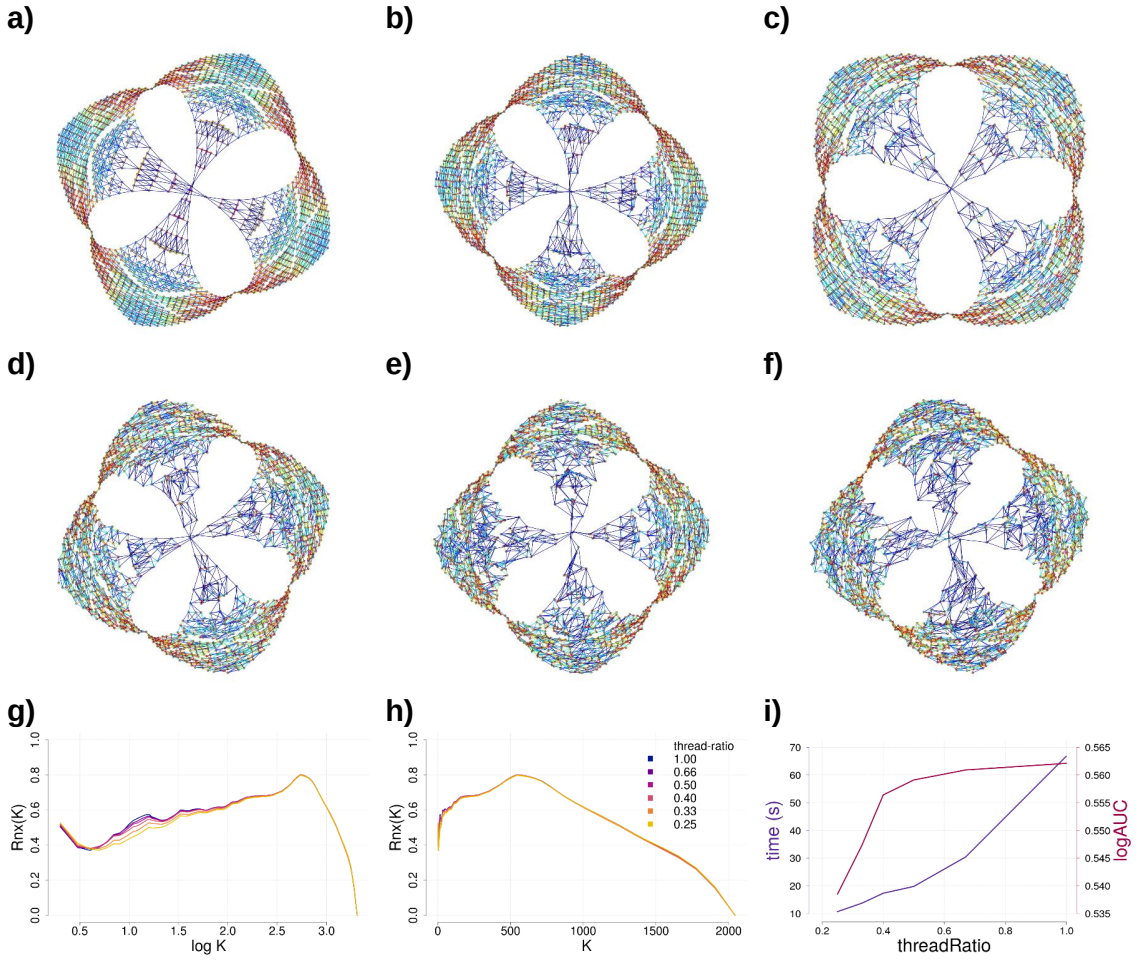

Figure 2: **Speed/Accuracy trade-off** Panels a) to f) ptSNE visualization for $ppx = 1948$ and thread-ratio $\rho = \{1.0, 0.67, 0.5, 0.40, 0.33, 0.25\}$; the local structure deteriorates as the thread-ratio decreases. Colors depict relative pair-wise distances in original space (red:closer, blue:farther); g) log and h) linear kNP; i) running-time and $logAUC$ versus thread-ratio.

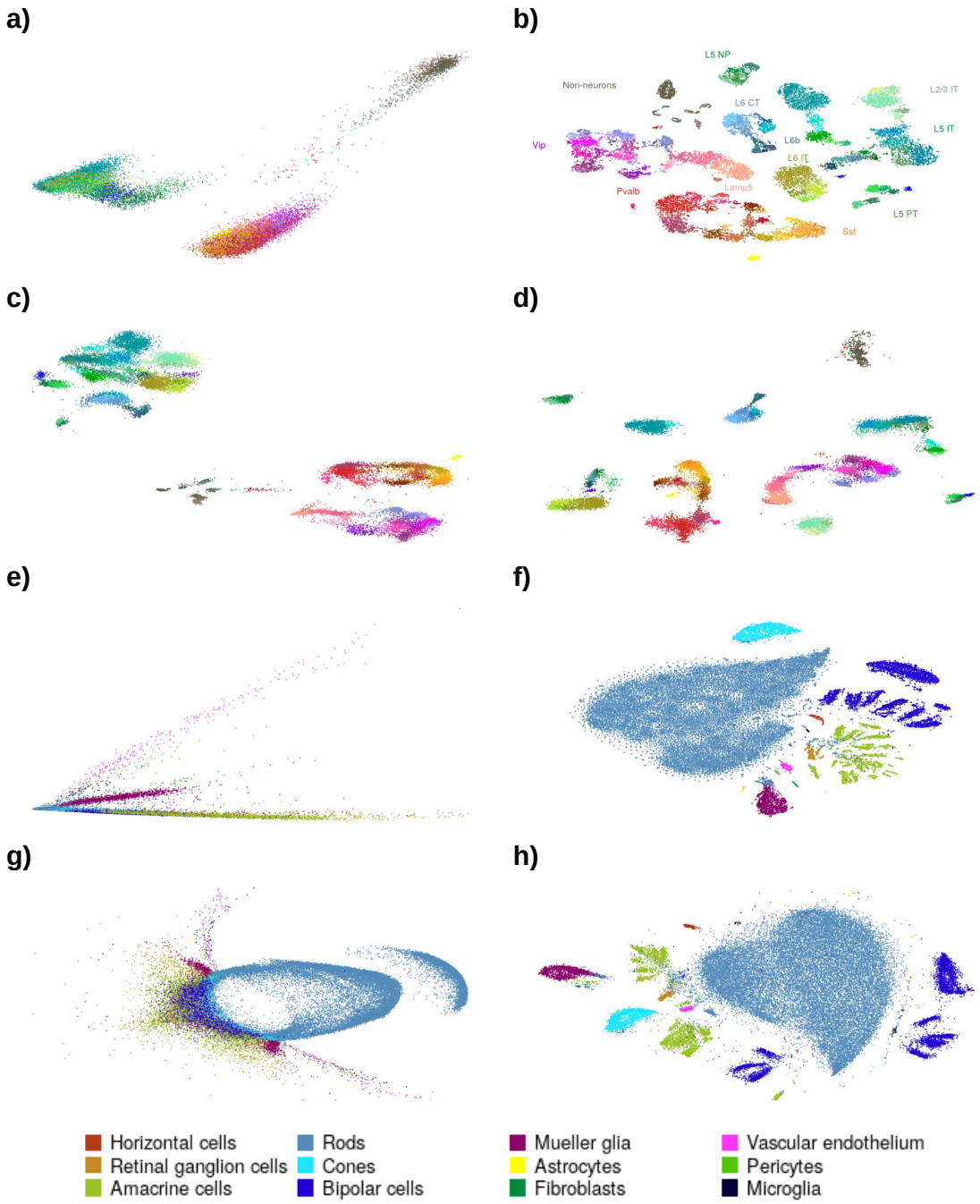

Figure 3: **UMI-based transcriptomic data** Panels a to d: [26] $n = 23822$ cells from adult mouse cortex classified into 133 clusters. Warm colors correspond to inhibitory neurons, cold colors correspond to excitatory neurons, brown/gray colors correspond to non-neural cells. Panels e to h: [27] $n = 44808$ cells from the mouse retina classified into 39 classes. Amacrine cells (green) and bipolar cells (dark blue) comprise 8 and 21 subclasses respectively, not shown in the legend, but visible in the embedding. Panels a, e: Depiction of the two first principal components of the data. Panels b, f: FIt-SNE embedding as shown in [19] (multi-scale affinities combining ppxs 30 and $n/100$). Panels c, g: pt-SNE embedding with high perplexities (20% and 40%). Panels d, h: pt-SNE embedding with low perplexities (0.01% and 0.005%).

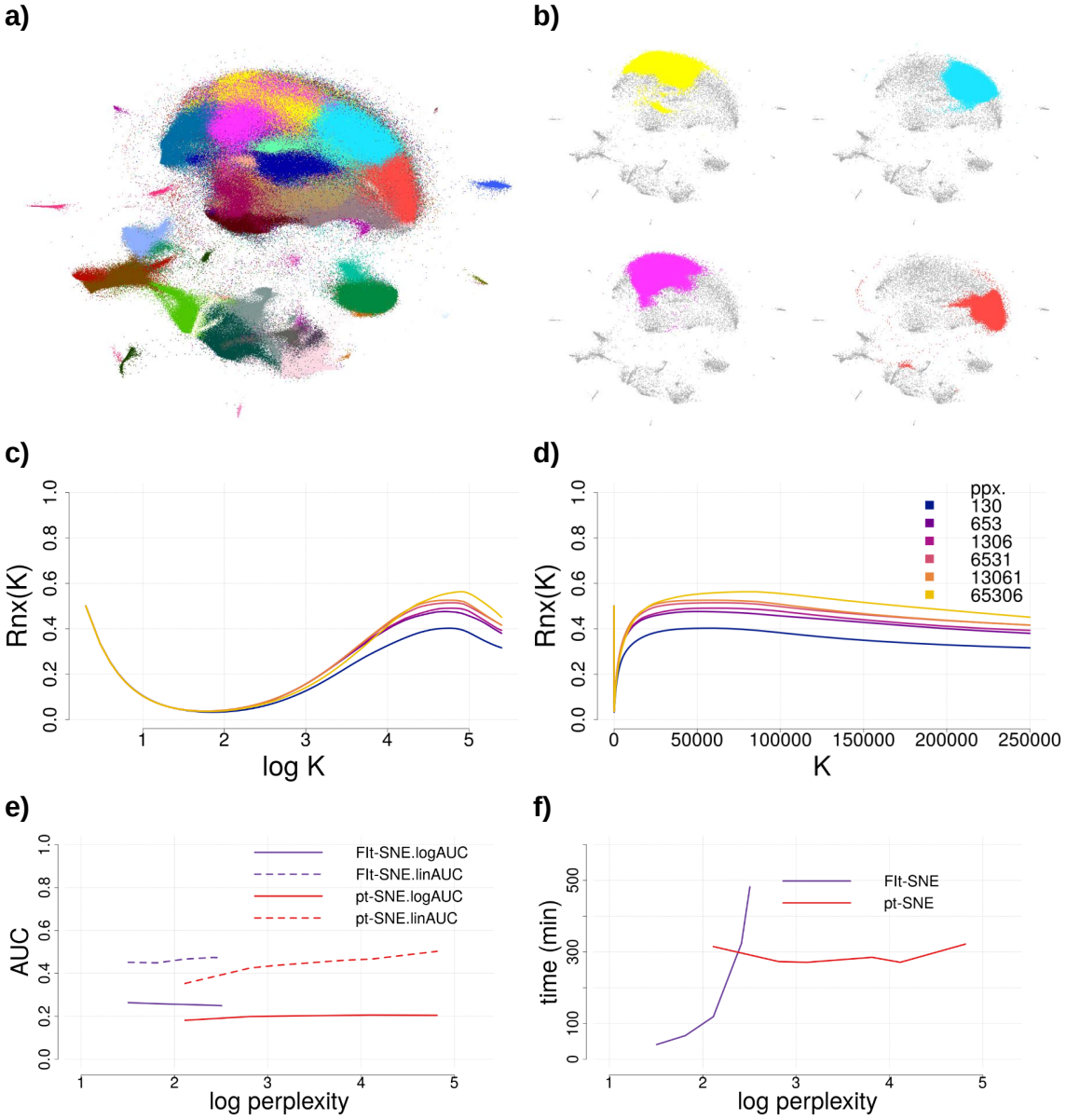

Figure 4: **1.3 Million Brain Cells from E18 Mice** [28]. Clustering labels taken from [14]. a) pt-SNE embedding with $ppx = 65306$ b) Overlay of the 4 largest classes; c), d) log and linear kNP.; e), f) pt-SNE vs. FIt-SNE, AUC and running times for different values of ppx.

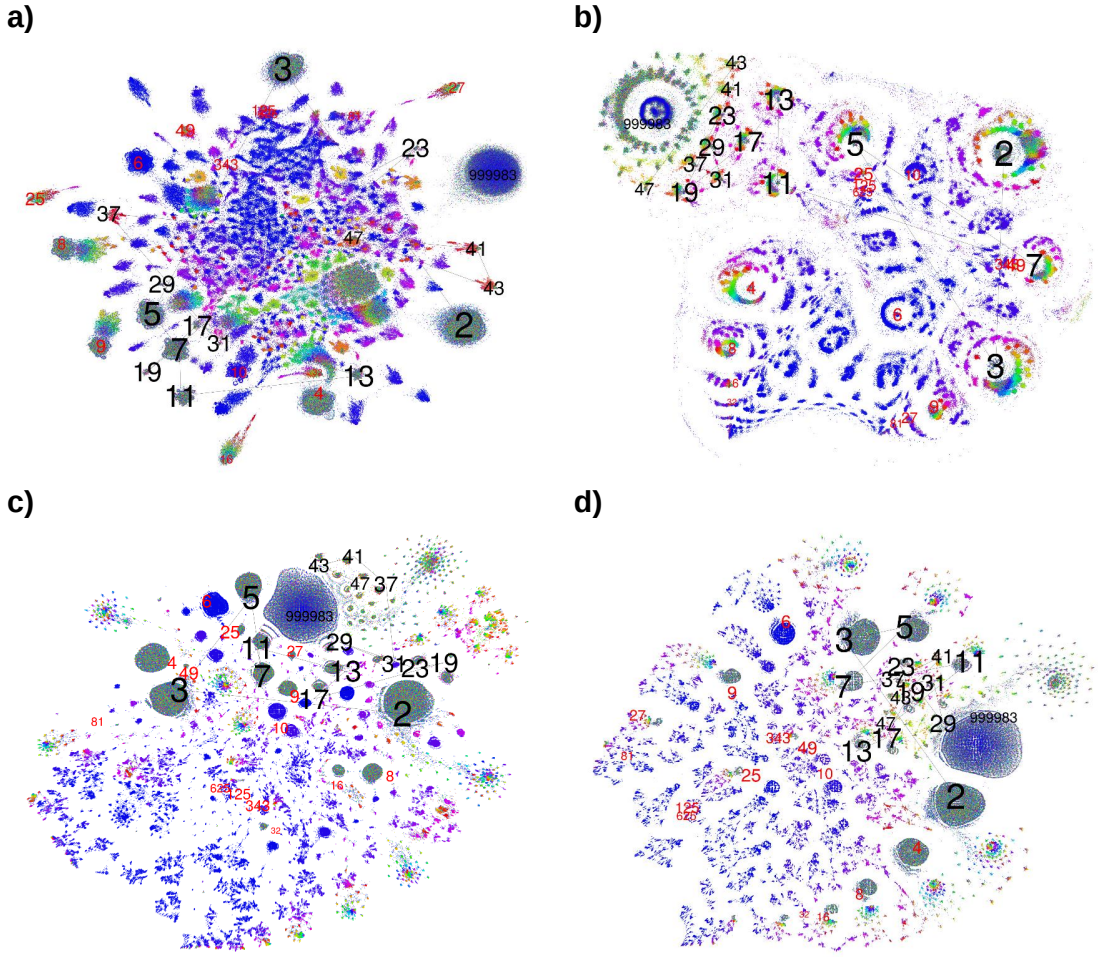

Figure 5: **Primes-1M data set. Embedding.**. a), b) pt-SNE visualization with $ppx = \{1000, 50000\}$; c), d) FIt-SNE visualization with $ppx = \{50, 400\}$. Hue color component is a combination of the two first prime factors while saturation depict the factors' powers. Embedding position of the first 15 primes (2 to 47) and of the last prime (999983) displayed in black. Embedding positions of some primes' powers of interest (e.g. 4, 8, 16, 32) displayed in red.

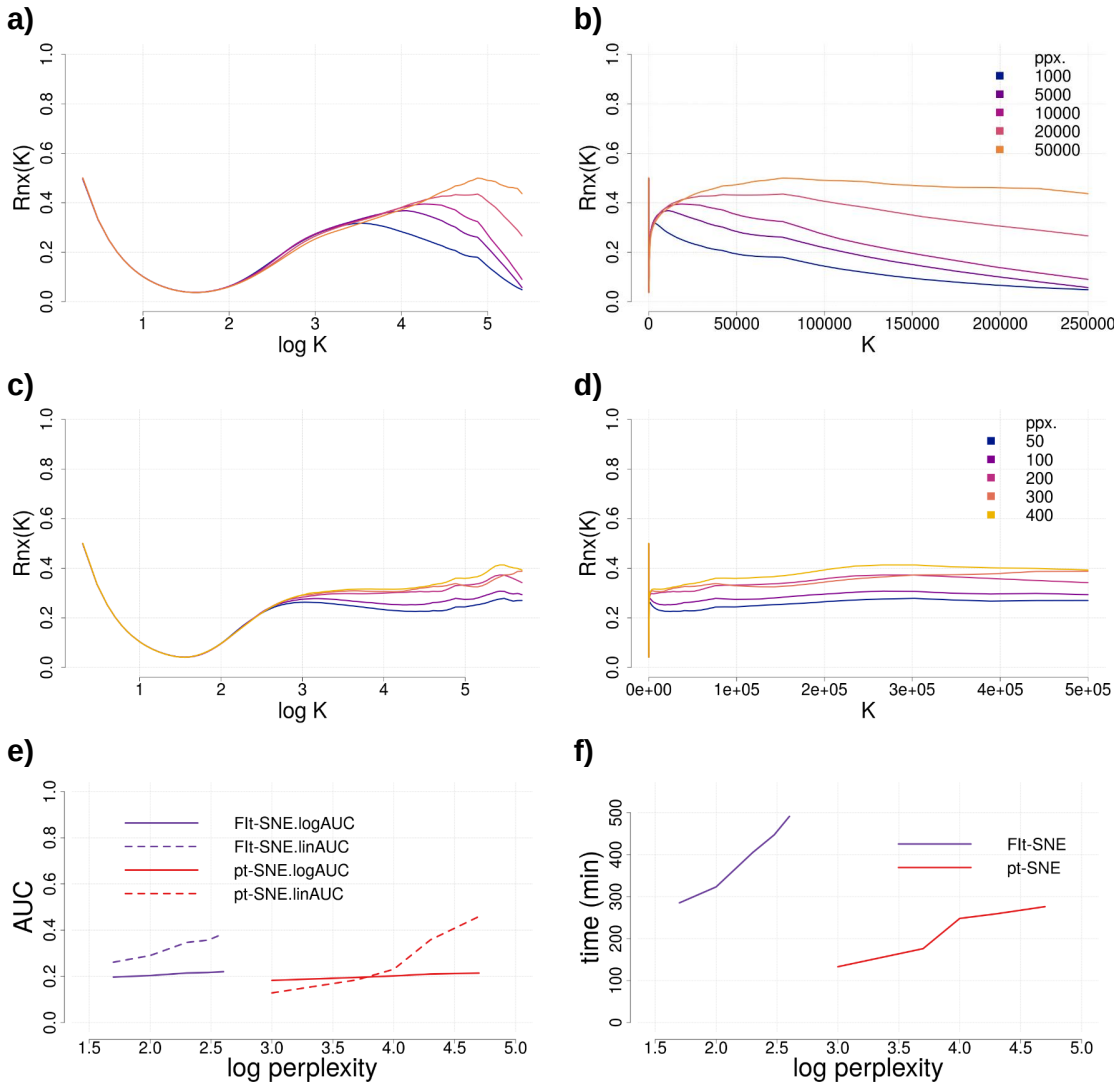

Figure 6: **Primes-1M data set. kNP and running times.**. a), b) pt-SNE log and linear kNP; c), d) FIt-SNE log and linear kNP; e) pt-SNE vs. FIt-SNE logAUC; f) pt-SNE vs. FIt-SNE running times.

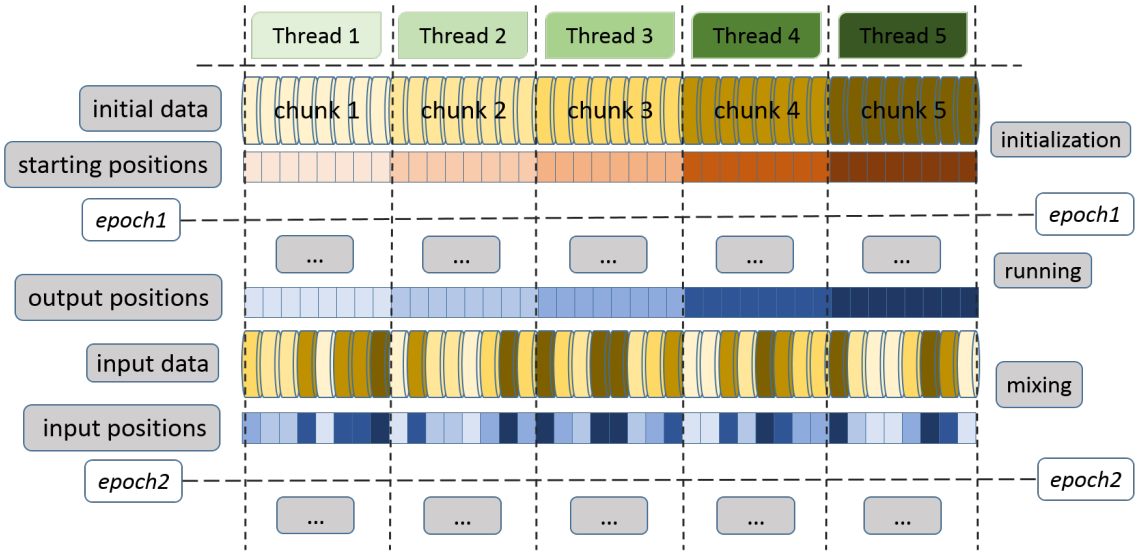

Figure 7: **ptSNE basic parallelization scheme**. The parallelized implementation runs a number (5 in this example) of instances of the t-SNE algorithm in an alternating scheme of short runs and mixing of partial solutions. Each run-and-mix phase is an epoch. In this example, each thread iterates on a single chunk of data, starting with random mapping positions. After a number of iterations the partial t-SNEs are pooled together and mixed. A new epoch is started with each thread iterating on a new chunk of data and its current mapping positions.

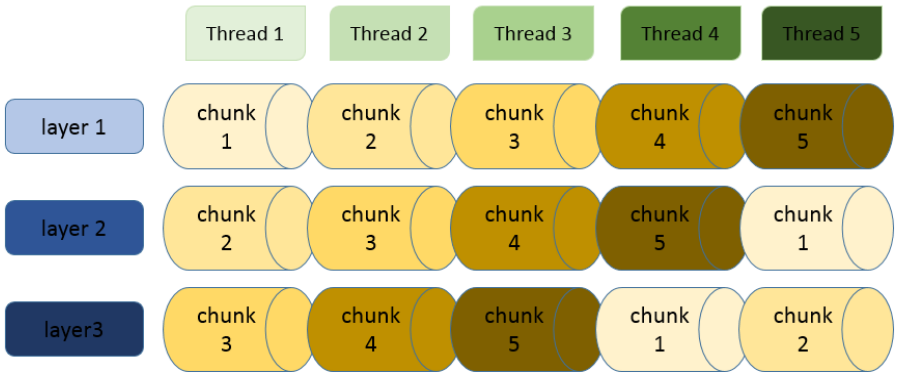

Figure 8: **ptSNE parallelization scheme with 3 layers**. Each thread iterates on 3 chunks of data sharing each one of them with successive threads. Common data points create a link between the partial solutions that favors convergence. As each data point is running on 3 different threads we get 3 different mapped positions for each one. After pooling all partial solutions we get 3 global mapping layers.

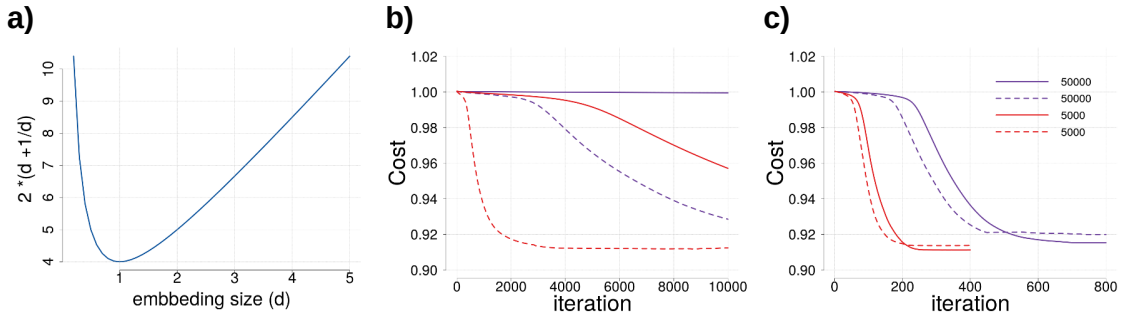

Figure 9: **Effect of the parameters in the gradient descent**. a) Plain learning rate as resulting from Eq.10; the learning-rate increases proportionally at both sides of $d = 1$. b) Plain gradient descent as resulting from Eq. 10 (solid lines) and after correcting for sample size by a factor $\log(\nu N_i)$ (dashed lines). c) Gradient descent including gain $g(i,t)$ (Eq. 9 (solid lines), and gradient descent with learning-rate (Eq. 9) and momentum (Eq. 11), (dashed lines). Color indicates sample size.

supporting information