Data Visualization with t-SNE in Theory and Practice

Solveig Tränkner

Geboren am 12. März 2002 in Wiesbaden, Hessen

21. März 2025

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Jochen Garcke

Zweitgutachter: Dr. Bastian Bohn

Institut für Numerische Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Deutsche Zusammenfassung

Der t-Distributed Stochastic Neighbor Embedding (t-SNE)-Algorithmus wird häufig zur Visualisierung hochdimensionaler Daten in verschiedenen Bereichen eingesetzt, darunter Einzelzellbiologie, natürliche Sprachverarbeitung und Computer Vision. Trotz seiner weiten Verbreitung ist das theoretische Verständnis des Algorithmus vergleichsweise begrenzt. In dieser Arbeit geben wir einen umfassenden Überblick über t-SNE und untersuchen sowohl dessen theoretische Grundlagen als auch praktische Aspekte. Wir analysieren eine Clustering-Garantie für t-SNE und untersuchen eine neu skalierte Variante, die für große Datensätze eine konsistente Einbettung gewährleisten soll. Darüber hinaus betrachten wir algorithmische Verbesserungen zur Beschleunigung des Verfahrens und zur automatischen Bestimmung optimaler Stoppzeitpunkte. In Experimenten analysieren wir anhand von realen Daten die Auswirkungen verschiedener Parametereinstellungen auf die Qualität der Visualisierungen.

Contents

1. Introduction

In our modern world, we are confronted with an ever-growing amount of data. Statista estimated the amount of data created, captured, copied and consumed worldwide in 2023 to be a staggering 123 zetabytes (123 billion terabytes), up from just two zetabytes in 2010 [Pet24]. As the amount of data grows, it becomes much more complicated to analyze. This is particularly evident in fields such as bioinformatics, where single-cell sequencing technologies have enabled the generation of vast datasets that capture gene expression at the resolution of individual cells. Analyzing large datasets becomes even more difficult due to the fact that they often very high-dimenional. In computer vision, for instance, images are represented by hundreds if not thousands of pixels and in natural language processing, words are transformed into high-dimensional vectors, e.g. via word2vec [Mik+13]. Traditional data visualization methods, such as scatter plots or boxplots, are unable effectively to visualize more than three dimensions.

To address this issue, various methods for embedding high-dimensional data into lower-dimensional spaces have been developed. One particularly influential approach is t-Distributed Stochastic Neighbor Embedding (t-SNE), introduced by van der Maaten and Hinton in 2008 [MH08], building upon earlier work on Stochastic Neighbor Embedding (SNE) [HR02]. Since its introduction, t-SNE has gained significant popularity across diverse domains, particularly in bioinformatics [Cie+20; Pla13; TR16], where it is widely used in single-cell transcriptomics [Mac+15; Tas+18; Cao+19], cancer research [DDP18; Gan+18; Sha+21], and beyond. Moreover, t-SNE has also been used in areas like finance [Gre+20], computer security [HS19; Xue+20; ZS21] and natural language processing [SS23; SR21; Wan+18].

This widespread adoption of the algorithm can be explained by its ability to generate visually interpretable representations of high-dimensional data. Unlike classical dimensionality reduction methods, t-SNE is specifically designed to retain local structures, which makes it effective at revealing clusters within the data. Since t-SNE addresses the so-called crowding problem, see Section 2.1, the embeddings it produces are often visually appealing.

At a high level, the t-SNE algorithm constructs probability distributions that measure pairwise similarities between data points in both the original high-dimensional space and the target low-dimensional space. It then iteratively adjusts the positions of the low-dimensional representations such that the difference between these two distributions, as measured by the Kullback-Leibler divergence, is minimized. This process can also be understood through the lens of dynamical systems, as described by [LS22]. One can think of every datapoint as a physical particle which experiences two types of forces: an attractive

force to its nearest neighbors in the high-dimensional space and a repulsive force towards all other particles [KB19].

Despite its success, t-SNE has notable limitations. One of the main challenges is its sensitivity to hyperparameter selection. While commonly used libraries [PSZ24; Bui+13] provide default settings, practitioners often need to experiment with multiple parameter configurations to achieve meaningful results. In particular, the perplexity parameter, which controls the number of effective neighbors considered in the high-dimensional space, has been studied extensively [WVJ16; SS23; Skr+24; CW17] and is often perceived as being the only parameter that needs to be tuned [KB19]. But "under the hood [...] there are also various optimisation parameters (such as the learning rate, the number of iterations, early exaggeration factor, etc.)", which have been shown to also have a great impact on the quality of the embedding [KB19]. The necessity of tuning multiple parameters makes it difficult to ensure reproducibility and interpretability and is impractical for users unfamiliar with the algorithm's inner workings.

Secondly, the theoretical foundations of t-SNE remain an active area of research. As is the case for many machine learning methods, empirical studies have demonstrated t-SNE's effectiveness, while rigorous mathematical analyses of its properties are still developing. Recent work has explored clustering guarantees and asymptotic behavior in the large-data limit, yet many theoretical insights rely on assumptions that may not always hold in practical applications. Bridging the gap between real-world usage and theoretical research remains challenging.

Objective of This Work

The goal of this thesis is to provide an overview of the current state of research on both the theoretical and practical aspects of the t-SNE algorithm. In particular, we aim to investigate how well the theoretical research on t-SNE carries over to practical applications with real-world data. We also want to explore the purely practical and algorithmic aspects, with a focus on examining how t-SNE can be accelerated via automatic stopping [Bel+19] and running experiments on a range of different hyperparameters.

Structure of the Thesis

We start by giving an overview of dimensionality reduction methods in Chapter 2 before discussing the t-SNE algorithm in detail in Chapter 3. In Chapter 4, we present the current state of theoretical research on t-SNE, focusing on clustering guarantees and behavior in the large-data limit. In Chapter 5, we then consider the practical aspects of t-SNE, including techniques to accelerate the algorithm and hyperparameter optimization. Finally,

in Chapter 6, we run experiments using different datasets, testing various hyperparameter settings and suggestions from theory.

Contributions

- After an introduction to t-SNE, we summarize the existing theoretical literature on t-SNE, including results on clustering guarantees and asymptotic behavior in the large-data limit.
- We investigate the practical aspects of t-SNE, offering an overview of current guidelines for selecting hyperparameters and discussing their impact on embedding quality.
- We implement the automatic stopping strategy outlined in [Bel+19] on top of the openTSNE library [PSZ24].
- We empirically test the claims in [MP24] regarding the limitations of standard t-SNE in the large-data regime and assess the proposed rescaled t-SNE variant.
- We conduct a hyperparameter study across datasets of varying sizes.

Acknowledgements

I want to thank Prof. Dr. Garcke for suggesting such a current and multifaceted topic and for his continued guidance and support throughout the development of this thesis. Thank you also to Dr. Bohn for agreeing to be the second advisor. Finally, I want to thank Jimena and Felipe for proofreading this thesis.

2. Dimensionality Reduction

This chapter provides context for t-SNE as a nonlinear dimensionality reduction method. Dimensionality reduction is an important tool in machine learning and data analysis, used to reduce computational complexity, for feature engineering, or to find substructures in the data [Gar21]. Beyond data visualization, it is widely used as a preprocessing step before applying downstream algorithms.

Dimensionality reduction aims to map high-dimensional data to a lower-dimensional space while preserving as much meaningful structure as possible. For instance, it is often assumed that the high-dimensional data lies on a lower-dimensional manifold, giving it a lower *intrinsic dimensionality*. The goal of dimensionality reduction is often to attempt to reduce the dimensionality of the data down to this intrinsic dimensionality, although this is not explicitly the case for data visualization, where we instead always aim for embeddings in up to three dimensions.

It is important to keep in mind that a lower-dimensional embedding can never fully preserve the structure of data with higher intrinsic dimensionality. For example, consider a tetrahedron in $\mathbb{R}^3$. It is impossible to flatten this shape onto $\mathbb{R}^2$ while exactly maintaining all pairwise distances.

2.1. The Curse of Dimensionality

High-dimensional spaces exhibit several counterintuitive phenomena, which are collectively known as the *curse of dimensionality*. These phenomena complicate data analysis in high-dimensional settings.

The first notable problem is that Euclidean distances become less meaningful as the number of dimensions increases due to the *concentration of measure* or *concentration of norms*. This becomes evident, for example, when we sample standard Gaussian distributions of increasing dimensionalities and compute the distribution of pairwise Euclidean distances between points drawn from each Gaussian, see [LCD22] for a visualization. We observe that the mean of these distributions grows at a rate of $\sqrt{d}$ whereas the variance of the distributions stays constant. As explained in [LCD22], the distribution of distances arising is a χ distribution with d degrees of freedom. This means that small distances very rarely occur in high dimensional datasets and, most importantly, that we lose relative contrast

between small and high distances as d increases. In fact, given a set of points in d dimensions with pairwise Minkowski distances D_{ij} where $D_{d,\max}$ denotes the minimal distance and $D_{d,\min}$ denotes the maximal distance, the following holds [PGW24]:

$$\lim_{d\to\infty} \frac{|D_{d,\max} - D_{d,\min}|}{D_{d,\min}} = 0.$$

Another phenomenon that can be observed is an exponential increase in volume of a space associated with an increasing number of dimensions. Consider, for instance, the tesselation of a d-dimensional hypercube [LCD22]. If we want to regularly sample each dimension with N points, the total number of points required grows as N^d. This implies that for increasingly high-dimensional datasets, the points occupy an increasingly small fraction of the space, leaving a lot of empty space.

This is also important in the context of data visualization. Say we are given points on a d-dimensional manifold that are approximately uniformly distributed in a hypersphere with radius r around a point x_i. We cannot accurately model the pairwise distances between the points and x_i in $\mathbb{R}^2$ since the hypersphere around x_i only has a volume of r^2 compared to r^d in the original data [Gar21]. This is called the *crowding problem*: in a lower-dimensional representation, there is insufficient space to accommodate all the nearest neighbors of a point while preserving their relative distances. As a result, points that are moderately distant in the high-dimensional space must be mapped much farther apart in the low-dimensional representation [MH08]. This discrepancy introduces small attractive forces between these moderately distant points, causing the entire embedding to contract toward the center of the map. Without an appropriate balancing mechanism, all points would cluster too closely together in the middle of the embedding.

2.2. Linear and Nonlinear Methods

Dimensionality reduction techniques can be broadly categorized into linear and nonlinear methods. Different dimensionality reduction methods aim to preserve different aspects of data structure, with some focusing on a preservation of global structure and others on preserving local structure [MH08].

Some desirable characteristics of dimensionality reduction methods include:

- Reproducibility: The method should involve minimal randomness to ensure consistent outcomes.
- Out-of-sample extension: It should facilitate the embedding of new points without necessitating a complete recomputation.
- Parameter robustness: The technique should exhibit low sensitivity to variations in parameter choices.

- Interpretability: The resulting low-dimensional representation should lend itself to meaningful interpretation.

- Computational efficiency: A fast runtime is essential, particularly for large datasets.

In this section, we provide a brief overview of representative methods before explaining t-SNE in detail in Chapter 3.

Linear Dimensionality Reduction

Linear methods assume that the primary variability in data occurs only along a few directions. The most widely used linear technique is Principal Component Analysis (PCA), which projects data onto the directions of maximum variance while minimizing the reconstruction error [Gar21].

Given a dataset represented as a matrix $X \in \mathbb{R}^{N \times d}$, PCA finds an eigenbasis of the covariance matrix and selects the eigenvectors corresponding to the k largest eigenvalues. These then define a new basis for a lower-dimensional representation of the data.

Algorithm 1: Principal Component Analysis (PCA)

Input: Data matrix $X \in \mathbb{R}^{N \times d}$, target dimension $p \leq d$
Output: Lower-dimensional representation $Z \in \mathbb{R}^{N \times p}$

1 Center the data:

$$X^c = X - \mu, \text{ with } \mu = \frac{1}{N} \sum_{i=1}^{N} X_i$$

where X_i denotes the ith row of X

2 Compute the covariance matrix:

$$C = \frac{1}{N} (X^c)^T X^c$$

3 Solve the eigenproblem $Cv_i = \lambda_i v_i$ via an eigendecomposition or SVD

4 Sort the eigenvalues λ_i in descending order and select the top p corresponding eigenvectors

$$V_p = [v_1, v_2, \ldots, v_p] \in \mathbb{R}^{d \times p}$$

5 **return** $Z = X^c V_p$

The advantages of PCA include a linear runtime, high interpretability and the fact that it is a parametric method, making embedding new points easy. However, PCA and other linear methods are limited in their ability to capture complex, nonlinear structures in data, see [Gar21].

Nonlinear Dimensionality Reduction

Many real-world datasets do not lie in a simple linear subspace but instead on a low-dimensional manifold within the high-dimensional space. In such cases, nonlinear methods, which try to discover the underlying manifold the data has been sampled from, are more effective.

Apart from t-SNE, one example of a nonlinear method is UMAP (Uniform Manifold Approximation and Projection), which builds on concepts from topological data analysis [McI+18]. It constructs a weighted graph representation of the high-dimensional data and then optimizes a low-dimensional embedding that preserves the fuzzy topological structure of this graph. It is often used as an alternative to t-SNE.

Spectral methods, including spectral clustering and diffusion maps [CL06], utilize the eigendecomposition of a similarity or Laplacian matrix derived from the data. By analyzing the spectrum of these matrices, one can extract lower-dimensional representations that capture both local and global properties of the data.

3. The t-SNE Algorithm

This chapter follows the descriptions of t-SNE in [MH08], [Maa14] and [CM22].

3.1. Computing Similarities

Let $\mathcal{X} = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$ with be a set of high-dimensional points that we wish to visualize and let $\mathcal{Y}^{(0)} = \{y_1^{(0)}, \dots, y_N^{(0)}\} \subset \mathbb{R}^2$ be the initial low-dimensional representation. For now, we sample each $y_i^{(0)}$ from $\mathcal{N}(0, 10^{-4}I)$, see more on initialization in Section 5.2. One may also consider embeddings into $\mathbb{R}$ or $\mathbb{R}^3$, but for the purpose of the thesis, we focus on two-dimensional embeddings.

High-dimensional Affinities

Instead of using the Euclidean distances between the data points, which, as pointed out in Section 2.1, are unreliable in high dimensions, t-SNE measures pairwise similarities via probabilities. Starting out with the high-dimensional affinities, we define a joint probability distribution over all pairs of data points $\{(x_i, x_j)\}_{1 \le i \ne j \le N}$ via

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \in \{1,2,\dots,N\} \setminus \{i\}} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}.$$

In order to obtain symmetric probabilities, we define

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

Since we are only interested in the pairwise similarities between points, we set $p_{ii} = 0$ for all i. In matrix form, we write $P = (p_{ij})_{1 \le i,j \le N}$. Note that while we use the Euclidean norm here, any norm and even precomputed distances will work.

Intuitively, one can think of $p_{j|i}$ as follows: if neighbors for the point x_i are chosen from $\mathcal{X} \setminus \{x_i\}$ according to a Gaussian centered at x_i with bandwidth σ_i, then $p_{j|i}$ is the probability that the point x_j is chosen. Thus, large p_{ij} values indicate that the points x_i and x_j are close to each other, while small values suggest they are far apart.

Perplexity

Let us return to the variance σ_i of the Gaussian centered at each datapoint x_i. What is a good value to choose? If we fixed a single value σ to be the same for every datapoint, this is likely not a good choice because real-life data often does not have a constant density everywhere but instead has sparser and denser regions. Given that we want to consider approximately the same number of nearest neighbors for each x_i, we opt to choose larger values of σ_i for sparse regions and smaller bandwidths for dense regions.

As described in [MH08], for a fixed datapoint x_i, choosing a particular value of σ_i induces a probability distribution P_i over all other datapoints. The entropy of this distribution increases as σ_i increases. The user can specify a so-called *perplexity*

$$\kappa = \mathrm{Perp}(P_i) = 2^{H(P_i)}$$

where $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$ denotes the Shannon entropy of P_i.

One can then determine the bandwidth to be used for a specific datapoint by performing a binary search for the value of σ_i that produces the user-specified perplexity κ.

Perplexity is essentially a smooth measure of the effective number of neighbors being considered in the calculation of the p_{ij}. This aligns with the common interpretation of the Shannon entropy as a measure of surprise. If we choose higher perplexity values, we essentially allow for more surprise with how we choose neighbors, i.e. not only the closest points will be considered but also points further away.

Low-dimensional Affinities

In order to measure similarities between points in the low-dimensional embedding, a first approach would be to also use a Gaussian distribution to convert pairwise distances into probabilities, as above. In fact, this is the approach taken originally in SNE [HR02].

However, to address the crowding problem outlined in Section 2.1, we instead use the more heavy-tailed Student t-distribution, with one degree of freedom, also called a Cauchy distribution, see Figure 3.1. As we aim to minimize the difference between p_{ij} and q_{ij}, using a heavier-tailed distribution for calculating Q means that points need to be pushed further away from each other for similarities to decrease. This way we can allow for moderate distances in the high-dimensional space being modeled by much larger distances in the low-dimensional embedding.

Using the Student t-distribution, we thus compute affinities for points in the low-dimensional embedding as follows:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k,l=1,k\neq l}^{N} (1 + \|y_k - y_l\|^2)^{-1}}$$

where we again define $q_{ii} = 0$ for all $1 \leq i \leq N$. We can collect all pairs of similarities in a symmetrical matrix $Q = (q_{ij})_{1 \leq i,j \leq N}$.

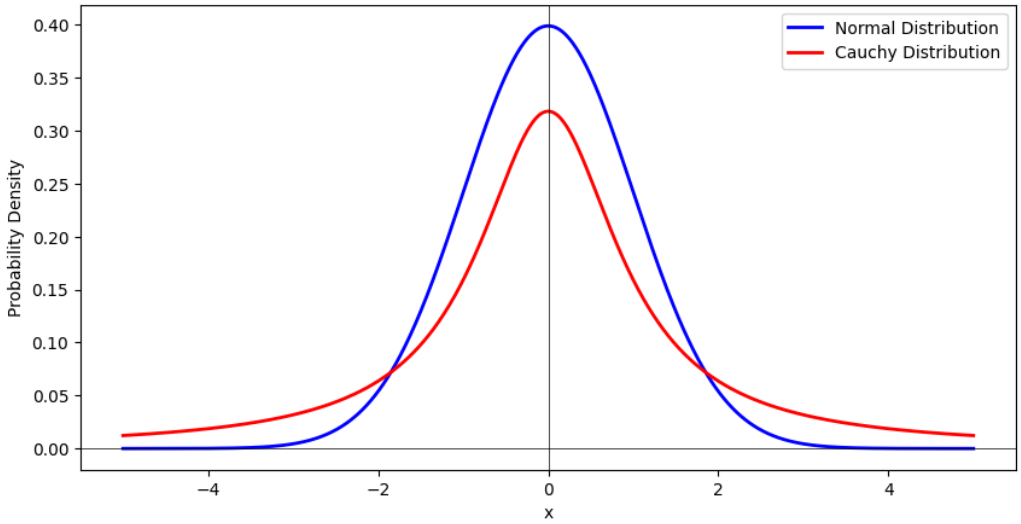

Figure 3.1.: Probability density functions of a standard Gaussian versus a Student t-distribution with one degree of freedom.

3.2. Optimizing the Embedding

Once we have computed the probabilities P and Q, the goal of the algorithm is to get these affinities to be as similar to each other as possible. A common choice for quantifying the similarity between two distributions is the Kullback-Leibler divergence.

Definition 3.1 (Kullback-Leibler Divergence)**.** The *Kullback-Leibler (KL) divergence* between two discrete probability distributions P and Q over the same probability space is defined as

$$D_{\mathrm{KL}}(P \parallel Q) \coloneqq \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}$$

where $\mathcal{X}$ is the domain of the distributions.

The t-SNE algorithm searches for the low-dimensional representation $\mathcal{Y} = (y_1, \ldots, y_N)$ that minimizes the KL divergence between the similarity matrices P and Q, with loss function given by

$$C(\mathcal{Y}) \coloneqq D_{\mathrm{KL}}(P \parallel Q) = \sum_{\substack{i,j=1 \\ i \neq j}}^{N} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

This leads to the following optimization problem:

$$\arg\min_{y_1,\dots,y_N} C(\mathcal{Y}) = \arg\min_{y_1,\dots,y_N} \sum_{\substack{i,j=1 \\ i\neq j}}^{N} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Note that the Kullback-Leibler divergence is, in fact, not a metric since it is not symmetric. One can observe that a large p_{ij} being modeled by a small q_{ij} leads to a bigger summand than using a large q_{ij} to model a small p_{ij}. This means that our loss function places a large cost on using far-apart points to model points that are close in the original dataset. On the other hand, there is only a small cost to model points that are actually far apart as nearby in the embedding. This shows that, as intended, we can expect a bigger focus on the preservation of local structure, which is important to keep in mind.

Minimizing the loss function can be achieved using gradient descent, with an updating equation of the form

$$y_i^{(t+1)} = y_i^{(t)} - \eta \frac{\partial C}{\partial y_i}^{(t)} + m^{(t+1)}(y_i^{(t)} - y_i^{(t-1)})$$

for $i = 1, \dots, N$, where t denotes the current iteration, $\eta > 0$ is a prespecified step size parameter, also called learning rate, and $m^{(t)} \geq 0$ is an optional momentum parameter used to speed up the convergence.

The gradient of our loss function with respect to y_i at iteration t is given by

$$\frac{\partial C}{\partial y_i}^{(t)} = 4 \sum_{\substack{1\leq j\leq n \\ j\neq i}} (p_{ij} - q_{ij}^{(t)}) q_{ij}^{(t)} Z^{(t)} (y_i^{(t)} - y_j^{(t)})$$

where $Z^{(t)}$ is a global normalization constant:

$$Z^{(t)} = \sum_{k\neq l} (1 + \|y_k^{(t)} - y_l^{(t)}\|^2)^{-1}.$$

Details of the derivation can be found in [MH08]. For reasons of computational efficiency, stochastic gradient descent with momentum and an adaptive learning rate determined by Jacob's scheme [Jac88] is used.

One may observe that the loss function is non-convex. This means that t-SNE embeddings may differ from run to run due to random initialization and the inherent randomness of stochastic gradient descent. This is a drawback for the interpretability and reproducibility of t-SNE embeddings. We also point out that t-SNE is a non-parametric method, making out-of-sample extensions thus very difficult.

Algorithm 2: Basic Version of t-SNE

Input: data set $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$, perplexity κ, number of iterations T, learning rate η, momentum $m(t)$

Output: Low-dimensional representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \dots, y_N\}$

Compute p_{ij} with perplexity κ

Sample initial solution $\mathcal{Y}^{(0)} = \{y_1^{(0)}, y_2^{(0)}, \dots, y_N^{(0)}\} \sim \mathcal{N}(0, 10^{-4}I)$

for $t = 1$ **to** T **do**

  Compute low-dimensional affinities q_{ij}

  Compute gradient $\frac{\delta C}{\delta \mathcal{Y}^{(t)}}$

  Update the low-dimensional embedding:

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} - \eta \frac{\delta C}{\delta \mathcal{Y}} + m(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$$

return $\mathcal{Y}^{(T)}$

3.3. Early Exaggeration

Early exaggeration, from now on also abbreviated as EE, was first proposed as a method of optimizing t-SNE in [MH08]. The idea is to multiply all p_{ij} by a value $\alpha > 0$ for the first few iterations of the algorithm. Since our loss function encourages the q_{ij} to model the p_{ij} as closely as possible, we achieve artificially large q_{ij} values this way. One thereby allows relatively tight clusters to form during the early stages of the optimization, which can then move around more easily in space. This makes it easier to find a good global organization of the clusters.

We can also understand early exaggeration from a dynamical systems viewpoint, cf. [LS22]. Notice that we can split the gradient (3.2) into two parts

$$\frac{\partial C}{\partial y_i}^{(t)} = 4\left(\sum_{j \neq i} p_{ij} q_{ij}^{(t)} Z^{(t)} (y_i^{(t)} - y_j^{(t)}) - \sum_{j \neq i} (q_{ij}^{(t)})^2 Z^{(t)} (y_i^{(t)} - y_j^{(t)})\right) = 4(F_{\text{attr}} + F_{\text{rep}})$$

where F_{attr} denotes the sum of all attractive forces and F_{rep} the sum of all repulsive forces.

Why does it make sense to call these attractive and repulsive forces? Since we want to minimize the cost function, we perform gradient descent and step in the direction of the negative gradient, so we consider the term

$$-\frac{1}{4}\frac{\partial C}{\partial y_i}^{(t)} = \sum_{j \neq i} p_{ij} q_{ij}^{(t)} Z^{(t)} (y_j^{(t)} - y_i^{(t)}) - \sum_{j \neq i} q_{ij}^2 Z (y_j^{(t)} - y_i^{(t)}).$$

The first term is considered the attractive term since it moves the point $y_i^{(t)}$ towards a weighted average of the other y_i. The weights $p_{ij} q_{ij}^{(t)} Z^{(t)}$ are bigger if the two points are

close to each other, both in the low- and high-dimensional space. The second term has the opposite sign and thus pushes $y_i^{(t)}$ away from a weighted average of the other points. This time, however, the weights only depend on the closeness of points in the low-dimensional space. Put together, this means that the attractive term attracts points that are meant to be next to each other based on their similarity in the high-dimensional space, and the repulsive term pushes points apart that get too close in the embedding space, regardless of their real similarity.

4. Theoretical Results

While t-SNE has been widely used in practice, the theoretical understanding of the algorithm remains relatively underdeveloped. However, theoretical results might improve our understanding of how and why t-SNE works, and may guide new directions for the algorithm. This chapter will first provide an overview of recent theoretical work on t-SNE and then examine a few results in detail.

4.1. Literature Review

An overview of the theoretical literature on t-SNE can be found in [MP24] and [LS22].

Early theoretical work has been focused on establishing guarantees that t-SNE clusters well-clusterable data. Shaham and Steinerberger started this line of research with their paper "Stochastic Neighbor Embedding Separates Well-Separated Clusters" [SS17]. They prove a clustering guarantee for the precursor algorithm SNE, which has since been criticized as it is nontrivial only when the number of clusters is significantly larger than the number of points per cluster. This is an unrealistic assumption for most datasets t-SNE is commonly used on, see [AHK18].

Linderman and Steinerberger use a dynamical systems approach to prove that t-SNE suceeds in clustering clustered data in "Dimensionality Reduction via Dynamical Systems: The Case of t-SNE" [LS22]. We will dedicate Section 4.2 to discuss their result in detail. Building on [LS22], [AHK18] give a first formal framework for the problem of data visualization and formulate an improved clustering guarantee for some other definition of spherical and well-separated clusterable data.

In [CM22], a range of theoretical results on t-SNE are established, notably an asymptotic equivalence of the EE phase with power iterations in spectral embeddings. This means that for strongly clustered data, one can replace the EE phase with a spectral embedding, thereby speeding up the process. They also prove that embeddings remain localized within the initial range during EE. Furthermore, they separate the embedding phase (the phase with no exaggeration) into an amplification phase, where the embedding expands rapidly and a stabilization phase, during which the speed of the expansion slows notably.

Practical implications of their work include:

- Stopping EE early for noisy data to avoid overfitting. They suggest using only $\lfloor (\log N)^2 \rfloor$ iterations of EE.

- The observation that t-SNE is reliable in terms of cluster membership but not relative positions of clusters.

- The observation that false clustering may occur, see also [WVJ16]. One should thus run t-SNE multiple times if possible.

More recently, the focus of theoretical research on t-SNE has shifted to include the question of equilibrium distributions and convergence in the large data limit $N \to \infty$. A first work exploring this question is [AF23]. More specifically, the authors investigate the behavior of t-SNE when the input dataset consists of independent, identically distributed random variables and prove that, under certain conditions, the empirical measure of the output converges to an equilibrium distribution as the number of data points grows. We will focus on another work focused on large data limits, namely "Large data limits and scaling laws for tSNE" by Murray and Pickarski [MP24], which will be explored in detail in Section 4.3.

4.2. Clustering Guarantees

In this section, we analyze the theoretical clustering guarantees presented in [LS22]. Using a discrete dynamical systems viewpoint on t-SNE, the authors establish conditions under which the algorithm provably contracts clusters in the low-dimensional embedding. We summarize their assumptions, state the main theorem and evaluate its limitations.

Assumptions

The theorem is built on three assumptions regarding the dataset, the choice of parameters in the t-SNE optimization process, and the initialization of the embedding:

(i) **Clustered data**. We assume that there exists a number of clusters $k \in \mathbb{N}$ and a map $\pi : \{1, \dots, N\} \to \{1, \dots, k\}$ which maps each point to a cluster, such that the following holds: if $\pi(x_i) = \pi(x_j)$, then

$$p_{ij} \geq \frac{1}{10N|\Omega(i)|} \tag{4.1}$$

where $\Omega(i)$ is the cluster in which x_i and x_j lie. This ensures that points within the same clusters have high similarities, but it does not require inter-cluster affinities to be small.

(ii) **Parameter choice**. Learning rate η and exaggeration parameter α are chosen such that for some $1 \leq i \leq N$,

$$\frac{1}{100} \leq \alpha\eta \sum_{\substack{j \neq i \\ \pi(i)=\pi(j)}} p_{ij} \leq \frac{9}{10}.$$

(iii) **Localized initialization**. The initial embedding satisfies $\mathcal{Y} \subset [-0.01, 0.01]^2$.

Main Theorem

Theorem 4.1 (Shrinkage of Clusters [LS22])**.** *Under assumptions (i)-(iii), the diameter the embedded cluster containing y_i, $\{y_j : \pi(i) = \pi(j)\}$, decays exponentially until its diameter satisfies, for some constant $c > 0$,*

$$\operatorname{diam}\{y_j : \pi(i) = \pi(j)\} \leq c \cdot \eta \left(\alpha \sum_{j \neq i, \pi(i) \neq \pi(j)} p_{ij} + \frac{1}{N} \right).$$

To prove this theorem, the authors use the dynamical systems viewpoint on t-SNE and employ stability of the convex hull and a contraction inequality to show the exponential shrinkage of the cluster.

This result is applicable to a single cluster, but it can be applied to every cluster of a given dataset separately, given that (i) holds. It is important to note that this theorem does not rule out the possibility of multiple clusters merging into one in the embedding since it only makes a statement about the behavior of a single cluster. This was pointed out by [AHK18], who also suggest a different version of this theorem which solves the issue of cluster separation.

While Definition 4.1 establishes an important property of t-SNE, which would justify its usage as a means to cluster data, its assumptions can be criticized. Assumption (iii) can be easily realized in practice, but the parameter choices of $\alpha \sim N/10$ and $\eta \sim 1$ suggested by the authors in line with assumption (ii) do not align with settings that have proven to generate good empirical results. Furthermore, as pointed out in the paper itself, assumption (ii) only holds for datasets with at most $20,000$ points, a smaller size than most datasets on which t-SNE is commonly used.

Concerning the assumption of clustered data, the authors of [LS22] claim that their "assumptions on what it means to be clustered are so weak that a given data set does not necessarily have a unique decomposition into different clusters". However, this claim was criticized in [YCC21], who found that this supposedly weak condition still does not hold in many cases since even within a cluster, very small p_{ij} values can occur. We will investigate this claim on a real-world dataset in Chapter 6.

4.3. Large Data Limits

In the past, theoretical literature on t-SNE has not focused much on what happens when we let the number of data points go to infinity. After all, real-world datasets consist of a finite number of points. It is, however, an interesting question that allows us to get deeper insight into the inner workings of t-SNE. In this section, we will look into the analysis done on this topic in [MP24]. They first show that standard t-SNE embeddings do not have a consistent limit as $N \to \infty$ and then propose a rescaled model with a consistent limit which mitigates the asymptotic decay of the attractive forces.

Setup

For this section, instead of assuming a given dataset, we assume a probability distribution $\mu \in \mathcal{P}(\Omega)$ on $\mathbb{R}^d$ which is supported on a bounded and C^2 domain Ω. We further assume that μ has a bounded density function $\rho(x)$ with respect to the Lebesgue measure. Our points $X_1, \dots, X_N$ are then drawn from μ independently, and the p_{ij} and q_{ij} values are calculated as before. In matrix form, we write $P_N = (p_{ij})_{i,j=1}^N$ and $Q_N = (q_{ij})_{i,j=1}^N$. As our loss function, we do not consider the KL divergence between P_N and Q_N directly but instead define a functional similar to the KL divergence:

$$\mathrm{KL}_N(T) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}(T)}$$

where

$$q_{ij}(T) \coloneqq \frac{(1 + |T(X_i) - T(X_j)|^2)^{-1}}{\sum_{k \neq l}(1 + |T(X_k) - T(X_l)|^2)^{-1}}$$

and $T \colon \mathbb{R}^d \to \mathbb{R}^2$. Note the slight abuse of notation since this is no longer a KL divergence when considered a function of T. However, it is only a small reformulation of the original problem, where we minimized the KL divergence over all possible $\mathcal{Y}$. Here, we instead minimize the loss over all maps T.

Building on the view of t-SNE in terms of attraction-repulsion dynamics, [MP24] split up the t-SNE objective function into an attractive term $A_N[T]$, a repulsive term $R_N[T]$ and a purely data-dependent term D_N, which plays no role during gradient descent.

These terms are defined via

$$A_N[T] \coloneqq \frac{1}{N} \sum_{i=1}^N \frac{\sum_{j=1}^N \exp(-|X_i - X_j|^2/2\sigma_i^2) \log(1 + |T(X_i) - T(X_j)|^2)}{\sum_{j=1}^N \exp(-|X_i - X_j|^2/2\sigma_i^2)}$$

and

$$R_N[T] \coloneqq \log\left(\frac{1}{N^2} \sum_{i,j=1}^N \frac{1}{1 + |T(X_i) - T(X_j)|^2}\right).$$

We can then write $\mathrm{KL}_N(T) = A_N[T] + R_N[T] + D_N$. This definition of attractive and repulsive terms makes sense since the attractive term is minimized when $T \to 0$ and the repulsive term is minimized when $T \to \infty$. Thus, we obtain the following optimization problem:

$$\operatorname*{arg\,min}_{T\colon \mathbb{R}^d \to \mathbb{R}^2} \mathrm{KL}_N(T) = \operatorname*{arg\,min}_{T\colon \mathbb{R}^d \to \mathbb{R}^2} A_N[T] + R_N[T].$$

Regarding perplexity, our first thought would be to keep it constant as $N \to \infty$. However, theoretical analysis based on results in adaptive kernel density estimation shows that maintaining a fixed perplexity as the number of data points increases forces the bandwidths $\sigma_{i,N}$ to scale as $\mathcal{O}(N^{-1/d})$ [MP24]. Results from adaptive kernel estimation indicate that this is bad for the consistency of the estimator, so one instead allows perplexity to grow slowly with the number of samples, at a rate of N^β with $\beta > 0$.

Repulsive Forces Dominate in the Large Data Limit

From now on, we write the bandwidths as $\sigma_{i,N} = h_N \sigma_i$, isolating the stochastic part in σ_i with h_N being deterministic and write perplexity as

$$\mathrm{Perp}_N(X_i | h_N \sigma_i) = \exp\left(- \sum_{k \neq i}^{N} p_{k|i}(\sigma_{i,N}) \log(p_{k|i}(\sigma_{i,N})) \right).$$

If we let perplexity grow at a rate faster than $\log(N)$ and slower than N, we get the following limiting bandwidth.

Theorem 4.2 ([MP24])**.** *Let h_N be a sequence for which $\frac{N h_N^d}{\log(N)} \to \infty$ and $h_N \to 0$. Let $\rho(x)$ be a uniformly continuous density that is bounded above and below. If $\hat{\sigma}_N(x)$ is chosen so that $\mathrm{Perp}_N(x | h_N \hat{\sigma}_N(x)) = \kappa N h_N^d$, then for $\tilde{\Omega} \subset\subset \Omega$ we have almost surely*

$$\lim_{n \to \infty} \left\| \hat{\sigma}_N(x) - \frac{1}{\sqrt{2\pi e}} \left(\frac{\kappa}{\rho(x)} \right)^{1/d} \right\|_{L^\infty(\tilde{\Omega})} = 0.$$

We denote the limiting object from the theorem above as $\sigma_\kappa(x)$ from now on. Note that we see that bandwidths shrink to zero in this regime since $h_N \to 0$ and $\hat{\sigma}_N(x)$ converges to a fixed number.

In order to simplify the proofs, [MP24] now replace the stochastic $\sigma_{i,N}$ with the deterministic $h_N \sigma_\kappa(x)$ and define an averaged version of the energies

$$\tilde{A}_N[T] \coloneqq \int_\Omega \frac{\int_\Omega \exp\left(\frac{-|x-x'|^2}{2h^2 \sigma_\kappa^2(x)} \right) \log(1 + |T(x) - T(x')|^2) \rho(x') \, \mathrm{d}x'}{\int_\Omega \exp\left(\frac{-|x-x'|^2}{2h^2 \sigma_\kappa^2(x)} \right) \rho(x') \, \mathrm{d}x'} \rho(x) \, \mathrm{d}x$$

and

$$\tilde{R}_N[T] \coloneqq \log\left(\iint_{\Omega\times\Omega} \frac{1}{1+|T(x)-T(x')|^2}\rho(x)\rho(x')\,\mathrm{d}x\,\mathrm{d}x'\right).$$

It can be shown that $\mathbb{E}[A_N[T]] \to \tilde{A}_h[T]$ and $\mathbb{E}[R_N[T]] \to \tilde{R}[T]$ as $N \to \infty$. With this, we can state that the averaged t-SNE loss function does not have a limiting solution:

Theorem 4.3 ([MP24])**.** *Let T_N be a sequence of minimizers of the energies $\tilde{A}_{h_N}[T]+\tilde{R}[T]$. Then T_N does not converge pointwise to any $T^* \in L^\infty(\Omega,\mathbb{R}^2)$.*

For details of the proof, we refer to [MP24]. However, the main idea is again to consider how attraction-repulsion dynamics change as the number of samples grows. Since the bandwidths of our Gaussian kernels go to zero, the attractive term will also shrink to zero as $N \to \infty$. This means that for increasing sample sizes, the repulsive force dominates, resulting in embeddings that expand without bounds.

Note that this also gives us further intuition into why we perform early exaggeration. It is not only an optimization trick, but also a means to strengthen attractive forces, which is asymtotically consistent with the behavior of the algorithm.

A Rescaled, Consistent t-SNE

Taking the above theorem as motivation, [MP24] propose a rescaled version of t-SNE, where we not only apply exaggeration in the early iterations of the algorithm but instead multiply the entire attractive force by a sequence $1/h_N^2$, with $h_N \to 0$ and $\frac{Nh_N^d}{\log(N)} \to \infty$ as $N \to \infty$. The rescaled model for N samples thus minimizes the loss function

$$\widehat{\mathrm{KL}}_N(T) = \frac{A_N[T]}{h_N^2} + R_N[T].$$

This energy will then, for a fixed T, converge towards the limiting energy

$$\mathcal{KL}(T) \coloneqq \frac{\kappa^{2/d}}{2\pi e}\int_\Omega \sum_{i=1}^m |\nabla T_i(x)|^2 \rho^{1-2/d}(x)\,\mathrm{d}x + \log\left(\iint_{\Omega\times\Omega} \frac{\rho(x)\rho(x')}{1+|T(x)-T(x')|^2}\,\mathrm{d}x\,\mathrm{d}x'\right),$$

which behaves consistently, as shown by the following theorem.

Theorem 4.4 ([MP24])**.** *Let μ be a distribution supported on a compact, C^1 domain $\Omega \subset \mathbb{R}^d$ with Lebesgue density $\rho(x)$ bounded above and below on Ω. Then, for every $T \in C^2(\Omega;\mathbb{R}^2)$ we have $\lim_{N\to\infty} \widehat{KL}_N(T) \to \mathcal{KL}(T)$. If we assume Ω to be connected, then there exists $T^* \in H^1(\Omega,\rho)$ for which $\mathcal{KL}(T^*) = \inf_{T\colon \Omega\to\mathbb{R}^2} \mathcal{KL}(T)$.*

The hope is that this new approach may provide more stable and consistent embeddings in practice as well, which we will investigate in Section 6.10.

5. Practical and Algorithmic Aspects

5.1. Accelerating t-SNE

The original t-SNE algorithm is computationally expensive, requiring $\mathcal{O}(N^2)$ operations to compute the pairwise similarities p_{ij} and q_{ij}. The primary computational bottleneck arises from the normalization we use when computing these probabilities since it involves sums over $N(N-1)$ pairs of points. To address this, various acceleration techniques have been developed. This section provides an overview of two widely used approaches to accelerating t-SNE.

Barnes-Hut t-SNE

The first significant advancement in accelerating t-SNE was the Barnes-Hut approximation, proposed by van der Maaten in 2014 [Maa14]. This method leverages hierarchical space-partitioning structures to approximate the repulsive forces efficiently, reducing the computational complexity to $\mathcal{O}(N \log N)$.

Computing the attractive forces, as defined in Section 3.3, is computationally feasible if input similarities are approximated using a sparse representation. Recall that the input similarities p_{ij} are computed based on a Gaussian kernel, where dissimilar points yield very small similarity values. Instead of computing all $N(N-1)$ pairwise similarities, Barnes-Hut t-SNE focuses on a subset by considering only the $\lfloor 3\kappa \rfloor$ nearest neighbors of each data point. The set of nearest neighbors for a point x_i is denoted as $\mathcal{N}_i$.

The similarities are thus given by

$$p_{j|i} = \begin{cases} \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \in \mathcal{N}_i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases}$$

which are subsequently again symmetrized.

To efficiently determine the nearest neighbor sets $\mathcal{N}_i$, a vantage-point tree is constructed, enabling nearest neighbor search in $\mathcal{O}(\kappa N \log N)$ time. For further implementation details, see [Maa14].

For the repulsive forces, the approach outlined above is not feasible since the low-dimensional similarities q_{ij} change during the optimization. Instead, we use the Barnes-Hut algorithm to approximate the repulsive forces efficiently. The key observation underlying this method is that the influence of a distant cluster of points on a given point can be approximated by treating the cluster as a single entity. More precisely, if we consider points y_i, y_j and y_k with $\|y_i - y_j\| \approx \|y_i - y_k\| \gg \|y_j - y_k\|$, then the contributions of y_j and y_k to $F_{\text{rep},i}$ will be roughly equal.

To take advantage of this fact, the Barnes-Hut algorithm constructs a *quadtree* (for two-dimensional embeddings) or an *octree* (for three-dimensional embeddings) to hierarchically partition the embedding space. The algorithm then traverses the tree via a depth-first search, deciding at each node whether the entire cell can be used as a summary of the contributions to F_{rep} of all points in the cell.

A cell of the quadtree, centered at y_{cell}, can be used as an approximation if the following condition holds:

$$\frac{r_{\text{cell}}}{\|y_i - y_{\text{cell}}\|} < \theta,$$

where r_{cell} is the cell's diagonal length and θ is a user-defined threshold that balances speed and accuracy. If this condition is met, the contribution of all points within the cell is approximated using the center of mass. Otherwise, the algorithm recursively explores smaller subregions of the tree. This approach allows for efficient computation of the repulsive forces in $\mathcal{O}(N \log N)$ time while maintaining high accuracy in the embedding. For a visualization of the quadtree structure, we refer to [Maa14].

Fast Interpolation-Based t-SNE

As observed in [KB19], the Barnes-Hut implementation of t-SNE, while significantly faster than the originally proposed algorithm, still becomes prohibitively slow for datasets with $n \gg 100,000$. The FFT-accelerated, interpolation-based version of t-SNE proposed by Linderman et al. [Lin+19] further speeds up the computation of t-SNE embeddings.

Again, we focus on accelerating the computation of the repulsive forces since this is the most expensive step. The key idea is to use interpolation. We define p interpolation nodes that "mediate the interaction" between all points [Lin+19]. Using these, we can calculate the interaction of every point with the nodes in pN computations. Then, we compute the interaction of all the mediation nodes with each other. Naively, this would take p^2 computations, but using the Fast Fourier Transform (FFT), we can speed this step up to $p \log(p)$. At the end, we can interpolate for the interpolation nodes to all original points. This again requires pN computations. Overall, we end up with a runtime of $\mathcal{O}(pN)$, with p being much smaller than N.

We will now discuss how the algorithm works in detail, cf. Algorithm 3. Firstly, note that

at every step of gradient descent, the repulsive forces are

$$F_{\text{rep},k}(m) = \frac{\sum_{l\neq k} \frac{y_l(m)-y_k(m)}{(1+\|y_l-y_k\|^2)^2}}{\sum_{j=1}^{N}\sum_{l\neq j}(1+\|y_l-y_j\|^2)^{-1}}.$$

where $k = 1, \ldots, N$, $m = 1, 2$ and $y_i(m)$ denotes the mth component of y_i. Observe that these can be expressed via sums of the form

$$\phi(y_i) = \sum_{j=1}^{N} K(y_i, y_j) q_j$$

with either

$$K_1(y, z) = \frac{1}{1+\|y-z\|^2} \text{ or } K_2(y, z) = \frac{1}{(1+\|y-z\|^2)^2},$$

both of which are smooth and translation-invariant kernels, making the kernel matrix $\tilde{K}_{ij} = K(\tilde{y}_i, \tilde{y}_j)$, $i, j = 1, \ldots, N_{\text{int}}p$ Toeplitz.

Algorithm 3: FFT-accelerated Interpolation-based t-SNE (FIt-SNE, [Lin+19])

Input: embedding points $\{y_i\}_{i=1}^N$, weights $\{q_i\}_{i=1}^N$, number of intervals N_{int}, interpolation points per interval p

Output: $\varphi(y_i) = \sum_{i=1}^N K(y_i, y_j) q_j$ for $i = 1, \ldots, N$

For every interval I_l, form p equispaced nodes $\tilde{y}_{j,l} = 1/(2N_{\text{int}}p) + \frac{j-1+(l-1)p}{N_{\text{int}}p}$ for $j = 1, \ldots, p$.

for $l = 1$ **to** N_{int} **do**

  Compute the coefficients $w_{m,l}$ given by

$$w_{m,l} = \sum_{y_i \in I_l} L_{m,\tilde{y}^l}(y_i) q_i, \ m = 1, \ldots, p$$

  with Lagrange polynomial $L_{m,\tilde{y}^l}$.

Use FFT to compute values of $v_{m,n}$ given by

$$[v_{1,1}, v_{2,1}, \ldots, v_{p,N_{\text{int}}}]^T = \tilde{K}[w_{1,1}, w_{2,1}, \ldots, w_{p,N_{\text{int}}}]^T$$

where $\tilde{K}$ is the Toeplitz matrix given by $\tilde{K}_{ij} = K(\tilde{y}_i, \tilde{y}_j)$, $i, j = 1, \ldots, N_{\text{int}}p$.

for $l = 1$ **to** N_{int} **do**

  Compute $\varphi(y_i)$ at all points $y_i \in I_l$ via

$$\varphi(y_i) = \sum_{j=1}^{p} L_{j,\tilde{y}^l}(y_i) v_{j,l}$$

For the interpolation, we subdivide the embedding space into a number of intervals (in the one-dimensional case) or squares (in the two-dimensional case). In each interval, we choose a fixed number p of equispaced interpolation nodes. We then approximate each kernel by its low-order polynomial interpolant. This replaces the direct kernel evaluation between

every pair of points with an evaluation on a much coarser grid. Since the kernel matrix is Toeplitz, we can use FFT to speed up the calculation of the pairwise interactions between the interpolation nodes.

As pointed out in [PSZ24], FIt-SNE scales linearly with the number of samples, but it introduces additional computational overhead for each embedding. This is often unnecessary for smaller datasets, which is why openTSNE uses Barnes-Hut t-SNE for datasets with fewer than $10,000$ samples and FIt-SNE for data sets with at least $10,000$ datapoints.

5.2. Choosing Hyperparameters

Here, we collect guidelines for choosing t-SNE parameters and look at which parameters are standard in scikit-learn and openTSNE. After this overview, we will go into more detail with regard to an early stopping strategy.

Perplexity. Van der Maaten initially suggested $\kappa = 40$ [MH08] and later $\kappa = 50$ [Maa14]. Belkina et al. find that the exact value of perplexity does not greatly impact the embedding when choosing values between 30 and 50 [Bel+19]. By default, the scikit-learn and openTSNE libraries use $\kappa = 30$.

If one wants to increase the global consistency of t-SNE, one way to do it would be by increasing perplexity since this increases the sizes of the neighborhoods considered for constructing the probabilities p_{ij} [PSZ24]. However, this often leads to a loss in local structure. For instance, small, isolated clusters of points may get sucked into larger ones when using a very large perplexity. To remedy this effect, [KB19] propose to use a mixture of Gaussians with different bandwidths when constructing the p_{ij}. Instead of using the standard Gaussian kernel $\exp(-d^2/2\sigma_i^2)$, where d denotes distances, they use the multi-scale kernel

$$\frac{1}{\sigma_i}\exp\left(-\frac{d^2}{2\sigma_i^2}\right) + \frac{1}{\tau_i}\exp\left(-\frac{d^2}{2\tau_i}\right)$$

where the bandwidth of the first kernel σ_i is chosen such that the perplexity of this component is 30, and τ_i such that the perplexity of the second component is $N/100$. This achieves a combination of different perplexities. However, it is worth bearing in mind that using a perplexity of $N/100$ can get very expensive computationally and is only feasible for smaller datasets.

Initialization. The standard t-SNE algorithm starts with an initization of $y_i^{(0)}$ which are drawn independently from $\mathcal{N}(0, \delta^2 I)$ for some small $\delta > 0$, see [MH08] and [Maa14]. However, [KL21] show that informative initialization leads to embeddings that better preserve large-scale structures within the data. At times, this attempt to preserve global geometry of data can fail if the macroscopic structure is not adequately captured in the first two principal components. One could, for example, imagine a small isolated cluster that might

not appear isolated in the first two principal components because it simply does not have enough cells to contribute much to the explained variance.

In [KB19], it is argued that PCA initialization makes t-SNE more reproducible since it removes at least part of the randomness by introducing a deterministic initialization for the low-dimensional embedding.

Indeed, modern implementations of t-SNE in libraries like openTSNE [PSZ24] or Scikit-learn now all use PCA initialization by default. This means that we perform a principal component analysis on the input data $x_1, \ldots, x_N$ and use the output $y_1, \ldots, y_N$ as the initial points for the low-dimensional embedding.

Learning Rate. Van der Maaten [MH08] suggests $\eta = 100$ and using an adaptive learning rate scheme, the Jacob's scheme [Jac88]. Later on, an initial learning rate of $\eta = 200$ and an additional momentum term of weight 0.5 during EE and with weight 0.8 after EE are also suggested [Maa14]. In the theoretical paper [LS22], a learning rate of $\eta = 1$ is used, which is very small in comparison with all other guidelines. Empirical papers suggest rather larger learning rates, for example $\eta = N/\alpha$ in [Bel+19] and $\eta = \max(N/\alpha, 200)$ in [KB19]. By default, openTSNE uses a learning rate of $\eta = N/\alpha$, following [Bel+19]. Scikit-learn used to have a learning rate of $\eta = 800$, but in version 1.2, the default was changed to $\eta = \max(N/\alpha, 50)$.

Number of Iterations. [MH08] suggest a total number of $T = 1000$ iterations. [Bel+19] test t-SNE on large datasets and point out that a higher number of iterations is needed, see Section 5.3. This value $T = 1000$ is used in scikit-learn, whereas openTSNE only uses 500 iterations after EE, leading to a total of $T = 750$ iterations.

Early Exaggeration. In the original t-SNE paper, an exaggeration value of $\alpha = 4$ is proposed for the first 50 iterations of the algorithm [MH08]. In [Maa14], it is mentioned that EE is increasingly important for large datasets, and the recommended value is $\alpha = 12$ for 250 iterations. [Bel+19] compare embeddings with different values for EE and recommend using values between 4 and 12, suggesting that the exact value does not impact the quality of the embedding too much. In [LS22], $\alpha = N/10$ is used, the only time where an adaptive EE factor based on dataset size has been suggested. Both in openTSNE and scikit-learn implementations of t-SNE, $\alpha = 12$ is used for 250 iterations.

It has been proposed by [BBK22] that stronger attractive forces, through higher α values, result in a better representation of continuous manifold structures, whereas a smaller α value or no exaggeration altogether leads to a better recovery of discrete cluster structures. [KB19] also experiment with keeping exaggeration on for the entirety of the embedding. They suggest using an exaggeration factor of 4 after the EE phase is over for especially large datasets.

The amplification of the attractive force via exaggeration is also studied in [BBK22]. They point out that, by design, t-SNE tries to use all available space for the low-dimensional embedding. This means that clusters are sometimes only separated by thin boundaries, a

feature that can turn into a problem for particularly large datasets. Using exaggeration throughout the entire t-SNE run can lead to embeddings with more space between clusters. It was even shown that t-SNE with $\alpha = 4$ after EE produces UMAP-like embeddings and an even higher exaggeration leads to embeddings similar to the ones obtained when using Laplacian eigenmaps, which highlights the extreme flexibility of t-SNE.

Degrees of Freedom. [Kob+20] suggests that we can vary the degrees of freedom in the t-distribution used. Heavier-tailed distributions than the Cauchy distribution lead to smaller clusters and sub-clusters being visible.

5.3. Automated Stopping

One problem of standard software packages for t-SNE is that there is a default number of iterations, both for the early exaggeration phase as well as for the overall embedding, see Section 5.2. This is beneficial for practitioners who do not have a lot of experience with the details of the algorithm, but it does suggest that there is a "one size fits all" solution. However, Belkina et al. [Bel+19] find that for very large datasets, the standard number of iterations does not suffice.

In order to solve this problem, [Bel+19] propose an automated approach to stopping both the EE phase and the entire algorithm once satisfactory values are reached. They observed the change in KL divergence (our loss function) to plateau after a certain number of iterations, suggesting that the embedding does not change much after this point. So they track the relative rate of KL divergence change

$$\mathrm{KLDRC}_t = \frac{\mathrm{KLD}_{t-1} - \mathrm{KLD}_t}{\mathrm{KLD}_{t-1}}$$

where t is the current iteration. The idea then is to identify the local maximum of this KL divergence relative change and stop the EE phase after it has been reached, since the change in the embeddings starts decreasing from this point onwards. This is computationally inexpensive since we already compute the value of the KL divergence at every iteration for the gradient descent update.

They suggest stopping the algorithm entirely once

$$\mathrm{KLD}_{t-1} - \mathrm{KLD}_t < \mathrm{KLD}_t / X \tag{5.1}$$

where $X = 5000$ is suggested for cytometry data.

We implement and analyze the results of this strategy in Section 6.8.

6. Experiments

In this chapter, we present a range of experiments which build on the earlier chapters. We first describe the datasets used in Section 6.1 and present several different metrics we use to evaluate our embeddings in Section 6.2. Our first experiment investigates the effect of initialization on simulated datasets, see Section 6.3. Then, we test a range of hyper-parameters (perplexity, learning rate, number of iterations and EE-related parameters) in Section 6.4 to Section 6.7. In Section 6.8, we replicate the automatic stopping procedure from [Bel+19]. The last two sections are concerned with experimental investigations of theoretical results.

6.1. Experimental Setup

We use the openTSNE library [PSZ24] for all of our experiments. Unless otherwise specified, we always use the default openTSNE settings, which are described in Section A of the appendix.

We want to perform our experiments on a variety of different datasets, namely two typically used benchmark datasets, Macosko and MNIST, one very small (Iris) and one very large dataset (Flow18):

- **Iris** flower dataset [Fis36]. This small dataset contains 50 samples from three different species of Iris flowers, thus 150 points in total. Datapoints are comprised of four features, namely the length and width of sepals and petals.

- **Macosko** mouse retina cells dataset[Mac+15]. This is a common benchmark dataset for single-cell data. It contains a total of 44,808 datapoints. The features are given by the high-dimensional gene-expression profiles outlined in [Mac+15]. Here, we apply PCA to reduce the dimensionality of the dataset to 50 before performing t-SNE, as suggested in [Maa14].

- **MNIST** [Den12]: A classic computer vision dataset, contains $N = 70,000$ gray scale images of handwritten digits 0 through 9 (i.e. 10 classes). Each image has dimensionality $D = 28 \cdot 28 = 784$. Before applying t-SNE, we rescale the entries to values in the interval $[0, 1]$ and reduce the dimensionality of data points to 50 using PCA as before.

- **Flow18** cytometry dataset [Bel19]. One million datapoints, samples are human PBMC (peripheral blood mononuclear cells), for example different types of T cells, white blood cells or B cells. The dataset consists of 16 different classes of cells. For each cell, 18 parameters are recorded, of which we use 11 for our visualization, just as in [Bel+19].

6.2. Quality Evaluation Metrics

Measuring the quality of t-SNE embeddings is challenging. Of course, since the point of t-SNE is primarily visualization, we will look at the scatter plots the algorithm produces first and foremost.

However, we will also use several other quality evaluation metrics in order to get a broader understanding of the quality of our results:

- **KLD**: As in [Bel+19], we measure Kullback-Leibler divergence throughout the algorithm and compare endpoint KLD values; the smaller the better. It should be noted that KLD is not a good measure when comparing t-SNE embeddings with different perplexity values since this directly affects KLD. The same is true to some extent for early exaggeration.

- **KNN**: For every data point, we measure the fraction of k-nearest neighbors in the high-dimensional space that are preserved as k-nearest neighbors in the low-dimensional embedding space. We set $k = 10$. Typically KNN is used to measure how well local structures are preserved.

- **KNC**: This measures the percentage of k-nearest class means preserved in the low-dimensional embedding, averaged across all classes. We use $k = 3$. This "quantifies perservation of the mesoscopic structure" [KB19].

- **CPD**: We calculate the Spearman correlation between high and low-dimensional distances across 1000 randomly chosen pairs of distances, with values ranging from -1 to 1. This is repeated 10 times, we take the averaged result, as suggested by [Web23]. This is a measure to quantify the global structure of our data.

The KNN, KNC and CPD measures are used in [KB19], from which we also use their Python implementation.

6.3. Initialization

As mentioned in Section 5.2, the benefits of PCA initialization are well-studied. We thus do not perform comparisons of t-SNE with and without PCA initialization across all of the above mentioned datasets. Instead, we want to focus on a small proof-of-concept example: [KL21] show that t-SNE is able to approximately recover a synthetic 2D circle when initialized with PCA, but not without a PCA initialization. Building on this work, we test this on other geometric structures: a triangle and a square.

To do this, we sample $N = 7000$ points and add some Gaussian noise. We use the code provided in [KL21], only changing the geometric shape of the data and setting perplexity to 40. We also use two different seeds for each of the embeddings with random and PCA initialization in order to evaluate how much PCA initialization reduces the variance between embeddings, see Figures 6.1 and 6.2.

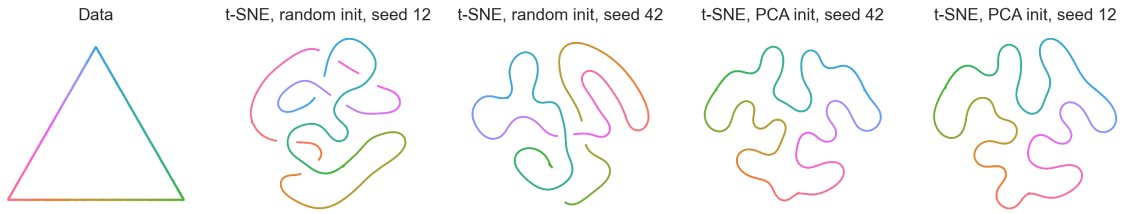

Figure 6.1.: t-SNE on an equilateral triangle in $\mathbb{R}^2$.

Figure 6.2.: t-SNE performed on a two-dimensional square.

As expected, we see much more variance in the embeddings when we use random initialization. While using PCA initialization eliminates some of this variance between embeddings, we can still observe small differences between the embeddings, especially of the triangle.

The embeddings also provide a nice visualization of PCA initialization, improving global structure. Take, for example, the red and green points in the triangle. These are far apart in the original data, so one would expect them to also be far apart in the embedding. But, in the embedding with random initialization and seed 12, we see that, in fact, the embedding lines cross each other. The PCA-initialized embeddings notably improve upon this.

6.4. Perplexity

As one of the key parameters of t-SNE, we want to study its impact on the embedding across datasets. We consider perplexity values in the standard range $\kappa = \{20, 30, 40, 50\}$, as well as adaptive ones based on the dataset size, as suggested by [KB19]. We compare two different adaptive perplexities $\kappa = \{N/1000, N/100\}$ with the standard value of $\kappa = 30$. Here, we want to test whether a higher perplexity leads to a better capture of global structure. For the adaptive perplexity comparison, we only use the Macosko and MNIST datasets since perplexities of $N/1000$ do not make sense for the small Iris dataset and computing a t-SNE embedding with perplexity $N/100$ for Flow18 would be very expensive computationally.

We see that the impact of perplexity on the embedding when choosing values from the common range is not very significant, see Appendix Figure 1. If anything, we observe a slight trade-off between local and global structure. The KNN values in Appendix, Figure 2 suggest that smaller perplexities lead to embeddings that preserve local structure slightly better. On the other hand, larger perplexity values lead to a better preservation of global structure. This makes sense when we think back to perplexity being a measure of the number of neighbors we consider when constructing the high-dimensional affinities.

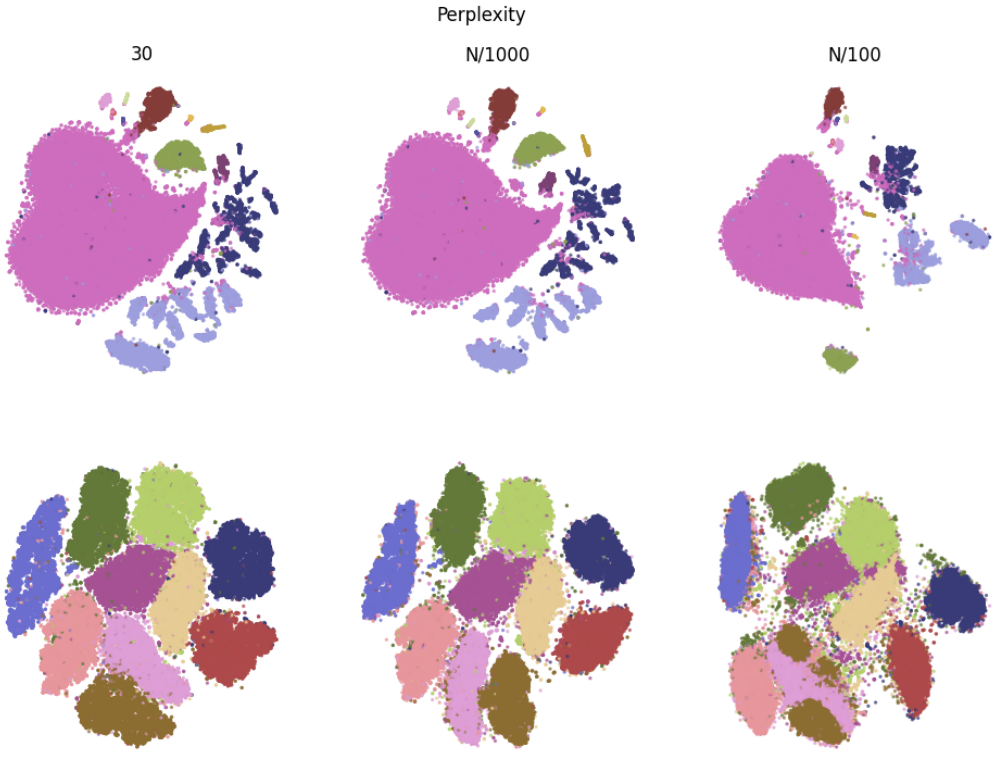

Figure 6.3.: Standard versus adaptive perplexity values on Macosko (first row) and MNIST (second row) datasets.

We observe the same phenomenon when using adaptive perplexities, see Figure 6.3. Since

$N/1000$ has the same order of magnitude as 30 for the datasets considered, we do not see big differences between these embeddings. But the comparatively large adaptive perplexity $N/100$ suggested by [KB19] produces embeddings where clusters are not as well visualized. It does, however, preserve global structure better, see Figure 6.4. But, as expected, a better global structure comes at a cost: for perplexity 30, the MNIST embedding took 183 seconds to run. Using $N/100$, it takes 1045 seconds.

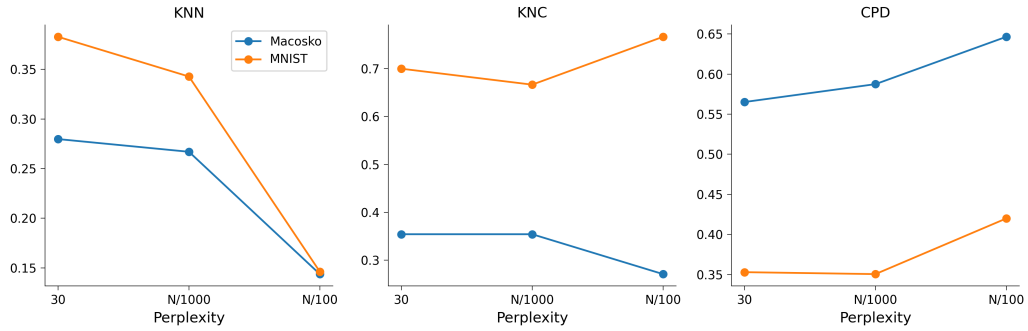

Figure 6.4.: Quality of t-SNE embeddings with different (adaptive) perplexities on the Macosko and MNIST datasets.

6.5. Learning Rate

In this section, we consider the effect of different learning rates on the embedding since different suggestions for the optimal η exist. We try $\eta = 1$ ([LS22]), $\eta = 200$ ([Maa14]), $\eta = 800$ ([Bui+13]) and $\eta = N/\alpha$ and the *auto* learning rate used in openTSNE, which adapts according to the current exaggeration used.

Choosing $\eta = 1$ does not lead to a good visualization of the different clusters for the larger datasets, as can be observed both in the scatter plots in Figure 6.6 and in the KNN measures Figure 6.5. The best results are achieved with the "auto" setting across the board, with the difference being most pronounced with the larger datasets, see Figure 6.7.

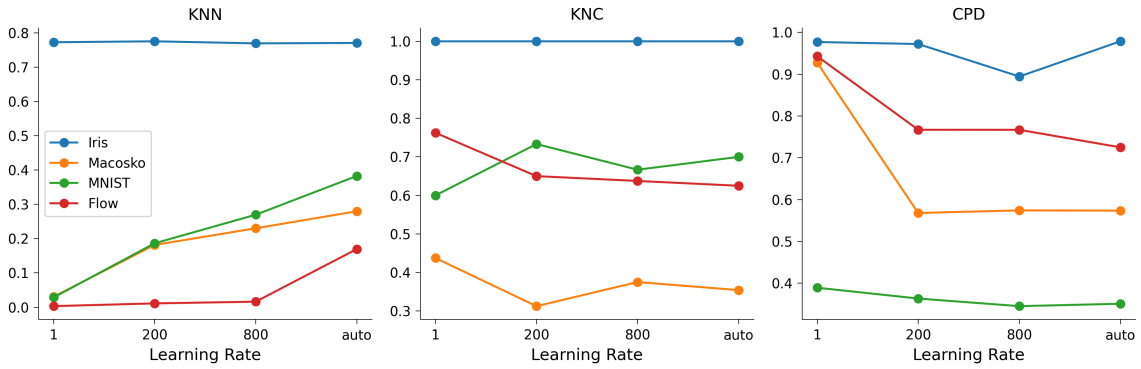

Figure 6.5.: Quality measures of t-SNE embeddings with different learning rates across all four datasets.

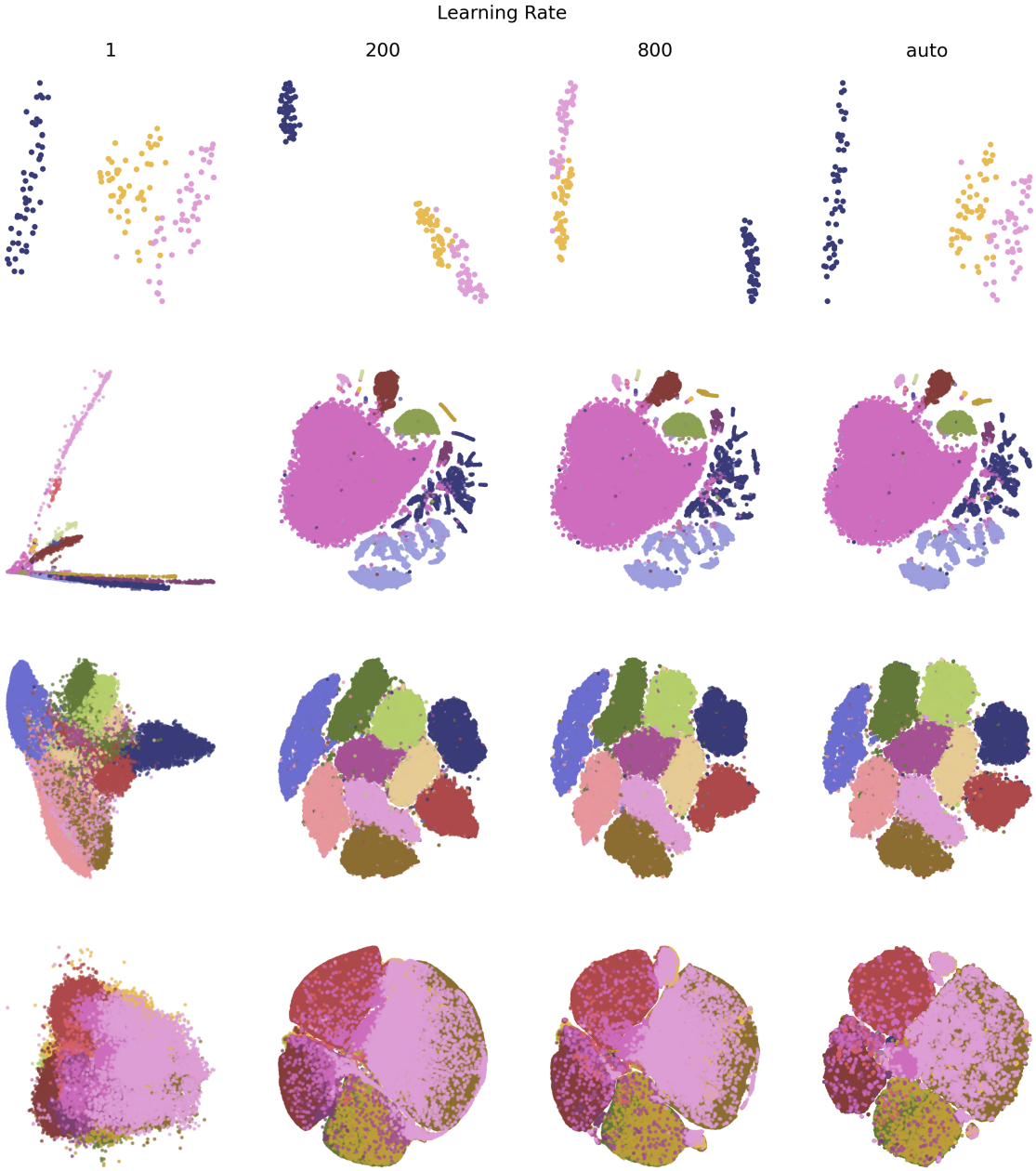

Figure 6.6.: Comparison of t-SNE embeddings using different learning rates on the Iris (first row), Macosko (second row), MNIST (third row) and Flow18 datasets (last row).

As for the Iris KL divergence plot, we see that quickly oscillating lines occur when using too large learning rates on the small Iris dataset, see Figure 6.7. This is a commonly seen phenomenon in machine learning and indicates the need for a lower learning rate.

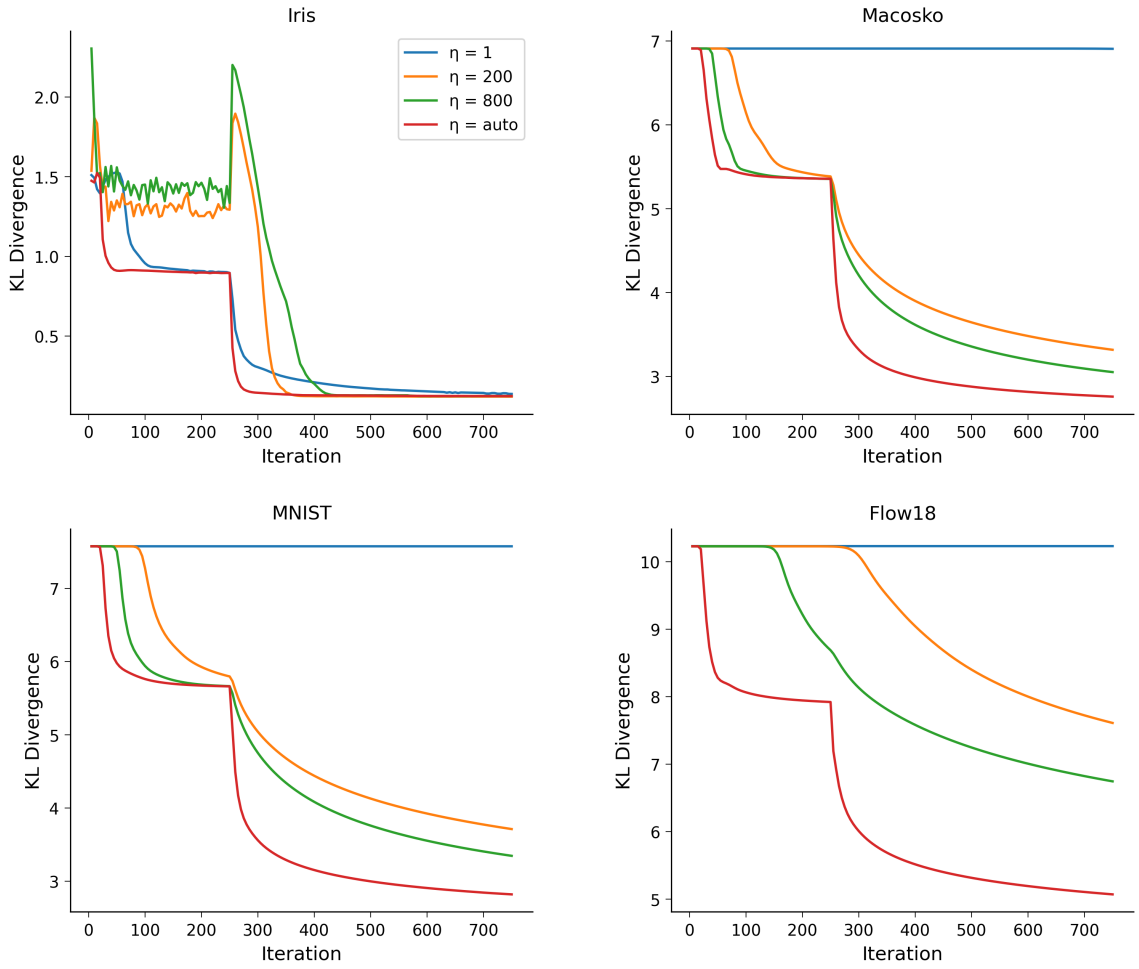

Figure 6.7.: Comparison of KLD values plotted throughout all iterations of the t-SNE runs on four datasets with different learning rates.

6.6. Number of Iterations

To test the hypothesis that bigger datasets profit from a longer t-SNE run [Bel+19], we ran t-SNE with a different number of iterations on the four datasets above. We use the following values in our experiments: $T = \{100, 750, 1500, 3000\}$. The EE length was kept at 25 percent of the total number of iterations.

We can observe that using only 100 iterations is enough on the small Iris dataset Figure 6.8.

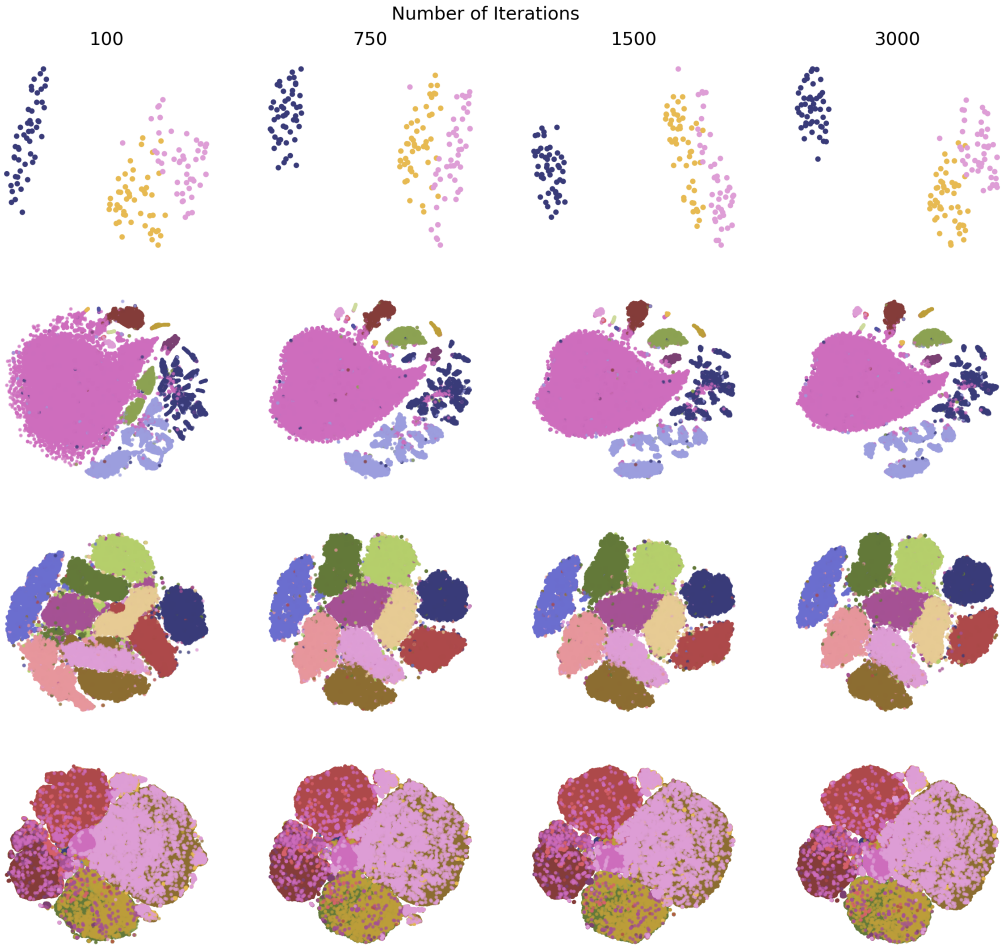

Figure 6.8.: Comparison of t-SNE embeddings using a different number of iterations on the Iris (first row), Macosko (second row), MNIST (third row) and Flow18 datasets (last row).

With 100 iterations, the Iris embedding took 0.55 seconds to run, whereas the standard number of 750 iterations took 2.42 seconds.

We also see that the largest dataset, Flow18, profits the most from the largest number of iterations since its KLD and KNN values drop most with more iterations Figure 6.9, which is also expected, see [Bel+19].

In general, our experiments seem to confirm our hypothesis that the larger the dataset, the more it profits from more iterations. Nevertheless, when looking only at the embedding grid Figure 6.8, we do not see very big differences between the embeddings from 750 iterations onward.

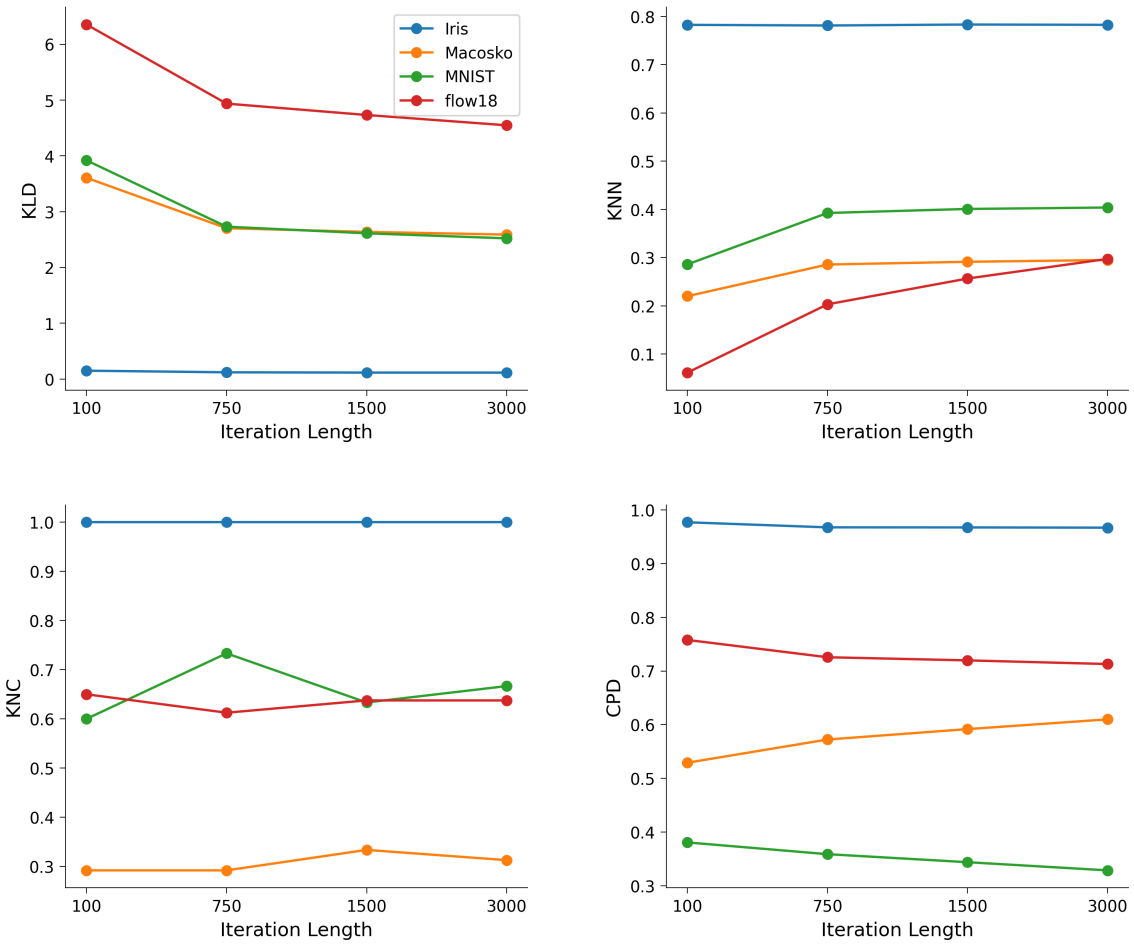

Figure 6.9.: Quality measures including endpoint KLD values of t-SNE embeddings with a different number of iterations across the four datasets.

6.7. Early Exaggeration

We analyze both the early exaggeration factor α and the length of EE. For the parameter α, we try out the values $\{4, 12, N/10\}$, in line with suggestions from the literature, as well as no exaggeration.

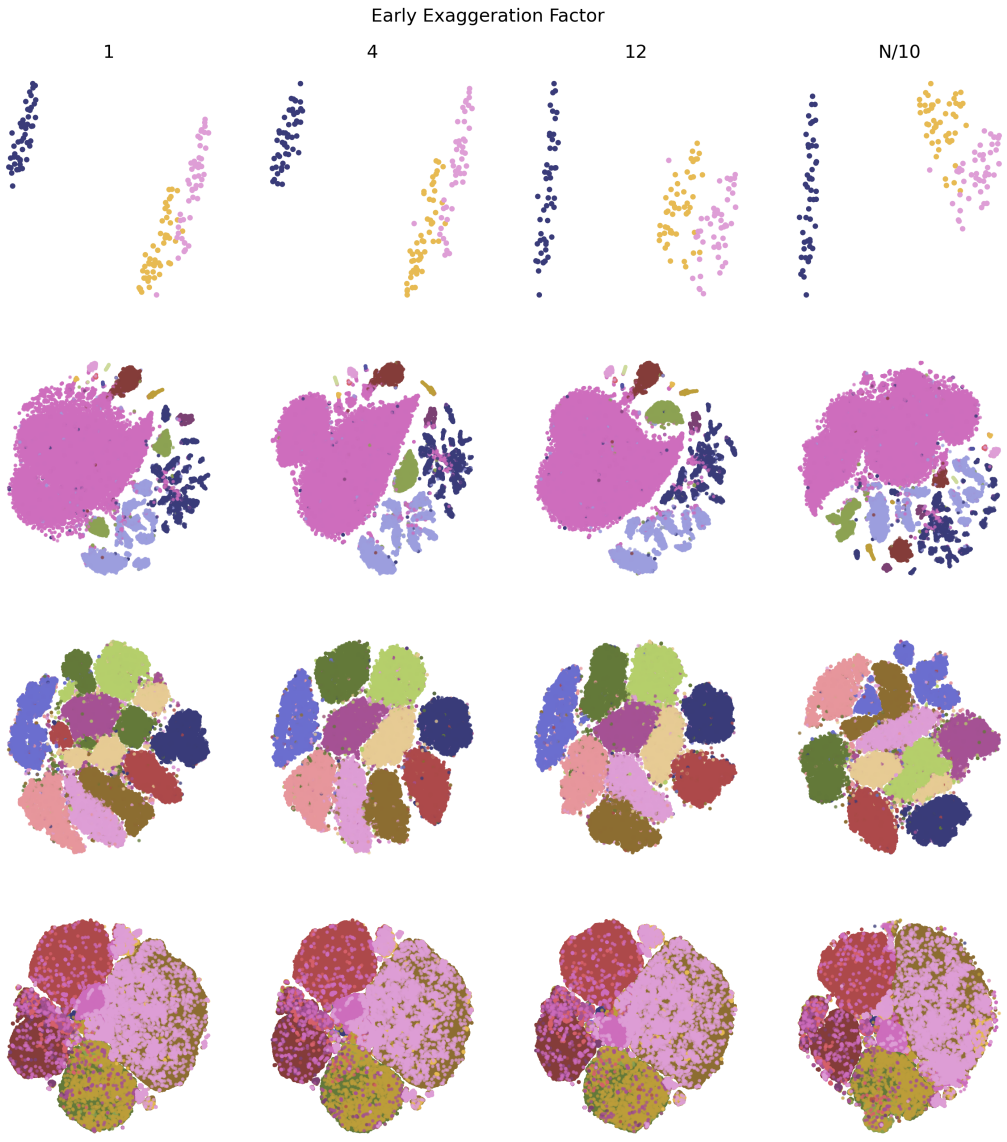

Figure 6.10.: Comparison of t-SNE embeddings using different EE factors α on the Iris (first row), Macosko (second row), MNIST (third row) and Flow18 datasets (last row).

Regarding the EE factor, we can see in Figure 6.10 that values 4 and 12 work best, coinciding with the consensus. For $\alpha = 1$, some of the clusters are split up in the Macosko dataset, for example, the light green one. This is also the case for the MNIST dataset: the blue and brown clusters are separated in the $\alpha = N/10$ case.

We now examine the length of EE, using endpoints of $125, 250, 500$ and 750, the latter coinciding with the end of the algorithm. In general, we observe better KNN values for a smaller number of iterations with early exaggeration, see Figures 6.11 and 6.12. KNN significantly drops when we never transition to a post-EE phase and keep EE on for all 750 iterations. In a similar way to results from experiments above, CPD benefits from longer EE and we do not observe significant differences across the KNC values.

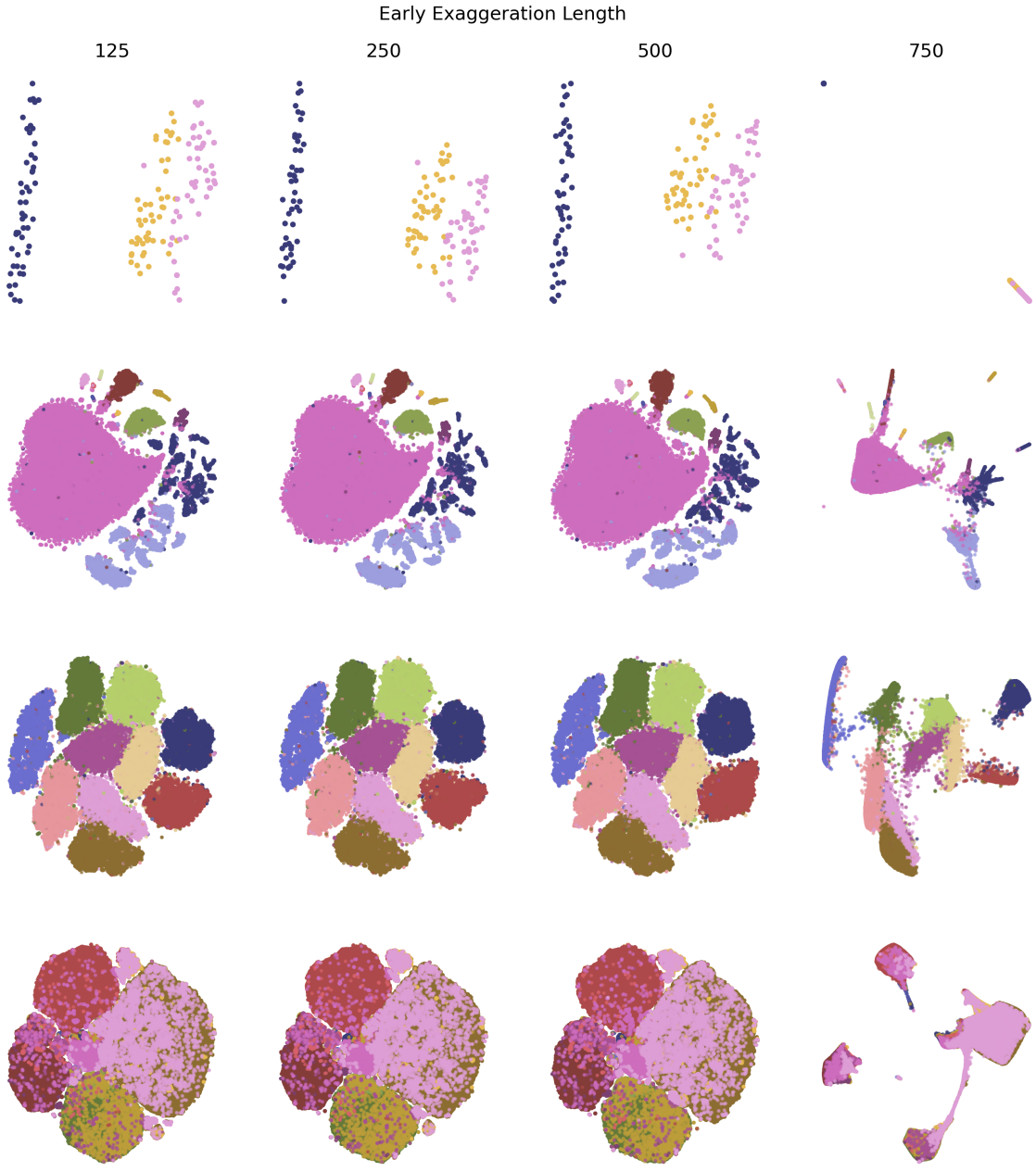

Figure 6.11.: Comparison of t-SNE embeddings using different lengths of the early exaggeration phase on the Iris (first row), Macosko (second row), MNIST (third row) and Flow18 datasets (last row).

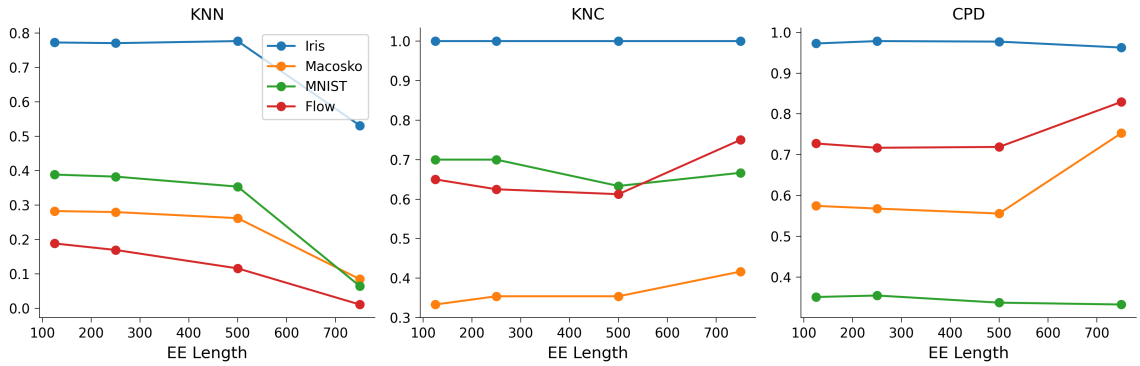

Figure 6.12.: Quality measures of t-SNE embeddings with different EE lengths across the four datasets.

6.8. Automated Stopping

Instead of fixing a certain number of iterations and a certain EE phase length, we can instead implement the automated stopping strategy outlined in Section 5.3.

In order to track KLD and KLD relative change, we implement trackers that record KLD values every 3 iterations during EE and every 5 iterations after the EE phase using the Callback functionality in openTSNE. We do not start the tracking right away but instead use a buffer of at least 15 iterations of EE and at least 150 iterations after EE, following [Bel+19].

To locate the stopping point for EE, we need to find the maximum of KLDRC. We stop the EE phase once KLDRC has decreased two times in a row. For the X value in (5.1), we choose $X = 5000$. These are all the same settings as used in the original C++ implementation of opt-SNE, see [Bel+19].

From looking at the embeddings in Figure 6.13, it seems like the standard version of t-SNE outperforms the version with automated stopping. In particular, some of the clusters are split up in the optimized plots, for example, the green cluster in the Macosko dataset or the yellow cluster in MNIST. While it is hard to know why exactly this happens, it might be the case that stopping the EE phase this early is not as benefitial and that more iterations are needed for tight clusters to form. It would be interesting to try out the same strategy with different parameters, e.g. an $X > 5000$ or requiring KLDRC to decrease three times in a row in order to be able to pinpoint possible improvements.

When we take a look at the runtimes in Table 6.1, we see that t-SNE with automated stopping is consistently faster than standard t-SNE. The relative difference is especially pronounced the smaller the dataset, which is interesting since this automated stopping strategy was designed in order to be able to deal better with large datasets. Our results indicate that, in fact, automated stopping could also be useful not only for large datasets, but across all dataset sizes.

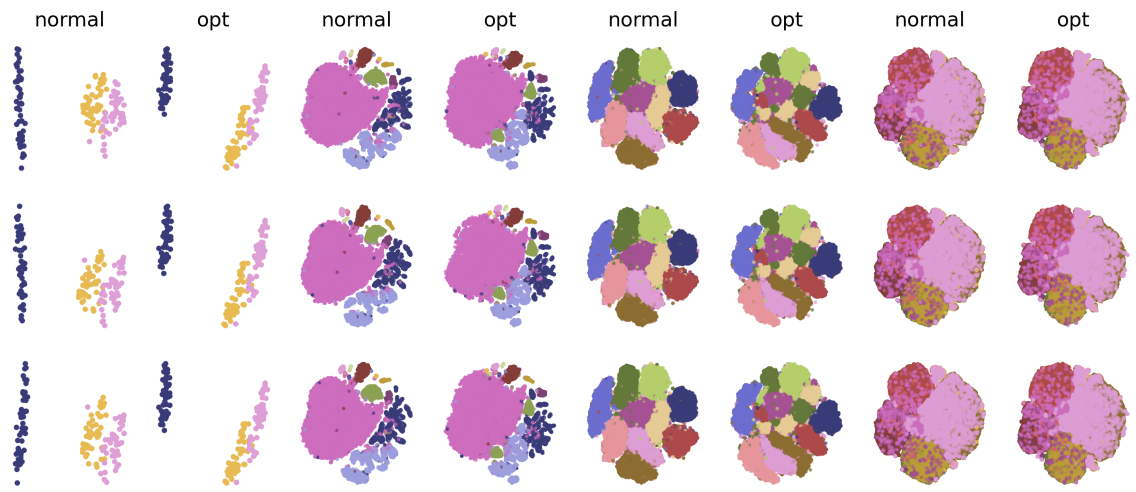

Figure 6.13.: t-SNE embeddings with (opt) and without automated stopping (normal) on the Iris, Macosko, MNIST and Flow18 datasets. Results shown for three different seeds.

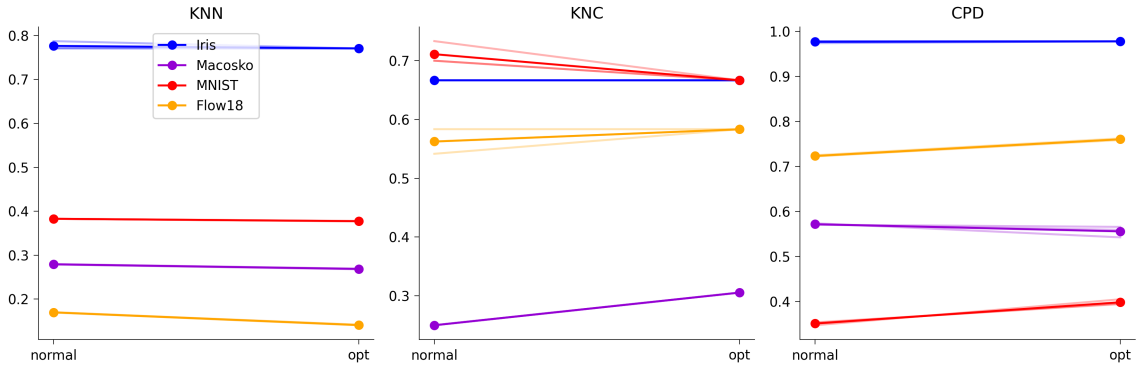

Figure 6.14.: Quality measures of t-SNE embeddings with and without automated stopping across the four datasets. We plot quality measures averaged across the three runs as well as the results for each individual seed (in a lighter color).

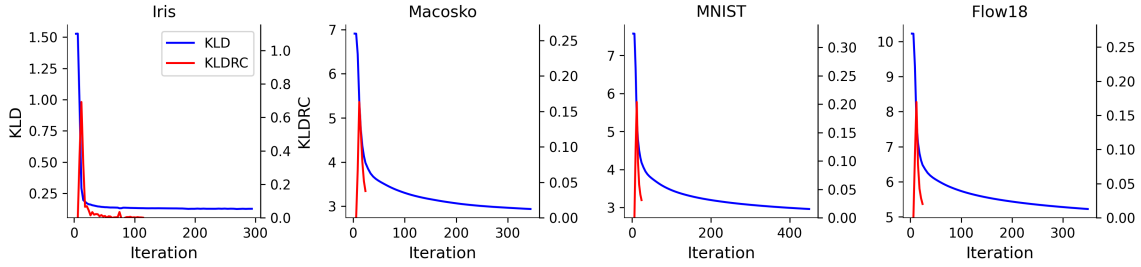

Figure 6.15.: KLD and KLD relative change of t-SNE embeddings with automated stopping plotted by iteration on the four datasets.

	Iris	**Macosko**	**MNIST**	**Flow18**
Standard t-SNE	26.1	183.4	254.4	1436.5
Using automated stopping	1.0	89.8	205.5	1223.1

Table 6.1.: Runtime in seconds of t-SNE embeddings averaged across three runs with different seeds, rounded to the nearest 0.1 seconds.

6.9. Clustering Guarantees in Practice

Here we investigate if assumptions made for the clustering result in [LS22] hold in practice, by looking at the Iris dataset. The reason that we use Iris as an example is that it is very small, which makes it feasible to look at all $N^2 = 150^2$ affinities p_{ij} individually. It is also relatively well-clustered, as t-SNE embeddings of the dataset above show. At least the dark blue cluster is very clearly separated from the other two across all parameter selections.

Recalling 4.1, we make the assumption that the data we perform t-SNE on is clustered:

$$p_{ij} \geq \frac{1}{10N|\Omega(i)|}$$

must hold for all pairs of points x_i, x_j that lie in the same cluster $\Omega(i)$. The question we now aim to answer is: is this condition fulfilled for a small, real-world dataset like Iris, where t-SNE is consistently able to visualize at least some clusters?

Ordering the Iris data points in terms of cluster membership (there are 50 points of each of the three clusters), we can visualize what the high-dimensional similarity values p_{ij} look like, see Figure 6.16. The three clusters are all visible, the first one more clearly than the other two.

So, we would expect that at least the first cluster satisfies the condition that

$$p_{ij} \geq \frac{1}{10N|\Omega(i)|} = \frac{1}{10 \cdot 150 \cdot 50} \approx 1.3 \cdot 10^{-5}.$$

However, when we analyze all p_{ij} values of the first cluster (excluding, of course, the diagonal elements, since they are always zero), we notice that the smallest p_{ij} value we observe is approximately $2.8 \cdot 10^{-7}$, lower than the supposed minimum bound above. Furthermore, there are a total number of 354 p_{ij} values between points in this first cluster are smaller than the lower bound required for it to be a cluster. Now, while [LS22] do mention that there is some flexibility with respect to the exact constant being used - we would have to increase it by two orders of magnitude for this result to hold.

For the other two clusters, we also observe an interesting phenomenon. Say we consider them as the same cluster, since they are often embedded close to each other with t-SNE.

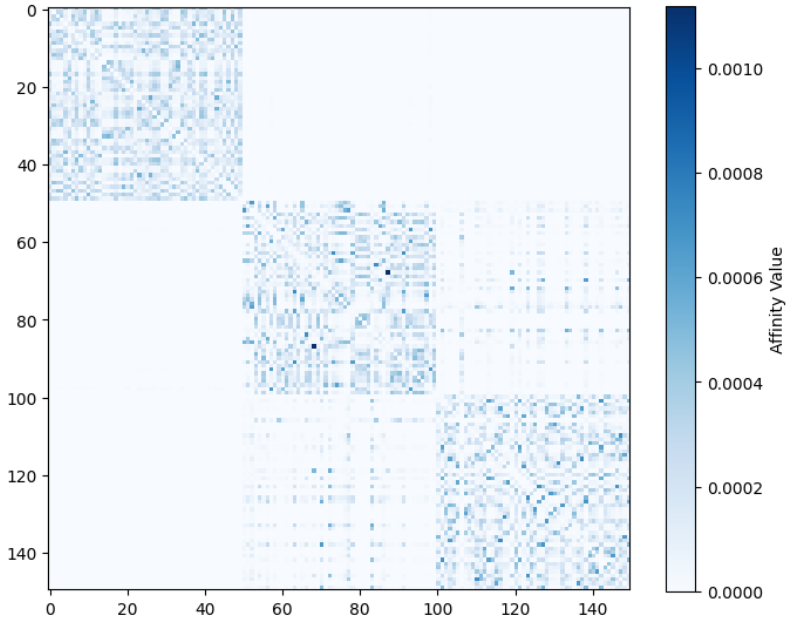

Figure 6.16.: All pairwise affinity values p_{ij} of the Iris dataset visualized. Points are ordered by cluster. The affinity matrix was generated using the standard perplexity value of 30.

Then we observe that the smallest p_{ij} between two points in this cluster is zero. This makes it virtually impossible to consider as a cluster, no matter how far we would scale the constant. One might argue that clusters 2 and 3 should be considered separately, but we also observe a minimal $p_{ij} = 0$ for one of them. This could be because not all points within the clusters are amongst the 3κ nearest neighbors of each other, at which point the p_{ij} are set to zero, see Section 5.1.

In any case, we see that even for a relatively well-clusterable and small dataset like Iris, we run into issues with the requirements of the clustering guarantee. At the very least, it cannot be claimed that the clustered data assumption is weak.

6.10. Rescaled t-SNE in Practice

We test the hypothesis that the rescaled t-SNE proposed in [MP24] is consistent in the large data limits. For this, we first have to find a sequence h_n with $nh_n^d/\log(n) \to \infty$ and $h_n \to 0$ for $n \to \infty$, where d is the dimensionality of our input data. We suggest a sequence of the form $h_n = n^{-b}$, where $1/d < b < 1$. We can quickly check that the requirements are fulfilled. For $n \to \infty$, we have

$$nh_n^d/\log(n) = nn^{-b\cdot d}/\log(n) > nn^{-1/d\cdot d}/\log(n) = 1/\log(n) \to \infty \text{ and } n^{-b} \to 0.$$

We also have to choose values for κ, the base perplexity, and β, which determines how fast the perplexity grows. We use $\beta = 0.05$ and $\kappa = 15$ for all experiments.

For our experiments, we then test on data growing in size, sampled from the same distribution. We want to see whether the rescaled t-SNE proposed in [MP24] outperforms the standard algorithm, especially with respect to a consistent limit.

For our first experiment, we sample from a Gaussian Mixture Model with five components, uniformly weighted. Each Gaussian component follows a multivariate normal distribution in 50 dimensions. For every component, each element of the mean vector is sampled from $\mathcal{N}(0, 25)$. The covariance matrices are chosen as the identity matrices.

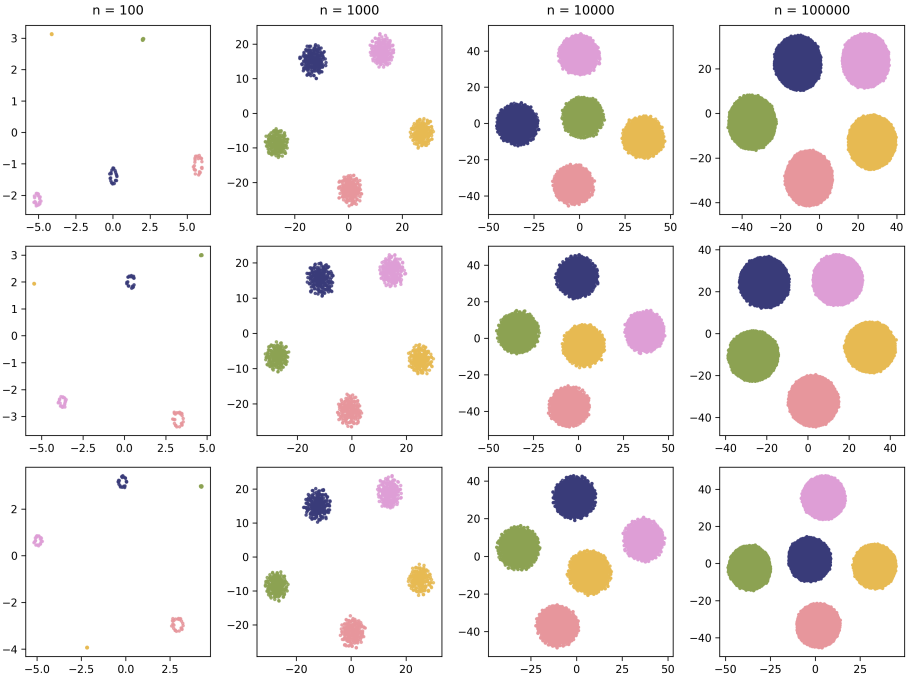

Figure 6.17.: Standard t-SNE embeddings on data generated from a mixture of five Gaussian distributions with increasing sample size. Rows show embeddings with different seeds.

We also explore how well this rescaled t-SNE works on real data, using MNIST as an example Figures 3 and 4. For this, we sample an increasing amount of points from the MNIST dataset.

We conclude that the rescaled t-SNE version does indeed seem to be more consistent in the large data limit. It especially seems to be expanding less in the simulated data case in Figure 6.18. We can also observe that there is less variation between the different t-SNE runs when using the rescaled version (Appendix Figure 6.18) when compared to standard t-SNE, see Appendix Figure 6.17.

However, it should be noted that this method does not reduce the amount of parameters we need to use. One can think of b as controlling the exaggeration, with a larger b leading to a more amplified attractive force But we have replaced the perplexity parameter with two parameters κ (the base perplexity) and β, the perplexity scaling factor. It is not obvious how to choose all these, and using the rescaled method does not eliminate the need for

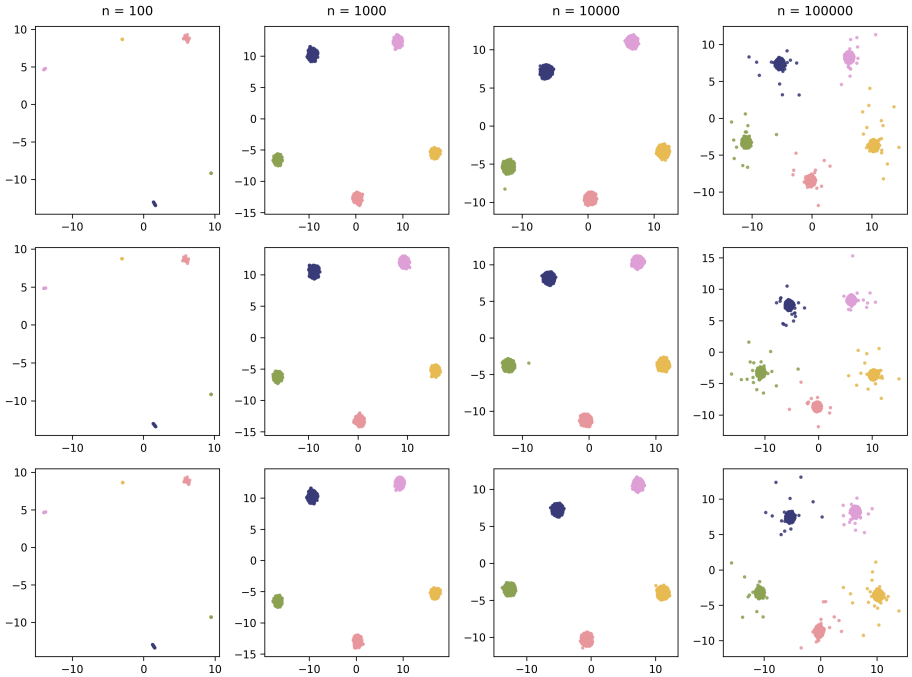

Figure 6.18.: Rescaled t-SNE embeddings with $b = 0.03$ on data generated from Gaussian Mixture Model with increasing sample size. The rows show embeddings with different seeds.

fine-tuning parameters. See, for instance, Appendix Figure 5. A smaller value of b leads to very localized embeddings, but visibility of the cluster structure decreases.

7. Conclusion

In this thesis, we explored both theoretical and practical aspects of the t-SNE algorithm with a focus on how different parameter settings influence the quality of the resulting embeddings. We were especially interested in understanding t-SNE's behavior on real-world datasets and investigating different suggested parameter settings and improvements of the algorithm.

Through small examples, we demonstrate that initializing t-SNE with PCA leads to a better preservation of a range of geometric structures. We thereby contribute to the existing evidence that PCA initialization provides the best starting point for the embedding process. Our investigation of the perplexity parameter also confirms previous findings: larger perplexity values tend to enhance global structure preservation while incurring higher computational costs and reducing the preservation of local structures. Similarly, the adaptive *auto* setting for the learning rate, as implemented in openTSNE, consistently performed well across various datasets. This demonstrates the efficacy of adaptive parameter choices. Moreover, our results indicate that larger datasets benefit from longer iteration lengths, which is also something that has been suggested before.

Regarding EE, our findings suggest that using factors within the commonly recommended range promotes effective cluster formation and minimizes the risk of clusters splitting. However, there appears to be a trade-off between preserving global versus local structure: shorter periods of EE improve local detail, while longer periods tend to maintain a more coherent global organization, albeit at the expense of visual clarity. Our implementation of the automated stopping criteria from [Bel+19] notably reduced runtimes across all datasets, even though it did not lead to a measurable improvement in embedding quality. We believe that the idea of automated stopping is useful, even though it did not lead to qualitatively superior results. The exact choice of parameters or even stopping criteria could be an interesting area of further research.

On the theoretical side, our experiments call into question the clustering guarantee presented in [LS22], as even a small, well-clustered dataset like Iris did not meet the assumptions made in their paper. In contrast, our implementation of the rescaled t-SNE variant, as suggested in [MP24], was more promising. Using the rescaled version of t-SNE, embeddings indeed became more consistent in the large data limit, supporting the central claim of the paper.

Altogether, it still remains difficult to interpret a given t-SNE embedding. Different parameter choices can lead to very different results and it can be difficult to evaluate how

good a given embedding is. This is especially the case for exploratory data analysis or cases where we have unlabeled data. As pointed out by [WVJ16], it is easy to see clusters in t-SNE plots where none exist in the underlying data, and as our experiments show, one cannot make any statements about global structure and distances in the t-SNE embeddings (remember, distances are not preserved). In particular, one should be cautious when t-SNE is used for outlier detection. It is simply not designed for such use cases.

Furthermore, it is not clear which metrics to use in order to measure the quality of the embeddings. We chose to use certain commonly used measures, but looking at other quality measures, e.g. rank-based criteria [LV09] might be an interesting direction for further research.

Moreover, it would be interesting to study the interplay between different parameters more. In this thesis, we only changed one parameter at a time, but it could be interesting to see how they interact.

Finally, there are still a lot of opportunities for a theoretical analysis of t-SNE. A particular difficulty with regard to clustering guarantees seems to be defining what even constitutes clustered data, a concept that is certainly difficult to formalize. Another very exciting area is the comparison of t-SNE to other dimensionality reduction methods. Here, the use of attraction and repulsion dynamics seems to be of central importance and is an active area of research, with new algorithms constantly being proposed, see [LC24].

Appendix

A. openTSNE Default Settings

In the following, we present the default settings of the openTSNE library that were relevant for our experiments.

- `n_component`: Dimension of the embedding space. Default: 2
- `perplexity`: 30
- `learning_rate`: "auto"
- `early_exaggeration_iter`: Length of EE phase. Default: 250
- `early_exaggeration`: EE factor. Default: 12
- `n_iter`: Number of iterations to run in the normal optimization regime. Default: 500
- `exaggeration`: Exaggeration factor to be used during the normal optimization regime. Default: None
- `dof`: Degrees of freedom of the Student t-distribution. Default: 1
- `theta`: Only used when `negative_gradient_method="bh"`. Trade-off parameter between speed and accuracy of the Barnes-Hut method. Default: "auto"
- `n_interpolation_points`: Only used when `negative_gradient_method="fft"` or its other aliases. Number of interpolation points to be used within each grid cell for interpolation based t-SNE. Default: 3
- `min_num_intervals`: Only used when `negative_gradient_method="fft"` or its other aliases. Minimum number of grid cells to use. Default: 50
- `ints_in_interval`: Only used when `negative_gradient_method="fft"` or its other aliases. Indicates how large a grid cell should be. Default: 1

- `initialization`: Initial point positions to be used in the embedding space. Default: “pca”
- `metric`: Metric to be used to compute affinities between points in the original space. Default: “euclidean”
- `initial_momentum`: Momentum to be used during the EE phase. Default: 0.8
- `final_momentum`: Momentum to be used during the normal optimization phase. Default: 0.8
- `max_grad_norm`: None
- `max_step_norm`: Maximum update norm. If the norm exceeds this value, it will be clipped. Default: 5
- `n_jobs`: Number of threads to use while running t-SNE. Default: 1
- `neighbors`: Nearest neighbor method to use. Default: “auto”
- `negative_gradient_method`: Negative gradient approximation method to use. Default: "auto"

B. Additional Results

Perplexity

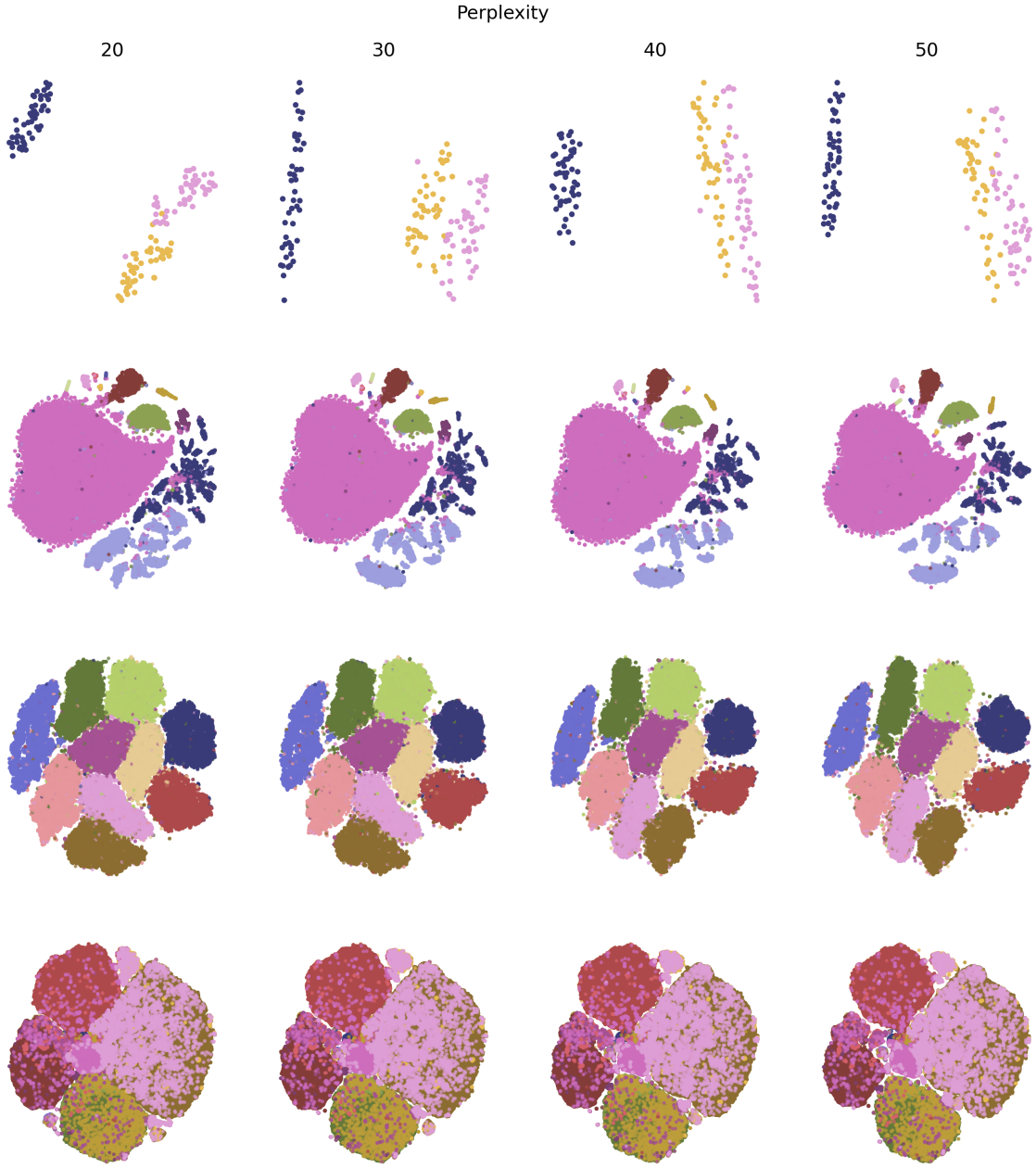

Figure 1.: Standard perlexity values on Iris (first row), Macosko (second row), MNIST (third row) and Flow18 (last row) datasets.

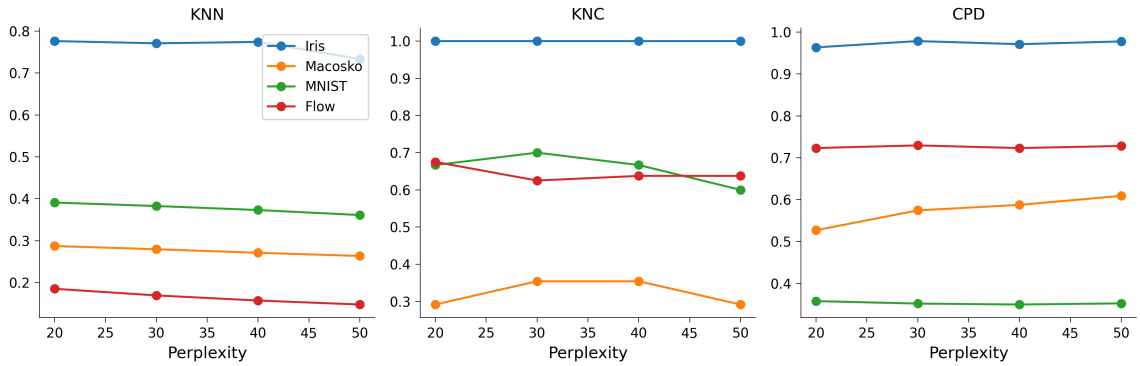

Figure 2.: Quality of t-SNE embeddings with different perplexities across all four datasets.

Rescaled t-SNE on MNIST

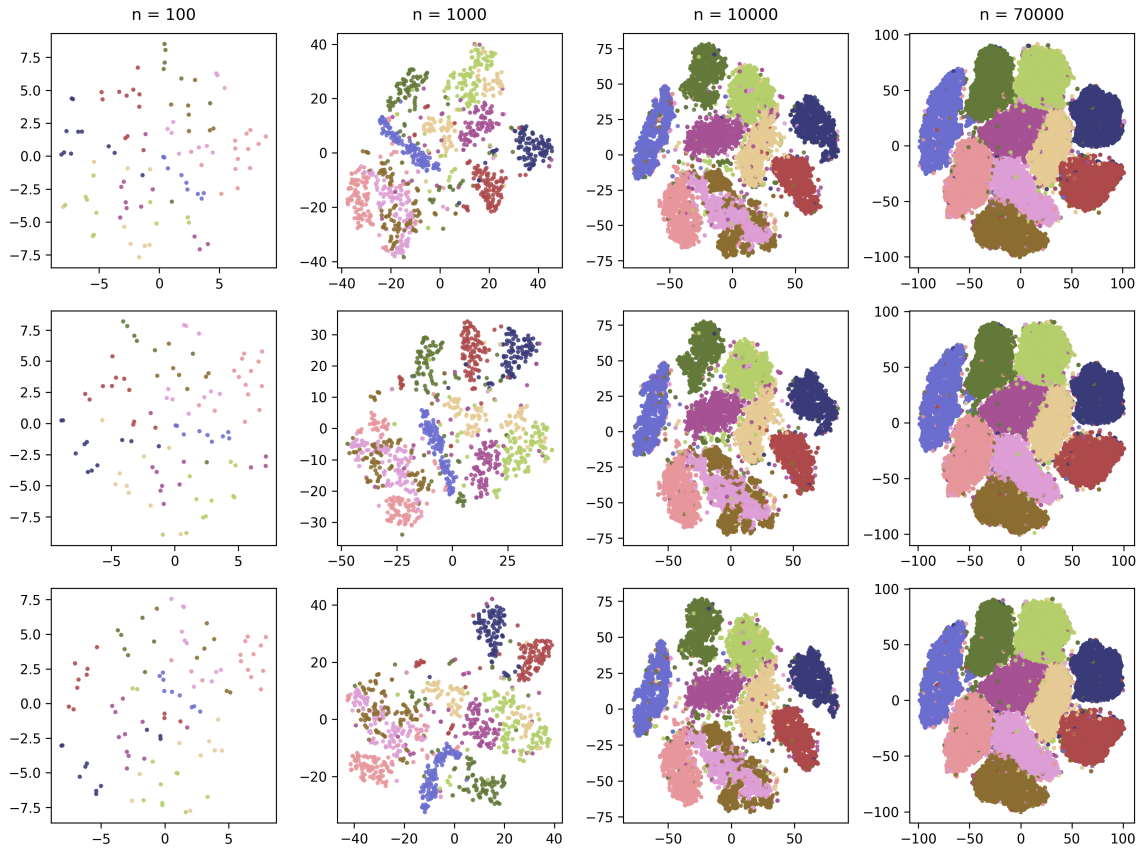

Figure 3.: Standard t-SNE embeddings on data sampled randomly MNIST with increasing sample size. The rows show embeddings with different seeds.

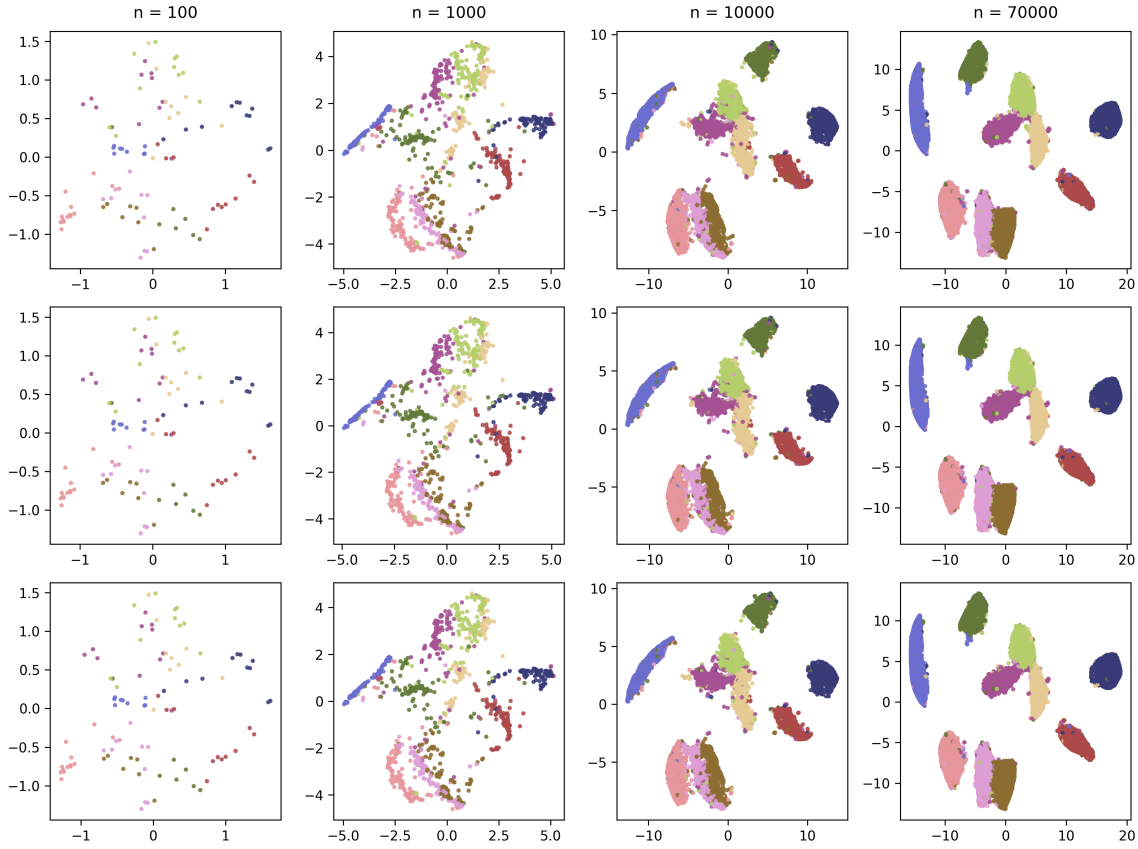

Figure 4.: Rescaled t-SNE embeddings with $b = 0.07$ on data sampled randomly MNIST with increasing sample size. The rows show embeddings with different seeds.

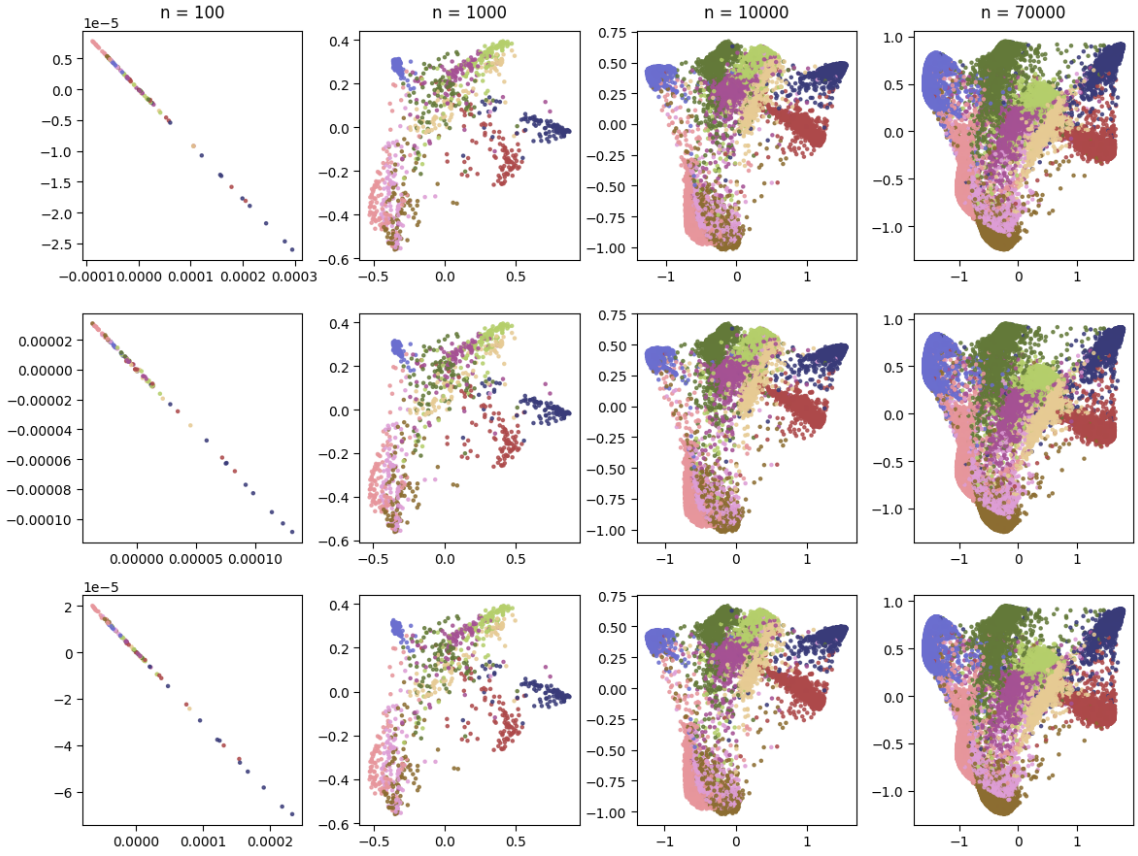

Figure 5.: Rescaled t-SNE embeddings with $b = 0.15$ and $\kappa = 20$ on data sampled randomly MNIST with increasing sample size. The rows show embeddings with different seeds.

Bibliography

[AF23] A. Auffinger and D. Fletcher. *Equilibrium Distributions for t-distributed Stochastic Neighbour Embedding*. 2023. arXiv: 2304.03727 [math.PR]. URL: https://arxiv.org/abs/2304.03727.

[AHK18] S. Arora, W. Hu, and P. K. Kothari. "An Analysis of the t-SNE Algorithm for Data Visualization". In: *Proceedings of the 31st Conference On Learning Theory*. Ed. by S. Bubeck, V. Perchet, and P. Rigollet. Vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1455–1462. URL: https://proceedings.mlr.press/v75/arora18a.html.

[BBK22] J. N. Böhm, P. Berens, and D. Kobak. "Attraction-Repulsion Spectrum in Neighbor Embeddings". In: *Journal of Machine Learning Research* 23.95 (2022), pp. 1–32. URL: http://jmlr.org/papers/v23/21-0055.html.

[Bel+19] A. C. Belkina et al. "Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets". In: *Nature Communications* 10.1 (Nov. 28, 2019), p. 5415. DOI: 10.1038/s41467-019-13055-y. URL: https://doi.org/10.1038/s41467-019-13055-y.

[Bel19] A. Belkina. "opt-SNE datasets". In: (Oct. 2019). DOI: 10.6084/m9.figshare.9927986.v1. URL: https://figshare.com/articles/dataset/opt-SNE_datasets/9927986.

[Bui+13] L. Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[Cao+19] J. Cao et al. "The single-cell transcriptional landscape of mammalian organogenesis". In: *Nature* 566.7745 (2019), pp. 496–502.

[Cie+20] M. C. Cieslak et al. "t-Distributed Stochastic Neighbor Embedding (t-SNE): A tool for eco-physiological transcriptomic analysis". In: *Marine Genomics* 51 (2020), p. 100723. ISSN: 1874-7787. DOI: https://doi.org/10.1016/j.margen.2019.100723. URL: https://www.sciencedirect.com/science/article/pii/S1874778719301746.

[CL06] R. R. Coifman and S. Lafon. "Diffusion maps". In: *Applied and Computational Harmonic Analysis* 21.1 (2006). Special Issue: Diffusion Maps and Wavelets, pp. 5–30. ISSN: 1063-5203. DOI: https://doi.org/10.1016/j.acha.2006.04.006. URL: https://www.sciencedirect.com/science/article/pii/S1063520306000546.

[CM22] T. T. Cai and R. Ma. "Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data". In: *Journal of Machine Learning Research* 23.301 (2022), pp. 1–54. URL: http://jmlr.org/papers/v23/21-0524.html.

[CW17] Y. Cao and L. Wang. *Automatic Selection of t-SNE Perplexity*. 2017. arXiv: 1708.03229 [cs.AI]. URL: https://arxiv.org/abs/1708.03229.

[DDP18] J. M. Dolezal, A. P. Dash, and E. V. Prochownik. "Diagnostic and prognostic implications of ribosomal protein transcript expression patterns in human cancers". In: *BMC Cancer* 18.1 (2018), p. 275.

[Den12] L. Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[Fis36] R. A. Fisher. *Iris*. UCI Machine Learning Repository. 1936. DOI: https://doi.org/10.24432/C56C76.

[Gan+18] P. Gang et al. "Dimensionality reduction in deep learning for chest X-ray analysis of lung cancer". In: *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*. 2018, pp. 878–883. DOI: 10.1109/ICACI.2018.8377579.

[Gar21] J. Garcke. *Scientific Computing II, Kernel Methods and Nonlinear Dimensionality Reduction*. 2021.

[Gre+20] P. Greengard et al. "Factor Clustering with t-SNE". In: *SSRN Electronic Journal* (2020). DOI: https://10.2139/ssrn.3696027. URL: https://ssrn.com/abstract=3696027.

[HR02] G. E. Hinton and S. Roweis. "Stochastic Neighbor Embedding". In: *Advances in Neural Information Processing Systems*. Ed. by S. Becker, S. Thrun, and K. Obermayer. Vol. 15. MIT Press, 2002. URL: https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf.

[HS19] Y. Hamid and M. Sugumaran. "A t-SNE based non linear dimension reduction for network intrusion detection". In: *International Journal of Information Technology* 12 (2019), pp. 125–134. URL: https://api.semanticscholar.org/CorpusID:199019113.

[Jac88] R. A. Jacobs. "Increased rates of convergence through learning rate adaptation". In: *Neural Networks* 1.4 (1988), pp. 295–307. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(88)90003-2. URL: https://www.sciencedirect.com/science/article/pii/0893608088900032.

[KB19] D. Kobak and P. Berens. "The art of using t-SNE for single-cell transcriptomics". In: *Nature Communications* 10.1 (2019), p. 5416. URL: https://github.com/berenslab/rna-seq-tsne.

[KL21] D. Kobak and G. C. Linderman. "Initialization is critical for preserving global data structure in both t-SNE and UMAP". In: *Nature Biotechnology* 39.2 (Feb. 1, 2021), pp. 156–157. DOI: 10.1038/s41587-020-00809-z. URL: https://github.com/dkobak/tsne-umap-init/.

[Kob+20] D. Kobak et al. “Heavy-Tailed Kernels Reveal a Finer Cluster Structure in t-SNE Visualisations”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by U. Brefeld et al. Cham: Springer International Publishing, 2020, pp. 124–139.

[LC24] J. Lu and J. Calder. *Attraction-Repulsion Swarming: A Generalized Framework of t-SNE via Force Normalization and Tunable Interactions*. 2024. arXiv: 2411.10617 [cs.LG]. URL: https://arxiv.org/abs/2411.10617.

[LCD22] S. Lespinats, B. Colange, and D. Dutykh. “Intrinsic Dimensionality”. In: *Nonlinear Dimensionality Reduction Techniques: A Data Structure Preservation Approach*. Cham: Springer International Publishing, 2022, pp. 31–44. ISBN: 978-3-030-81026-9. DOI: 10.1007/978-3-030-81026-9_2. URL: https://doi.org/10.1007/978-3-030-81026-9_2.

[Lin+19] G. C. Linderman et al. “Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data”. In: *Nature Methods* 16.3 (2019), pp. 243–245.

[LS22] G. C. Linderman and S. Steinerberger. “Dimensionality Reduction via Dynamical Systems: The Case of t-SNE”. In: *SIAM Review* 64.1 (2022), pp. 153–178. DOI: 10.1137/21M1446769. eprint: https://doi.org/10.1137/21M1446769. URL: https://doi.org/10.1137/21M1446769.

[LV09] J. A. Lee and M. Verleysen. “Quality assessment of dimensionality reduction: Rank-based criteria”. In: *Neurocomputing* 72.7 (2009). Advances in Machine Learning and Computational Intelligence, pp. 1431–1443. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2008.12.017. URL: https://www.sciencedirect.com/science/article/pii/S0925231209000101.

[Maa14] L. van der Maaten. “Accelerating t-SNE using Tree-Based Algorithms”. In: *Journal of Machine Learning Research* 15.93 (2014), pp. 3221–3245. URL: http://jmlr.org/papers/v15/vandermaaten14a.html.

[Mac+15] E. Z. Macosko et al. “Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets”. eng. In: *Cell* 161.5 (2015), pp. 1202–1214. ISSN: 0092-8674.

[McI+18] L. McInnes et al. “UMAP: Uniform Manifold Approximation and Projection”. In: *Journal of Open Source Software* 3.29 (2018), p. 861. DOI: 10.21105/joss.00861. URL: https://doi.org/10.21105/joss.00861.

[MH08] L. van der Maaten and G. Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[Mik+13] T. Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[MP24] R. Murray and A. Pickarski. *Large data limits and scaling laws for tSNE*. 2024. arXiv: 2410.13063 [math.ST]. URL: https://arxiv.org/abs/2410.13063.

[Pet24] T. Petroc. *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028*. Accessed on 09.03.2025. 2024. URL: https://www.statista.com/statistics/871513/worldwide-data-created/.

[PGW24] D. Peng, Z. Gui, and H. Wu. *Interpreting the Curse of Dimensionality from Distance Concentration and Manifold Effect*. 2024. arXiv: 2401.00422 [cs.LG]. URL: https://arxiv.org/abs/2401.00422.

[Pla13] A. Platzer. "Visualization of SNPs with t-SNE". In: *PLOS ONE* 8.2 (Feb. 2013), pp. 1–6. DOI: 10.1371/journal.pone.0056883. URL: https://doi.org/10.1371/journal.pone.0056883.

[PSZ24] P. G. Poličar, M. Stražar, and B. Zupan. "openTSNE: A Modular Python Library for t-SNE Dimensionality Reduction and Embedding". In: *Journal of Statistical Software* 109.3 (2024). DOI: 10.18637/jss.v109.i03. URL: https://opentsne.readthedocs.io/en/stable/api/index.html#openTSNE.TSNE.

[Sha+21] N. Sharma et al. "A Stochastic Neighbor Embedding Approach for Cancer Prediction". In: *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. 2021, pp. 599–603. DOI: 10.1109/ESCI50559.2021.9396902.

[Skr+24] M. Skrodzki et al. *Navigating Perplexity: A linear relationship with the data set size in t-SNE embeddings*. 2024. arXiv: 2308.15513 [cs.LG]. URL: https://arxiv.org/abs/2308.15513.

[SR21] P. Shetty and R. Ramprasad. "Automated knowledge extraction from polymer literature using natural language processing". In: *iScience* 24.1 (2021), p. 101922. ISSN: 2589-0042. DOI: https://doi.org/10.1016/j.isci.2020.101922. URL: https://www.sciencedirect.com/science/article/pii/S2589004220311196.

[SS17] U. Shaham and S. Steinerberger. *Stochastic Neighbor Embedding separates well-separated clusters*. 2017. arXiv: 1702.02670 [stat.ML]. URL: https://arxiv.org/abs/1702.02670.

[SS23] N. Sharma and S. Sharma. "Optimization of t-SNE by Tuning Perplexity for Dimensionality Reduction in NLP". In: *Proceedings of International Conference on Communication and Computational Technologies*. Ed. by S. Kumar et al. Singapore: Springer Nature Singapore, 2023, pp. 519–528.

[Tas+18] B. Tasic et al. "Shared and distinct transcriptomic cell types across neocortical areas". In: *Nature* 563.7729 (2018), pp. 72–78.

[TR16] E. Taskesen and M. J. T. Reinders. "2D Representation of Transcriptomes by t-SNE Exposes Relatedness between Human Tissues". In: *PLOS ONE* 11.2 (Feb. 2016), pp. 1–6. DOI: 10.1371/journal.pone.0149853. URL: https://doi.org/10.1371/journal.pone.0149853.

[Wan+18] Y. Wang et al. “A comparison of word embeddings for the biomedical natural language processing”. In: *Journal of Biomedical Informatics* 87 (2018), pp. 12–20. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2018.09.008. URL: https://www.sciencedirect.com/science/article/pii/S1532046418301825.

[Web23] P. Weber. “Uncertain Choices in Method Comparisons: An Illustration with t-SNE and UMAP”. Bachelor’s thesis. Ludwig–Maximilian–University Munich, Apr. 2023.

[WVJ16] M. Wattenberg, F. Viégas, and I. Johnson. “How to Use t-SNE Effectively”. In: *Distill* (2016). DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.

[Xue+20] J. Xue et al. “Classification and identification of unknown network protocols based on CNN and T-SNE”. In: *Journal of Physics: Conference Series* 1617.1 (Aug. 2020), p. 012071. DOI: 10.1088/1742-6596/1617/1/012071. URL: https://dx.doi.org/10.1088/1742-6596/1617/1/012071.

[YCC21] Z. Yang, Y. Chen, and J. Corander. *T-SNE Is Not Optimized to Reveal Clusters in Data*. 2021. arXiv: 2110.02573 [cs.LG]. URL: https://arxiv.org/abs/2110.02573.

[ZS21] Z. Zoghi and G. Serpen. “UNSW-NB15 Computer Security Dataset: Analysis through Visualization”. In: *CoRR* abs/2101.05067 (2021). arXiv: 2101.05067. URL: https://arxiv.org/abs/2101.05067.