Visualization beyond Riemannian Manifolds

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Jan Niklas Böhm

aus Berlin

Tübingen

2025

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	26.06.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter/-in:	PD Dr. Dmitry Kobak
2. Berichterstatter/-in:	Prof. Dr. Philipp Hennig

Abstract

This dissertation is about dataset visualization methods and recent developments in this field. Techniques like this take a set of high-dimensional points that have many different features and map it into the two-dimensional plane. This has often been achieved with neighbor embedding methods, which require that the data is locally Euclidean, also described as lying on a Riemannian manifold. This assumption has proved limiting in recent times, and methods that can work around this have been developed and are part of this thesis. Dataset visualizations are used in different research disciplines and help practicioners gain insight into complicated relationships that hide in their data. The topics in this thesis will describe how those methods work, the history of these methods, and recent developments that were researched by the author.

Zusammenfassung

Diese Dissertation beschreibt Datensatzvisualisierungsmethoden und ihre Entwicklung über die letzten Jahre. Techniken wie diese transformieren hochdimensionale Datenpunkte mit mannigfaltigen Eigenschaften in die zweidimensionale Ebene. Oftmals werden dafür Nachbarschaftseinbettungsmethoden verwendet, welche es benötigen, dass die Daten lokal Euklidisch zueinander sind, was auch als Riemann'sche Mannigfaltigkeit bezeichnet wird. Diese Erwartung hat sich als limitierend entpuppt und Methoden die dieses Problem lösen, wurden entwickelt und werden in dieser Thesis vorgestellt. Datensatzvisualisierungen werden in verschiedenen Disziplinen der Wissenschaft verwendet und helfen Praktikern dabei neue Einblicke in die komplizierten Beziehungen, welche sich in den Daten verstecken, zu erhaschen. Diese Thesis beschreibt wie diese Methoden funktionieren, die Geschichte dieser und darüber hinaus auch aktuelle Entwicklungen, die von dem Author erforscht wurden.

Contents

Preface

I would like to take this opportunity to thank everyone who has helped me along so that I could get here. Academically, my gratitude goes towards Dmitry, who has helped me in so many ways and without him, my last four years would have not looked the same. Philipp has provided even more advice and has always taken care that the environment in our group is in a healthy condition. I want to thank Sebastian, with whom I have enjoyed the academic exchange and the collaboration. The person that had the most influence on me psychologically is Rita. Without her the time in our office would have been much duller and I am glad that we got to spend so much time together, not just on an academic level, but also cooking together or going to both Rwanda and Japan.

There are many more people that have accompanied me on this journey. In the interest of brevity, let me just thank all of you, hoping I don't forget anyone: Ifeoma Nwabufo, Lisa Schmors, Jan Lause, Andrew Draganov, Thomas Höllt, Masha Tepliakova, Ziwei Huang, Kyra Kadhim, Tom Ganz, and Yves Bernaerts. I also want to thank the rest of the lab members, who I have not mentioned explicitly.

Let me also thank my proofreaders, Rita (again), and my two brothers Felix and Stefan. In general, I am delighted about all of the support I have received from my family and I want to also thank my mom and dad (no surprise there). Lastly, thank you Britt for being in my life, I appreciate that — and you — a lot.

Reflecting on my PhD makes me appreciate the last four years. I have had the pleasure of working in a laboratory where a supportive atmosphere dominates and overall it feels harmonic to be there. I would not have wanted to spend my time anywhere else.

Niklas Böhm
Tübingen, Germany
March 2025

Introduction

1

It is a truth universally acknowledged, that a dataset in possesion of good information, must be in want of a visualization. For the information contained within does not surface on its own. As mentioned by Iverson (1979), the notation of information or ideas influences the thought process behind it. The same holds for data. The presentation of data in any form influences how we interpret it (Tufte 2001) and as such this must be considered when visualizing any form of data. We require succinct notation for data if we want to have a chance of making sense of it. Long gone are the times when you could look at the set of numbers comprising a given data set and have it be small enough to keep it in your head as well as draw conclusions from it.

Kenneth Eugene Iverson (Oct. 1979). "Notation as a Tool of Thought". In: *Communications of the ACM* 23.8, pp. 444–465

For the scope of this dissertation, we concern ourselves with data visualization in the form of mapping a dataset of n points from a high-dimensional space into a two-dimensional (2D) space. As such, all of the mappings are of the form $f : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times 2}$. This is also called dataset visualization.

Dataset visualization is an important step in the analysis of data, as it makes a dataset intuitively interpretable and aids in exploration. As such, contemporary techniques have found broad application in for example the natural sciences, like single-cell data analysis (Becht et al. 2019; Diaz-Papkovich et al. 2019; Karczewski et al. 2020). Mapping complex high-dimensional data to a 2D space has improved the understanding of complex dynamics in these fields. As such, practitioners often turn to visualization methods in order to investigate their data.

What sets modern dataset visualization techniques apart from "plain" statistics is that the columns of the dataset (the feature dimensions) are either abstract or do not have an inherent quantity grounded in reality. As such, it is both related to statistics because it summarizes data, but also distinct, by inherently modeling the relationship between data points.

Historically, primitive forms of data visualization have existed for a long time. For example, geographic locations were mapped onto a sheet of paper, what is now known as a map. Maps even predate paper, as they have been created since prehistoric times, with examples dating as far back as approximately 1875 BCE (du Châtellier 1901; Nicholas et al. 2021) and 1150 BCE (Harrell & Brown 1992). By taking the birds-eye view of the landscape, it helps to *visualize* the geography and thus the understanding of how to move from one point to another.

Paul du Châtellier (1901). "Les pierres gravées de Penhoat, en Saint-Coulitz et de Sanct-Bélec, en Leuhan". In: *Bulletin de la Société Archéologique du Finistère* 28

The critical reader may now remark that every map distorts the reality it depicts; and they are completely right. The map cannot exactly represent the reality due to various reasons,[1] but it still provides useful information. In ancient times people used maps for gaining an overview of the states territory and as additional

1. One example would be trying to model the three-dimensional globe on a 2D map.

Figure 1.1: Crop of the map by Snow (1855), as lithographed by Charles Cheffins. Note the spatial correlation between the number of deceased and the water pump in the center. Image is in the public domain.

information in for example military campaigns. Despite the fact that "a map *is not* the territory" (Korzybski 1931), it still is a helpful tool.[2]

The entities displayed on a map are concrete objects that are grounded in reality, like a settlement or a river. More recently, the information displayed has become more abstract. An early example is the graphic by Snow (1855), detailing the spread of cholera in London. Without knowing the exact source of the outbreak, the spatial correlation of deaths gave the central clue which lead to the removal of the water pump that was surfacing contaminated ground water from a buried graveyard.

While the information that is displayed by Snow (1855) is still tanglible, it is no longer related to a physical object. The direction of information becoming more and more abstract has been a trend ever since, only accelerating with the advent of computing technology. Where simple datasets like the Iris flower dataset (Fisher 1936) only have four variables (both width and length for both the sepal and the petal), this is already too high-dimensional to visualize directly in 2D. The number of features only increased, for example now it is common to treat every color channel of every pixel in an image as a feature, which then creates a 3072-dimensional feature space for only small images, like the 32×32 images in the Canadian Institute for Advanced Research (CIFAR) dataset (Krizhevsky 2009).

Our visual system is attuned to recognizing spatial similarity. The visual cortex is also the most developed brain system, with a lot of cortical volume dedicated to it. So we are trained to recognize spatial patterns when the data is presented in a two-dimensional layout. This allows us to draw conclusions with more ease about either similarity or notice outliers (Treisman & Gelade 1980). Both of these patterns then inform further investigation that often leads to new insights. For some examples, see Böhm, Berens, & Kobak (2023, Figures A.10–12).

By succinctly presenting information in more and more abstract forms, the information density increased over time. From the simple beginnings that predate paper, we now have large datasets that encode the genetic expressions in cells. Simply displaying the raw data in not feasible as the number of samples and dimensions easily exceeds what can actually be read by an expert, let alone be used to draw conclusions

2. This line of thought follows similar arguments (and in fact predates) the idea that "all models are wrong" (Box 1976) because it is not possible to *exactly* model the world. In machine learning, this is often also expressed in the bias-variance tradeoff — implying that you need some underlying assumptions for your model (e. g. Bishop 2006).

about the underlying mechanisms. As such, finding a quality summarization of the data is crucial for drawing conclusions from it. Dimension reduction techniques are such a method.

Continuing to modern times, data is collected in vast quantities. When statistics was still in its cradle, people would manually take measurements of humans in order to make assumptions about them. With the field growing up, this has truly erupted with single-cell analysis now collecting millions of samples from small areas of biological tissue.

In conjunction with recent advances in machine learning, the ability to learn on large datasets has also dramatically improved. While Moore's law (Moore 1998) has slowed down for single central processing unit (CPU) speed, the parallelism inherent to graphics processing units (GPUs) has picked up the slack. The advent of deep learning has harnessed the massively parallel computing provided by GPUs which in turn fueled the rapid development of new machine learning techniques (Krizhevsky, Sutskever, & Hinton 2017).

Both the amount of data as well as the amount of available computation power has increased over the last decades. This thesis presents new approaches that can visualize datasets in novel ways as well as explain how popular algorithms for dataset visualization work.

1.1 MOTIVATION

WHY is there interest in dataset visualization methods? For most datasets, this is one of the first things that can be tried in order to get a rough understanding of what the data consists of. The more data is collected, the more important the analysis thereof becomes. In a way, dataset visualization can be thought of computing summary statistics, like the mean, of the entire dataset. The description cannot do justice to the full dataset, but it helps in understanding some underlying structure. Most people find that a dataset visualization is much more useful than the mean of the dataset as it almost literally paints a picture of all of the relationships within the dataset.

With the first actual use of such methods dating back to only the middle of the last century (see Section 1.2), it is a relatively new method for presenting information. Nevertheless, it has garnered interest in many fields of science, where complex information arises. Modern researchers are often presented with large amounts of data, which can easily grow into the millions, as is the case for single-cell data (Zhang et al. 2025) or text corpora (González Márquez et al. 2024). Especially for text data, current large language modelss (LLMs) are trained on the entire internet, easily eclipsing for example the 800 GB dataset "The Pile" (Gao et al. 2020), which is only a few years old at the time of writing.

Finding salient information in a vast sea of data is a daunting task, as the effort to sift through the entire dataset by hand ranges from prohibitively expensive to impossible. In the natural sciences, the information that is gathered can be inherently complex, even if the number of samples do not scale to sizes as large as the entire internet. While the sample size n is small, the number of dimensions d is quite large. In fields like transcriptomics, the number of dimensions correspond to the genes present in all recorded cells, and the data is the genes expression[3] values in each cell. This information is hard to directly interpret and thus some further

3. Gene expressions are the occurrence of a gene substring within the DNA/RNA. What is stored is how often this substring occured within the cell, so the features of a cell are the counts of gene expressions.

computer-aided analysis is required before the treasure trove can be opened and new scientific insight can be gathered.

This is where dataset visualization comes in. It takes the features, which can grow into the 10 000s, and maps every cell down to 2D. The resulting map can then be inspected by an expert, who can derive relationships between cell cohorts and validate those for their experiments. This technique is pervasive and used throughout laboratories across the world.

While analysis of single-cell corpora is certainly something that invites dataset visualization, those techniques are by no means limited to just this one domain. Text corpora have been analyzed with the same kind of visualization methods (Schmidt 2018; González Márquez et al. 2024). There, features are first extracted from the text and then subsequently visualized in 2D. Other domains include for example analyzing activations of a neural network (for example Mnih et al. 2015; Kipf & Welling 2017), and visualizing maximally exciting images (MEIs) for the visual cortex in the Macaque brain (Willeke et al. 2023).

For dataset visualization, the most prominent methods are based on the concept of neighbor embedding (NE), where the underlying idea is to preserve neighbors from the high-dimensional space in the low-dimensional (in this case always 2D) space. The most popular techniques for this are called t-Distributed Stochastic Neighborhood Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP). Both are widely used and have different strengths and weaknesses (Chapter 2).

Criticism of visualization methods

While the methods have so far been presented as a panacea for all kinds of scientific endeavours, dataset visualizations also had their fair share of criticism. One common criticism is the fact that the distances of NE methods are abstract and can be distorted. Since these methods are designed to preserve the neighborship, data points which are not neighbors can be placed almost arbitrarily with respect to each other, as most loss functions only exert a repulsive force on them that decays with the distance. Common pitfalls for interpreting t-SNE have been outlined by Wattenberg, Viégas, & Johnson (2016).

Perhaps most prominent single critique is the work by Chari & Pachter (2023), who claim that state-of-the-art methods are as good as a method that can create arbitrary shapes.[4] This work received a rebuttal by Lause, Berens, & Kobak (2024) , who claim that aforementioned paper bases their claims on the wrong metrics. Lause, Berens, & Kobak (2024) demonstrate that there are in fact significant differences between the various methods. That is to say that NE methods do preserve meaningful relationships of the data in the 2D layout that they produce, unlike others which enforce a 2D elephant.

Most NE methods have a large set of parameters that influence the embedding. For example, the popular openTSNE implementation by Poličar, Strazar, & Zupan (2019) in the version 1.0.2 features 26 parameters, most of which are directly related to the embedding layout. For UMAP (McInnes, Healy, & Melville 2018), there are 39 parameters in the version 0.5.4. How these parameters influence the embedding is not immediately clear to most researches using t-SNE or UMAP — it is often a source of confusion for practitioners. The correct application of these models is not trivial and requires some care in parameter selection. As Grobecker, Sakoparnig, & van Nimwegen (2024) point out: "many tunable parameters [...] in practice seem

4. They give a prominent example of a visualization in the form of an elephant, which clearly distorts the embedding and remove much of the visual quality that you would expect well-known datasets to exhibit.

Jan Lause, Philipp Berens, & Dmitry Kobak (Oct. 2024). "The art of seeing the elephant in the room: 2D embeddings of single-cell data do make sense". In: *PLOS Computational Biology* 20.10, pp. 1–5

to be mostly set by trial-and-error". Nevertheless, over the years there have been various publications that focus on best practices for parameter selection and on explaining the mathematical intuition behind various parameters. These automate the parameter selection and thus simplify the usage of dataset visualization methods (Belkina et al. 2019; Kobak & Berens 2019).

❦

Dataset visualization is a core activity of contemporary science. While it has some downsides, the visualizations make it possible to get a glimpse into the structure and relationships of datasets which would otherwise remain hidden. For exmple, even the US government under the Obama administration used visualization techniques to find structure in population genomic data (All of Us 2024). While the methods are not always easy to wield, they still provide useful insight that is otherwise hard to obtain as the 2D visualization often informs the research questions themselves.

1.2 Related work

There is an abundance of visualization techniques. However, a lineage can be traced from data visualization techniques that have been invented before modern computers to contemporary neighbor embedding methods which run on GPUs.

History of dataset visualization methods

Pearson (1901) and Hotelling (1933) independently introduced principal component analysis (PCA). PCA creates a linear projection from the original space along the principal axes that capture the most variance. In the special case that only the first two principal components are used, you create a scatter plot of a dataset, by mapping every datum from the original data space to 2D. Note that this is not a visualization technique per se. Hence, it would take several decades until this application was actually used in the field. Jolicoeur & Mosimann (1960) created the first PCA plot in this fashion by analyzing the painted turtle. Before this, Rao (1948) created a plot based on linear discriminant analysis (LDA), which resulted in a similar visualization technique. The underlying data dates back to an anthropometric survey carried out by Mahalanobis, Majumdar, & Rao (1949) in the late colonial British empire. These measurements of parts of the Indian population were translated into an abstract concept of similarity by LDA.

Calyampudi Radhakrishna Rao (1948). "The Utilization of Multiple Measurements in Problems of Biological Classification". In: *Journal of the Royal Statistical Society* 10.2, pp. 159–193

There does not seem to be any dataset visualization in this fashion that was carried out by hand. Even the very first plot by Rao (1948) was aided by Mallock's machine, based in Cambridge (Mallock 1933).

Since both LDA and PCA are linear methods, the prior assumption is that the relationship between the high- and low-dimensional data is linear, too. This severely limits the modeling capacity of both methods.

Going forward a bit, Torgerson (1952) introduced multidimensional scaling (MDS). It is a means to take a dataset and project all distances between each pair of data points down to a lower dimension. Initially, this was not used for visualization, hence this was not 2D. The first application of MDS for visualization was by Sammon (1969), who essentially applies MDS in the case that the number of

output dimensions are 2 or 3. MDS is the first method that aims to preserve distances between points and hence circumvents the linear limitation that plagues PCA and LDA. Instead of preserving the data itself, the distances between all pairs of points are preserved when mapping from the original data space to the low-dimensional space. As such, it attempts to preserve the relationships between points instead of the properties of the points directly. Preserving the distance itself proved to be a valuable goal and it still underlies modern visualization methods.

One downside to MDS is the fact that it scales quadratically with the number of data points. Another drawback is that the distances between points in high-dimensional space follow a different distribution than the low dimensions; hence it is not clear that preserving all distances is a meaningful endeavour. As we will see, both of these shortcomings have been addressed by other researchers later on.

Another avenue for visualization is non-negative matrix factorization (NMF). First proposed and used under a different name — self modeling curve resolution (Lawton & Sylvestre 1971) — it was used for analysis within the chemistry domain. The term NMF surfaced under its nowadays recognizable name in Paatero et al. (1991) and through analysis and extensions by Lee & Seung (1999) was made feasible to run for data analysis. Notably, this technique was widely used in recommender systems, for example by the winning team of the Netflix prize (Koren, Bell, & Volinsky 2009). NMF reconstructs two approximate matrices that, when multiplied, will result in the original matrix. In the case that the feature matrices have only two dimensions, the representation can be visualized straightforwardly.

So far, all of the methods presented in this section optimize an error function. For example, PCA maximizes the variance of the dataset when projecting from the original space to 2D. Similarly, MDS minimizes the change in distances when projecting the points to 2D.

In contrast to that, there are two approaches that are quite dissimilar to previous work as they do not directly optimize an error function. One approach, inspired by neuroscience, is called self-organizing maps (SOM), which uses competitive learning instead of gradient descent (Kohonen 1982). The other is the implication of the Lemma by Johnson & Lindenstrauss (1984). The method is known as random projection (RP), which takes the original data and projects it to a new space with a simple matrix multiplication of a random matrix. The lemma guarantees that the distances between all points will be preserved up to a factor of $1 + \epsilon$, where ϵ depends on the output dimension.

Graph drawing There is a striking similarity between preserving the distances between points and graph drawing, where edges are preserved instead of distances. Interpreting the edges as a distance will actually reveal very similar results — in fact this is a central aspect discussed in Böhm et al. (2025) — and hence we provide a small overview of graph drawing methods.

The first graph drawing method was by Tutte (1963), showing that every polyhedral graph[5] can be drawn in the plane. While this only applies to a specific type of graph, it has been extended to general graphs (Eades 1984). While the planarity cannot be guaranteed anymore, drawing graphs in the plane still proved to be useful. Thus, further research has been invested into method development, with Kamada & Kawai (1989) as well as Fruchterman & Reingold (1991) publishing two approaches that have become popular and are widely implemented (for example by Hagberg, Swart, & S Chult 2008).

Jan Niklas Böhm, Marius Keute, Alica Guzmán, Sebastian Damrich, Andrew Draganov, & Dmitry Kobak (Jan. 2025). "Node Embeddings via Neighbor Embeddings". In: *Under review*

5. A polyhedral graph is a graph which stays a connected component as long as you remove at most three nodes.

The works by Noack (2009) expanded the field of graph drawing by formulating a family of graph layout algorithms that generalize the algorithm put forward by Fruchterman & Reingold (1991). One specific instance of this family has been taken and implemented efficiently (Jacomy et al. 2014) using the Barnes–Hut (BH) approximation scheme (Barnes & Hut 1986), named ForceAtlas2 (FA2).

Modern dimension reduction At the end of the century, the notion of using geodesic distances was developed (Tenenbaum 1997), called Isometric feature Mapping (ISOMAP). Geodesic distance does not take the direct, Euclidean distance into account, but instead only travels via neighboring nodes that are close-by. This seemingly innocuous change removed the constraint of the input space being fully Euclidean, but instead allowed it to model a Riemannian manifold. While the geodesic distance allows for flexible modeling, it also has the same drawbacks as MDS. By needing to minimize a loss function over a dense distance matrix, both methods require a lot of computation. As the datasets grew over the years, this problem became more pronounced, as improvements in computations were not able to keep up with the quadratic complexity $\mathcal{O}(n^2)$ that is required to solve both ISOMAP and MDS.

A trick that brought partial relief to this is called Locally Linear Embedding (LLE), introduced by Roweis & Saul (2000). The idea behind LLE is to only consider the distances within a local neighborhood, instead of trying to optimize all pairwise distances. This was the first example of a NE algorithm. Another, seemingly unrelated, approach was to formulate a similar problem as LLE in terms of a generalized eigenproblem, which resulted in the algorithm called Laplacian Eigenmaps (LE). This eigenproblem can then be solved with a traditional solver (Belkin & Niyogi 2002, 2003) and the smallest non-trivial eigenvectors correspond to the low-dimensional representation. LE was connected to the general NE paradigm by Linderman & Steinerberger (2019).

Stochastic Neighborhood Embedding (SNE) was developed as an extension to LLE, in order to model the local density adaptively. This allows for dense regions in the ambient space to spread out in the low-dimensional representation. To allow for this, the neighborship relation was determined with a Gaussian kernel, with a width that was optimized for each individual point, hence adapting to the local density. The problem of modeling the similarities by a Gaussian in both the input as well as the output space lead to the crowding problem, where the majority of points would end up in the center of the embedding. Van der Maaten & Hinton (2008) proposed replacing the output kernel by a t-kernel with a degree of freedom equal to one (this is equivalent to a Cauchy kernel). This kernel has a heavier tail towards the end of the distribution, which allows the data points in the output space to spread out more, resulting in a better visualization. The method has since persisted as state-of-the-art, though there have been some improvements in the runtime complexity as well as investigations into which parameters play a significant role (Belkina et al. 2019; Kobak & Berens 2019; Kobak & Linderman 2021).

Laurens van der Maaten & Geoffrey Everest Hinton (Nov. 2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605

McInnes, Healy, & Melville (2018) introduced a new algorithm, named UMAP, which has an approach that appears similar to t-SNE, but it is motivated differently (by using fuzzy set terminology and slightly different kernels). However, Damrich & Hamprecht (2021) and Böhm, Berens, & Kobak (2022) have shown that UMAP is intimately related to t-SNE and not as different as it was made out to be.

Jan Niklas Böhm, Philipp Berens, & Dmitry Kobak (Mar. 2022). "Attraction-Repulsion Spectrum in Neighbor Embeddings". In: *Journal of Machine Learning Research* 23.95, pp. 1–32

History of contrastive learning

Compared to the previous section, the history of contrastive learning (CL) is comparatively short. CL has been put forward as a method that learns a representation of image datasets (Chen et al. 2020). It is not common to interpret CL as a visualization method, this has been the focus of the research presented in Böhm, Berens, & Kobak (2023), see also Chapter 3.

Ting Chen, Simon Kornblith, Mohammad Norouzi, & Geoffrey Everest Hinton (2020). "A simple framework for contrastive learning of visual representations". In: *International Conference on Machine Learning*, pp. 1597–1607

The theory behind contrastive learning goes back to Hadsell, Chopra, & LeCun (2006), where it was identified that a representation of data points can be learned by contrasting the data distribution to a noise distribution. Essentially, this turns the learning problem into a pseudo-supervised problem where the goal is to distinguish data from the real distribution with data coming from the noise distribution. This method has been developed further by Gutmann & Hyvärinen (2012), extending it to un-normalized distributions. Information noise-contrastive estimation (InfoNCE) was introduced by van den Oord, Li, & Vinyals (2018) as an extension to noise-contrastive estimation (NCE). It reformulates the contrastive objective as a multi-class supervised problem instead of only a binary one. Notably, InfoNCE is the part of the loss function in simple contrastive learning representation (SimCLR), the method developed by Chen et al. (2020).

SimCLR being self-supervised made it widely applicable across a diverse range of domains. With this, investigations into how it works on a theoretical level were also carried out, for example by Wang & Isola (2020) or Zimmermann et al. (2021). Both works investigate the relationship between the real and the noise distribution in the CL framework and how it affects the resulting representation.

In a similar direction, we investigated the connection between contrastive learnings and neighbor embeddings (Damrich et al. 2023). We showed that they have similarities and that CL can be expressed in the NE framework and vice versa. This shows how intimately related the two approaches are and informs a range of loss functions that work for both approaches.

Sebastian Damrich, Jan Niklas Böhm, Fred A. Hamprecht, & Dmitry Kobak (July 2023). "From t-SNE to UMAP with contrastive learning". In: *International Conference on Learning Representations*

With CL being a comparatively young subdiscipline, there is a shorter lineage of works. Nevertheless, in the years since it's first popular introduction by Chen et al. (2020) it has steadily expanded. While SimCLR (Chen et al. 2020) uses the InfoNCE loss to constrast data points from each other, there have been numerous extensions that instead contrast the dimensions across the batch (Caron et al. 2020; Zbontar et al. 2021). Interestingly, Garrido et al. (2023) have shown that those two approaches — called sample-contrastive and dimension-contrastive — optimize a similar loss. One gap in the works connecting sample-contrastive to dimension-contrastive is how they can be tied together exactly. Garrido et al. (2023) only show that there exists a duality, but not how it is constructed. As such, the exact connection to NEs also exists, but as it hinges on the results between dimension- and sample-contrastive self-supervised learning (SSL), the construction reimains elusive.

Other developments

Over the years there have been a plethora of dimension reduction methods that extend or improve upon t-SNE or UMAP. Most notably are the optimization techniques for t-SNE that employ the BH approximation (Yang, Peltonen, & Kaski 2013; van der Maaten 2014) and the interpolation based on fast multipole method (FMM) by Rokhlin (1985), introduced to the t-SNE optimization method by Linderman et

al. (2019). The optimization brings the costly optimization on the order of $\mathcal{O}(n^2)$ down to $\mathcal{O}(n \log n)$ and $\mathcal{O}(n)$, respectively.

One other important avenue is creating a parametric embedding method that allows mapping points down to 2D via a parametric function that is learned. This has been implemented for both t-SNE (van der Maaten 2009) and UMAP (Sainburg, McInnes, & Gentner 2021).

Attraction-Repulsion Spectrum in Neighbor Embeddings

2

Jan Niklas Böhm, Philipp Berens, & Dmitry Kobak (Mar. 2022). "Attraction-Repulsion Spectrum in Neighbor Embeddings". In: *Journal of Machine Learning Research* 23.95, pp. 1–32

Before discussing the publication itself, there will be some brief introduction to the notation used throughout this and the following chapters. As already mentioned on Page 1, the dataset visualization function maps the dataset $\mathbb{R}^{n\times d}$ from $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times 2}$. A NE algorithm is an instance of dataset visualization and the main focus of this work. NE have the additional, intermediate step, where they create the neighbor matrix $\mathbb{R}^{n\times n}$. We take a slightly adapted version from Böhm (2020):

Definition 1 *Neighbor embedding dimensionality reduction is a mapping from a high-dimensional space to a low-dimensional space* $f \circ g : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times 2}$, *with* $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times n}$ *creating a neighbor matrix and* $g : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times 2}$ *operating on that matrix.*

This definition lets us decompose the visualization into two steps, as we first create the neighbor matrix and only then we create the actual visualization. The research of my PhD consisted of analyzing both parts of the visualization algorithms. The publication discussed in this chapter focuses on the latter part of Definition 1, answering the following research question:

How do the different popular NE methods relate to each other?

2.1 Summarizing layout algorithms along a spectrum

The results in Böhm, Berens, & Kobak (2022) connect different layout algorithms. Most notably, both UMAP and t-SNE are related and t-SNE can be used to generate a layout that mimics the one of UMAP.

The two algorithms are motivated differently and for a long time, the exact relationship was unknown, with claims that UMAP was superior due to for example the loss function (Oskolkov 2019) or the initialization (Becht et al. 2019). Over the years, there were investigations into best practices for using t-SNE (Belkina et al. 2019; Kobak & Linderman 2021), but there was no comprehensive treatment of how those two algorithms relate to one another.

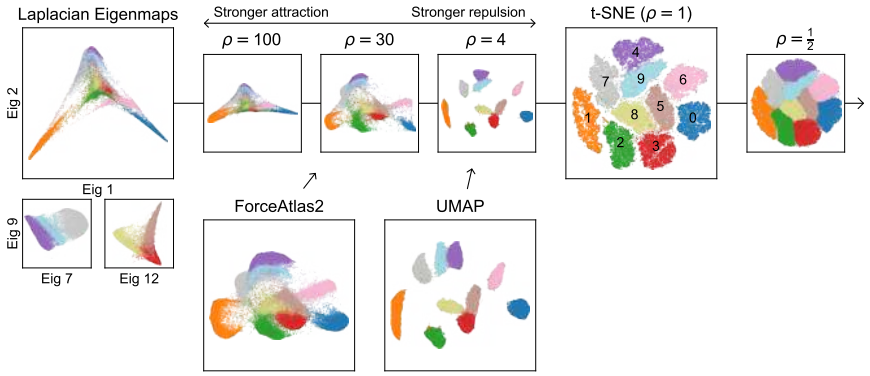

Figure 2.1: The attraction-repulsion spectrum. Figure 1 from Böhm, Berens, & Kobak (2022).

for t-SNE we have the following loss function:

$$\begin{aligned}\mathcal{L}_{\text{t-SNE}} = \mathrm{KL}(P \parallel Q) &= \mathbb{E}_P\left[\log(P \oslash Q)\right] \\ &= \mathbb{E}_P\left[\log P\right] - \mathbb{E}_P\left[\log Q\right] \\ &= \mathbb{E}_P\left[\log P\right] - \mathbb{E}_P\left[\log W\right] + \log Z\,, \end{aligned} \tag{2.1}$$

with $\oslash$ being the Hadamard division, the elementwise division. Here, P is the probability matrix for neighborship in high-dimensional space and $Q = W/Z$ for the low-dimensional neighbors. When taking the derivative with respect to Q, the first part $\mathbb{E}_P\left[\log P\right]$ will vanish. See Böhm (2020) for a more in-depth treatment of the precise definitions.

Jan Niklas Böhm (Sept. 1, 2020). "Dimensionality Reduction with Neighbor Embeddings". MA thesis. University of Tübingen

In comparison, the loss function for UMAP — after some simplification — reads (again, see Böhm 2020, for more details) :

$$\mathcal{L}_{\text{UMAP}} = \mathbb{E}_{P'}\left[\log(P \oslash W)\right] \tag{2.2}$$

The principal difference is that W is not normalized by Z, unlike in the t-SNE loss in Equation (2.1). By investigating the remarkable difference between the loss functions, while both produce workable visualizations, we noticed that the way UMAP optimizes its loss leads to a different expectation as well, here denoted by $\mathbb{E}_{P'}$. While notationally small, this is the connecting piece between the two approaches, with other smaller details not having as big of an influence.

Follow-up work was done by Damrich & Hamprecht (2021), where the authors arrived at a similar conclusion by way of analyzing the gradient.

2.2 FORCEATLAS2 ON THE ATTRACTION-REPULSION SPECTRUM

Another connection that was established in Böhm, Berens, & Kobak (2022) is connecting FA2 to the attraction-repulsion spectrum, as schematically shown in Figure 2.1. The reason for this was that fact that the attractive forces in FA2 do not decay with the distance. The neighborhood relationship in P (cf. Equations (2.2) and (2.1)) will thus always exert an attractive force, irrespective of the distance in the low-dimensional embedding. While the loss function for FA2 looks differently,

$$\mathcal{L}_{\text{FA2}} = \sum_{i,j}\left(v_{ij}d_{ij} + \frac{(h_i+1)(h_j+1)}{d_{ij}}\right)\|y_i - y_j\|^2\,, \tag{2.3}$$

it actually resembles the other ones quite well. Again, the interested reader is referred to Böhm (2020) and Böhm, Berens, & Kobak (2022). Interestingly, the concept of force-directed layouts can be generalized (Noack 2009), which then encompasses for example the Fruchterman–Reingold (FR) layout (Fruchterman & Reingold 1991). Whether the entire family of force-directed graph layout algorithms can be mapped onto this spectrum remains an open question.

Graph layouts and neighbor embeddings Graph layout algorithms are designed for visualizing arbitrary graphs, thus they can visualize k-nearest neighborhood (kNN) graphs, too. One question that came up during the research is whether this would work the other way around, too: can t-SNE be used to visualize arbitrary graphs? For an answer to this question, see Chapter 4.

Unsupervised Visualization of Image Datasets Using Contrastive Learning

Jan Niklas Böhm, Philipp Berens, & Dmitry Kobak (July 2023). "Unsupervised visualization of image datasets using contrastive learning". In: *International Conference on Learning Representations*

3

After the previous chapter concerned itself with the dimension reduction part of the visualization process, we now try to answer a different question: How can we visualize a dataset, without first constructing the full neighbor graph? As a reminder, in Chapter 2, we introduced the NE visualization definition in Definition 1 as mapping from $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times 2}$. We now explore the setting where we can only sample from a neighbor graph, but we cannot fully observe it. This happens when the data is not locally Euclidean, so as soon as we deal with a non-Riemannian manifold.

3.1 Visualizing image datasets

Visualizing data that does not lie on a manifold has an inherent difficulty. The assumption that NE algorithms make is that the local Euclidean distance is meaningful, which gives rise to the Riemannian manifold (do Carmo 1992) that the data must lie in. When this is not the case anymore, then the notion of a neighbor is not correct anymore and NE algorithms, like t-SNE, break down.

One example that are not Euclidean are image datasets. The Euclidean distance between pixel values has almost nothing to do with the semantic meaning of an image. Instead we need to find a way to designate neighbors in the image domain, where the image neighbors are similar in a way that our visual system perceives it.

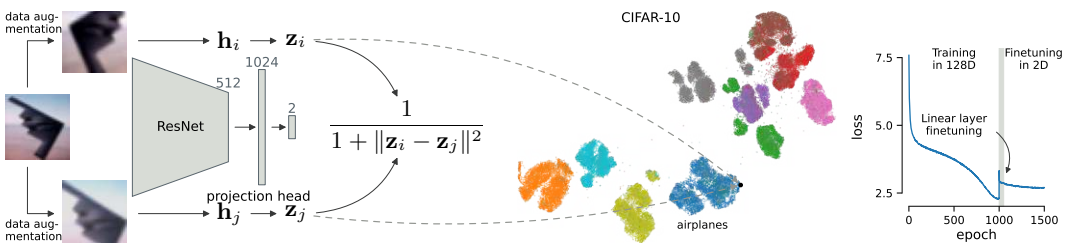

Figure 3.1: Figure 1 from Böhm, Berens, & Kobak (2023).

The method we present *generates* neighboring images. As data augmentations are plentiful in the image domain, we randomly augment an image twice to get two new images that should still be semantically similar. Figure 3.1 gives an overview on how this works, the resulting two images are then passed through a neural network and a new loss function is applied. This loss function is derived from two parts: (1) the t-SNE loss function provides the similarity kernel and (2) the SimCLR loss in the form of InfoNCE is used to optimize the neural network. This results in the method named t-Distributed Similarity-based Contrastive Neighborhood Embedding (t-simCNE), a parametric visualization method for image datasets.

3.2 Neighbor embeddings and contrastive learning

Sebastian Damrich, Jan Niklas Böhm, Fred A. Hamprecht, & Dmitry Kobak (July 2023). "From t-SNE to UMAP with contrastive learning". In: *International Conference on Learning Representations*

The work by Damrich et al. (2023) highlighted the connection between NE and CL. Through this, we can express both setups in the same domain. For example, this means that the InfoNCE loss function can be applied to t-SNE and the optimization will still work. Through that, we also get the option of having a parametric t-SNE embedding.

The idea is that the contrastive loss can be reinterpreted as sampling an edge from a graph, instead of a data point directly, as done in Figure 3.1. Both nodes connected to this edge are then the "augmented" views of the original data point. With this interpretation we can then carry out the optimization with non-parametric embeddings in the NE domain.

There is one caveat, however. Since contrastive learning augments the image itself, it creates an entirely new data point. Under the NE framework, the data points are all fixed and identifiable. This means that the latter can be optimized non-parametrically, whereas the former one needs a parametric mapping in order to learn something meaningful.

Conceptually, this means that the graph that is being optimized over in CL is infinite, and we basically only sample edges from this graph once. On the other hand, the NE graph is fixed, with identifiable nodes. However, since NEs can still be optimized parametrically, they still fit into the same general framework.

3.3 Contrastive visualization

With the connection between NE and CL established, we can now explain how t-simCNE works. It is an adaptation of the contrastive learning technique SimCLR (Chen et al. 2020). SimCLR learns a self-supervised representation of an image dataset by optimizing the similarity between two augmented images from the same source (cf. Figure 3.1). The similarity function employed by SimCLR in unfortunately not suitable for visualization, as they take the cosine similarity in the output space. For 2D this would be a circle, as everything is constrained to lie on a $(d-1)$-dimensional sphere. As such, we adapt the loss function from

$$\mathcal{L}_{\text{SimCLR}}(i,j) = -\log \frac{\exp\big(\operatorname{sim}(z_i,z_j)/\tau\big)}{\sum_{k\neq i}^{2b}\exp\big(\operatorname{sim}(z_i,z_k)/\tau\big)} \tag{3.1}$$

$$= -\operatorname{sim}(z_i,z_j)/\tau + \log\sum_{k\neq i}^{2b}\exp\big(\operatorname{sim}(z_i,z_k)/\tau\big)\,. \tag{3.2}$$

to a more suitable form

$$\mathcal{L}_{\text{t-simCNE}}(i,j) = -\log \frac{1/(1+d_{ij}^2)}{\sum_{k\neq i}^{2b} 1/(1+d_{ik}^2)} \tag{3.3}$$

$$= -\log \frac{1}{1+d_{ij}^2} + \log \sum_{k\neq i}^{2b} \frac{1}{1+d_{ik}^2}\,. \tag{3.4}$$

The difference between $\mathcal{L}_{\text{SimCLR}}$ in Equation (3.4) and $\mathcal{L}_{\text{t-simCNE}}$ in Equation (3.4) is that we replace the SimCLR similarity function exp ∘ sim with a t-SNE inspired similarity kernel in the form of $(1+d_{ij}^2)^{-1}$, where the distance d_{ij} is the squared Euclidean distance $\|z_i - z_j\|^2$. This means that instead of taking the cosine metric between two points (which is how sim is defined by Chen et al. 2020), we instead take the Euclidean metric in the output space.

Simply changing the loss function leads to suboptimal results, which we were able to significantly improve upon with a custom training regime in order to learn features in a low-dimensional output. For the details, the interested reader is referred to Böhm, Berens, & Kobak (2023).

Node Embeddings via Neighbor Embeddings

Jan Niklas Böhm, Marius Keute, Alica Guzmán, Sebastian Damrich, Andrew Draganov, & Dmitry Kobak (Jan. 2025). "Node Embeddings via Neighbor Embeddings". In: *Under review*

4

Tying together Definition 1, formulated in Chapter 2, and Chapter 3 we now ask two questions: (1) how do NE methods visualize arbitrary graphs and (2) can contrastive methods, as introduced in Chapter 3, compete with other node embedding techniques?

The mapping $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times 2}$ is now essentially cut short. We are given a fixed neighbor matrix $\mathbb{R}^{n\times n}$ in the form of a graph and want to map this to 2D. Furthermore, we look at the case when we have a high-dimensional representation, and apply the SimCLR loss directly to a graph dataset. How does the resulting representation compare to other approaches that embed graphs?

4.1 Graph Visualization and Representation

There are plenty of works that investigate on how to amend t-SNE in order to make it work for general graph visualization (Kruiger et al. 2017; Pitsianis et al. 2019; Zhu et al. 2020a; Zhong et al. 2023). For the most part, the approaches deviate significantly from the original t-SNE approach of visualizing data.

We discovered that t-SNE can readily visualize graphs and improve upon state-of-the-art results in 2D visualization, while having a conceptually simpler approach.

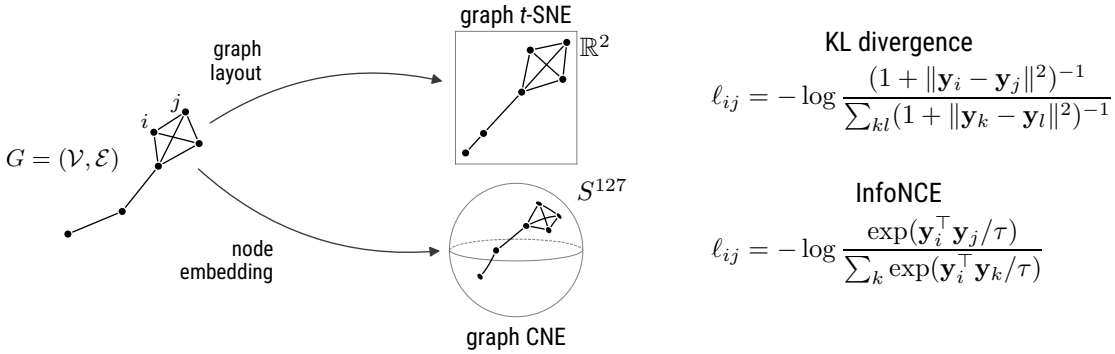

Figure 4.1: Two ways of embedding a graph into either $\mathbb{R}^2$ or S^{127}. Figure 1 from Böhm et al. (2025).

Combining this with the results in Damrich et al. (2023), this gives rise to also applying CL to graphs directly. This investigation revealed that nonparametric node embeddings can also be optimized with the InfoNCE loss.

Doing so yields impressive results with retaining the neighborhood structure much better than other approaches. The work presented in Böhm et al. (2025) spells out two critical improvements: (1) t-SNE can be applied to almost arbitrary graphs and (2) that graph contrastive algorithms are suitable node embeddings, outperforming most other approaches.

4.2 GRAPH CONTRASTIVE LEARNING REQUIRES LOW TEMPERATURE

CL has a hyperparameter for the so-called temperature τ. It governs how strongly the negatives will be penalized and is usually set to a constant scalar throughout (Chen et al. 2020). When applying CL to graph data, we noticed that the neighbor recall strongly correlates to the temperature used. When the temperature is high, for example the default value of $\tau = 0.5$, then the recall peaks within the first epoch and the local neighborhood structure cannot be retained. On the other hand, when decreasing the temperature by an order of magnitude to $\tau = 0.05$, this problem is largely resolved.

Fortunately, it is not required to tune the temperature parameter by hand. Making it a learnable parameter also leads to the temperature τ converging to similar values and thus actually removes the burden of choosing a hyperparameter from the process.

4.3 NEIGHBOR EMBEDDING METHODS CREATE MEANINGFUL NODE EMBEDDINGS

The beauty of the approach in Böhm et al. (2025) is that the optimization procedure leverages the lessons already learned from both t-SNE and SimCLR. As such, the embedding quality exceeds all other approaches in terms of neighbor recall and has similar results in both linear and kNN classification accuracy.

All of these comparisons were made by training nonparametric models. It would be interesting to extend the analysis to parametric models, which are more widely used throughout the graph neural networks (GNN) literature.

All in all, it is encouraging to see that applying the lessons learned from NE and CL to graph data improves upon the state-of-the-art results.

Discussion

5

During my PhD I researched dataset visualization methods. The findings include some novel connections between different algorithms (Böhm, Berens, & Kobak 2022) and loss functions (Damrich et al. 2023), novel visualization methods for image datasets (Böhm, Berens, & Kobak 2023) as well as a connection to a new domain of visualization in the form of graph data (Böhm et al. 2025).

Furthermore, we have looked into inherent behavior in self-supervised machine learning methods (Draganov et al. 2025) and applied the methods to neural time-series data (Schmors et al. 2025) as well as medical histology and microscopy data (Nwabufo et al. 2024).

The connecting theme of the research is learning representations of data without a supervisory signal, with a specific focus on NE visualization methods. Where initially the goal was to understand various classical methods, we were able to find a connection that allows expressing various algorithms along the attraction-repulsion spectrum (Böhm, Berens, & Kobak 2022). Going forward, the knowledge about neighbor embedding methods then resulted in the creation of a new method that can visualize images, which is achieved with a generalization and combination of NEs with CL (Böhm, Berens, & Kobak 2023).

In the end, as hinted at in Chapter 3, NE and CL are optimizing a similar goal. The major difference between them is the sampling strategy. Where NE assumes a static, unchanging neighbor graph, CL works by sampling data from an augmentation graph, which is less well understood. Further implications of this is that where NE has identifiable data points, self-supervised CL only ever trains on augmentations thereof. This implies that you cannot train the latter non-parametrically, as the neighborship relation is ephemeral. Essentially, NE can just iterate over the (nonzero) elements in the adjacency matrix $\mathbb{R}^{n \times n}$, while CL indexes the data points in $\mathbb{R}^{n \times d}$ and augments each data point twice, thus sampling from a distribution conditioned on the indexed data point. The hope is that out of the n distributions in the dataset, there is enough overlap in the probability mass[1] so that semantically similar points will force similar actviations in the neural network through the optimization.

1. Since the support is in $\mathbb{R}^{n \times d}$ it trivially overlaps.

By its nature, the augmentation graph is harder to analyze than the simpler neighbor graph. For the latter, there are theorems that aid in its analysis. As an example: LE will properly decompose the neighbor graph into disconnected graphs (von Luxburg, Belkin, & Bousquet 2008). Due to the stochastic nature of the augmentation graph, it is not possible to make similar statements without further theoretical work.

Going the other way around, combining the neighbor graph with CL is much easier. Concurrently to the publication about t-SimCNE (Chapter 3 & Böhm, Berens, & Kobak 2023), Damrich et al. (2023) established the connection between NE and CL. Equipped with this, we applied methods from both domains directly to graph

data and found that we could match or beat the performance of state-of-the-art methods (Böhm et al. 2025). In terms of neighbor preservation our approach performs best, which means that the local relationships are distorted the least; an important property, especially when considering the visualization aspect of it. In the preparation of the work on node and neighbor embeddings (Böhm et al. 2025) there was ample cross-fertilization between this work and the work by Draganov et al. (2025), which aided the understanding of the role of vector norms and how they behave during training in a high-dimensional space.

The two works by Nwabufo et al. (2024) and Schmors et al. (2025) were applying the groundwork laid in Böhm, Berens, & Kobak (2023) to novel data domains. This required adaptations to the augmentations to make them more suitable to their respective modalities.

The former focused on medical microscopy images. Such images are markedly different from natural images in that they exhibit more symmetries that should be exploited in order to improve the learned representation. Since there is no real orientation in the images, it makes sense to rotate and reflect the images such that the variance of the images during training increases. We showed that this improves the representation and visual quality for medical microscopy images.

In the latter (Schmors et al. 2025), we exploited the nature of the data to automatically sample contrastive pairs without the need of an extra augmentation. By noticing that neuroscientific trials have an inherent struture with repeated experiments, the data augmentations could be redefined such that this structure can be used for the contrastive augmentation pipeline. By sampling trials before taking the mean, it is possible to average out the inter-trial noise that was recorded during the experiment. Interestingly, this works better in the face of biological noise compared to the previously used techniques of applying NE algorithms or other contrastive learning techniques (Vishnubhotla et al. 2023) to time-series data.

5.1 FUTURE WORK

All in all, the methods presented herein show improvements in the visualization domain both methodical as well as domain-based. In the future I hope that both avenues will be further explored. There is some preliminary work already being done in those directions with applications to fine-art as well as further exploration into time-series data for disease progression based on medical images.[2]

2. Both are still actively being worked on.

One thing that greatly impacts usability is the time it takes to train a model. While the first publication of t-SimCNE reported roughly 18 hours for the training of the neural network (Böhm, Berens, & Kobak 2023, Table 1), this has been improved and similar results can now be achieved within only six hours (using the current version of the code published on GitHub[3]). Despite the improvements, this is still far off from classical NE methods that can create an embedding for a dataset of similar size within minutes, not hours. One of the bottlenecks for CL-based visualization is the data augmentation, which can be quite costly. It would be interesting to see if it is possible to attain a significant speedup by augmenting the dataset a number of times before the training and then only randomly sample from those candidates during the training. With that, the visualization quality hopefully would not deteriorate and the optimization could be significantly sped up as the expensive data augmentations would have been precomputed.

3. https://github.com/berenslab/t-simcne

Another interesting avenue would be to improve the runtime by implementing similar optimization strategies as it has been done for t-SNE in the form of BH (Barnes & Hut 1986) or FMM (Rokhlin 1985) optimization. Since a GPU operates differently than a classical CPU, it is not clear if it will speed up training significantly. It would also make the implementation more complicated, since the gradient optimization method is never explicitly formulated when using machine learning frameworks such as PyTorch (Paszke et al. 2019). That being said, the current implementations use simple subsampling in the form of only considering the batch for the loss calculation. This represents a deviation from how the loss is often formulated in theory, but works out in practice. Some more substantiated research in that direction might either improve current training regimes, or explain why the current approximations are sufficient.

metric mystery

One open question is why the cosine distance metric works better in high-dimensional space than the Euclidean metric. Resolving this would hopefully lead to a better understanding of how exactly the two metrics relate to each other. This effect has been noted and studied in other settings already, but there is still no general conclusion as to why the Euclidean metric performs worse in high-dimensional space (Aggarwal, Hinneburg, & Keim 2001; Domingos 2012; Mirkes, Allohibi, & Gorban 2020). As pointed out by Böhm, Berens, & Kobak (2023), the two metrics are equivalent up to a constant under normalization to the hypersphere. What this regularization means for the resulting representation is not clearly understood, but there are a few hypothesis, that try to explain it (Draganov et al. 2025) or use the inherent behavior to improve the overall representation (Kirchhof et al. 2022).

multi-modality

Another interesting direction for future research would be to create multi-modal visualization methods. This seems particularly promising to me, as multi-modality increased in relevance over the years, with for example Contrastive Language-Image Pre-training (CLIP) by Radford et al. (2021). The connection between the static neighbor graph as it is used in the NE framework and the dynamic augmentation graph as used within CL will probably become important, as the embedding would now reach across modalities. Thus, embedding samples from both domains will be challenging, since the principal graph edges would reach across modalities, instead of connecting samples from a single modality. When embedding text, another challenge is how to embed a variable-length sequence into a fixed-dimension vector space. Current approaches aggregate over the entire sequence or designate a specific token for the global text representation, but there may be a better reduction for the specific case of embedding 2D data points.

uncertainty

Yet another avenue concerns itself with the uncertainty of the embeddings. The location of a point in the embedding space could be annotated by the uncertainty that the model has associated to the datum, thus showing regions of high-certainty as well as potential sources for misclassification. Some work in the direction of uncertainties for CL has been done by Kirchhof et al. (2022) and Kirchhof, Kasneci, & Oh (2023). Interestingly, the norms of the embedding relate to the certainty of the model, which is a theme that has been picked up by Draganov et al. (2025), where the reason for growing norms in the CL training was researched. All of the aforementioned publications only deal with high-dimensional output, where the similarity function is based on the cosine metric. Translating this to both

the Euclidean metric as well as the 2D output would be an interesting topic for exploration. This probabilistic take on the embedding could also then be applied to the more classical NE methods, and thus retrofit an improvement. In general, Kirchhof et al. (2023) also showed that including uncertainties improves the model performance overall, which is why the inclusion of probabilistic embeddings should improve the visual quality as well.

model architecture

Further extensions to the work presented would include revising the architecture as well as the training regime. For example t-SimCNE was only trained with a Residual Network 18 (He et al. 2016), but other network architectures should either improve the efficiency (by using for example EfficientNet or MobileNet v2, Sandler et al. 2018; Tan & Le 2019) or the visual quality with more modern architectures such as vision transformers (Dosovitskiy et al. 2021).

pre-training

Especially the latter architecture runs counter to the usual requirement for visualizations to be generated quickly — usually training large transformer architectures takes large amounts of computations, which translates into long training times. To combat this, it would be pertinent to investigate using pre-trained models that will then "only" need to be finetuned to create a visualization quickly. This is not as simple as it is for the other applications, since we not only need to adapt to the data at hand, but also to the much more limited 2D output dimension. When pre-training in a supervised setting, the model inherents the bias of the labels and might fail to learn more subtle structure in the data. As we want to learn structure that is inherent to the data, this only leaves other approaches, that have not been trained with a supervisory signal. One first step into that direction would be to use other large-scale self-supervised models, which should not suffer from bias to labels (but have other drawbacks, for example dimensional collapse, see below).

data augmentation

For self-supervised learning, the data augmentations play a crucial role and depend on the learning problem that is optimized (Steiner et al. 2022). Investigating how this translates to the 2D output case would be an interesting field of study, where so far the same set of augmentations have shown promising results. Nevertheless, since the data augmentation plays a central role in the loss, there have been publications dedicated to a single parameter for a single augmentation.

geometry

Geometric deep learning is a technique that encodes the underlying geometry into the network architecture itself, and can thus be used to encode equivariances or invariances (Cohen & Welling 2016; Cohen et al. 2018). This concept maps quite naturally to CL, where the aim is to learn invariances in the data through data augmentations. Applying and combining those two techniques should improve the efficiency of CL, which would be in line with other future work pointed out in this section. Furthermore, it would put CL on a solid theoretical foundation. Note that not all augmentations that are employed in the usual self-supervised setting can be mapped to a geometric transformation, so both approaches solve a different goal, and instead compliment each other nicely.

dimensional collapse

There is an interesting phenomenon within the self-supervised literature called dimensional collapse (Jing et al. 2022; Shi et al. 2023). While the output dimension is usually high-dimensional, the model does not utilize the full space of it. By employing some mitigation strategies, it has been noted that the span of the output dimension correlates with the downstream accuracy positively. It is still not fully resolved why this collapse occurs or how to prevent it. In the 2D visualization the output dimension is already too restricted for the dimensional collapse to be a problem. But we have noted that the Euclidean metric we employ for t-SimCNE has a more pronounced dimensional collapse than the cosine metric does (Böhm, Berens,

& Kobak 2023, Figure A.6). This evidence gives a hint that the dimensional collapse could be related to the worse performance we observe in the high-dimensional space when comparing t-simCNE to SimCLR. One could also investigate how the dimensional collapse relates to the metric used in the training loss.

temperature

When preparing the manuscript for the most recent submission (Böhm et al. 2025), we noticed that the temperature plays an important role in the training dynamics of CL (see also Section 4.2). Why the parameter tends to differ so strongly between CL applied to graph data versus image data is not yet fully clear and would be interesting to further explore.

parameters

Last but not least, revisiting the parameter choices made in Böhm, Berens, & Kobak (2023) could prove to be beneficial. We noticed the dependency on the temperature parameter, as outlined in the previous paragraph, and there exists an interesting interplay between the weight decay parameter and the speed of convergence in self-supervised learning (Draganov, Vadgama, & Bekkers 2024; Draganov et al. 2025). Both examples show that there may be more to uncover. One simplification in this direction has already been carried out: we managed to move from a three-stage optimization procedure (cf. Figure 3.1) to a simple end-to-end optimization by annealing the dimensions themselves instead of changing the model.

❦

Concluding this section, there are many different directions left to explore. As often the case, the more a topic is being researched, the more questions arise. Especially with the explosion of machine learning and the wide applications thereof in every day life, the discipline has attracted a lot of researchers. As such, there are many more questions that await an answer.

5.2 CONCLUSION

In summary, this thesis presents recent advances in the domain of dataset visualization, ranging from understanding visualization techniques (Böhm, Berens, & Kobak 2022), finding connections between NE and CL methods (Damrich et al. 2023), new visualization techniques (Böhm, Berens, & Kobak 2023), or between NE and graph visualization (Böhm et al. 2025).

The knowledge that was gathered over the years allows the application of new methods to novel data modalities in an end-to-end fashion, giving rise to improved methods for dataset exploration. Thus, the field of data science benefits from this, as dataset visualization methods are already pervasive in applied data science in many scientific domains.

Improved visualization methods facilitate the understanding of complicated datasets. To that end, the research furthers the progress of science in general, as it will help uncover more subtle signals from data that were previously not visible and thus overlooked by researchers.

While visualization methods are able to inform research, they also require an understanding of how they work in order to bring out salient information. This is even more pronounced with methods based on CL as they add even more flexibility to model relationships in the data. Thus, dataset visualization is not a panacea and will not immediately solve every scientific inquiry that we currently have. Instead,

domain-specific knowledge is still required in order to harness the full potential, meaning that collaboration across disciplines will remain important.

This development is encouraging because it means that we are able to extract more information from data that has already been collected. It also runs against developments that suggest replacing experts by machine learning algorithms, as visualizations still requires interpretation by an expert.

The previous section outlined several avenues that can improve upon the current state of the art; roughly grouped into

- applications,
- technical improvements,
- method improvements, and
- theoretical developments.

Research into any of these topics should provide new and interesting insights into what drives visualization methods. The improved application will then in turn make it possible to better understand modern and complicated datasets.

More and more datasets are being produced where the data does not lie on a Riemannian manifold. Thus it is necessary to explore techniques that go beyond this approach, which has been outlined in this dissertation.

Publications by the author

This chapter includes the full papers prepared by the author during the doctoral studies.

- Jan Niklas Böhm, Philipp Berens, & Dmitry Kobak (Mar. 2022). "Attraction-Repulsion Spectrum in Neighbor Embeddings". In: *Journal of Machine Learning Research* 23.95, pp. 1–32
- Jan Niklas Böhm, Philipp Berens, & Dmitry Kobak (July 2023). "Unsupervised visualization of image datasets using contrastive learning". In: *International Conference on Learning Representations*
- Jan Niklas Böhm, Marius Keute, Alica Guzmán, Sebastian Damrich, Andrew Draganov, & Dmitry Kobak (Jan. 2025). "Node Embeddings via Neighbor Embeddings". In: *Under review*

In addition, there are multiple further works that were completed during my thesis, which also tie into the work presented herein:

- Sebastian Damrich, Jan Niklas Böhm, Fred A. Hamprecht, & Dmitry Kobak (July 2023). "From t-SNE to UMAP with contrastive learning". In: *International Conference on Learning Representations*
- Ifeoma Veronica Nwabufo, Jan Niklas Böhm, Philipp Berens, & Dmitry Kobak (2024). "Self-supervised Visualisation of Medical Image Datasets". In: *arXiv*
- Lisa Schmors, Dominic Gonschorek, Jan Niklas Böhm, Yongrong Qiu, Na Zhou, Dmitry Kobak, Andreas Tolias, Fabian Sinz, Jacob Reimer, Katrin Franke, Sebastian Damrich, & Philipp Berens (Jan. 2025). "TRACE: Contrastive learning for multi-trial time series data in neuroscience". In: *Under review*
- Andrew Draganov, Sharvaree Vadgama, Sebastian Damrich, Jan Niklas Böhm, Lucas Maes, Dmitry Kobak, & Erik Bekkers (Jan. 2025). "On the Importance of Embedding Norms in Self-Supervised Learning". In: *Under review*

Journal of Machine Learning Research 23 (2022) 1-32 Submitted 1/21; Revised 12/21; Published 3/22

Attraction-Repulsion Spectrum in Neighbor Embeddings

Jan Niklas Böhm JAN-NIKLAS.BOEHM@UNI-TUEBINGEN.DE
Philipp Berens PHILIPP.BERENS@UNI-TUEBINGEN.DE
Dmitry Kobak DMITRY.KOBAK@UNI-TUEBINGEN.DE
Institute for Ophthalmic Research, University of Tübingen
Otfried-Müller-Str. 25; 72076 Tübingen, Germany

Editor: Samuel Kaski

Abstract

Neighbor embeddings are a family of methods for visualizing complex high-dimensional data sets using kNN graphs. To find the low-dimensional embedding, these algorithms combine an attractive force between neighboring pairs of points with a repulsive force between all points. One of the most popular examples of such algorithms is t-SNE. Here we empirically show that changing the balance between the attractive and the repulsive forces in t-SNE using the exaggeration parameter yields a spectrum of embeddings, which is characterized by a simple trade-off: stronger attraction can better represent continuous manifold structures, while stronger repulsion can better represent discrete cluster structures and yields higher kNN recall. We find that UMAP embeddings correspond to t-SNE with increased attraction; mathematical analysis shows that this is because the negative sampling optimization strategy employed by UMAP strongly lowers the effective repulsion. Likewise, ForceAtlas2, commonly used for visualizing developmental single-cell transcriptomic data, yields embeddings corresponding to t-SNE with the attraction increased even more. At the extreme of this spectrum lie Laplacian eigenmaps. Our results demonstrate that many prominent neighbor embedding algorithms can be placed onto the attraction-repulsion spectrum, and highlight the inherent trade-offs between them.

Keywords: dimensionality reduction, neighbor embedding, visualization

1. Introduction

T-distributed stochastic neighbor embedding (t-SNE) (van der Maaten and Hinton, 2008) is arguably among the most popular methods for low-dimensional visualization of complex high-dimensional data sets. It defines pairwise similarities called *affinities* between points in the high-dimensional space and aims to arrange the points in a low-dimensional space to match these affinities (Hinton and Roweis, 2003). Affinities decay exponentially with high-dimensional distance, making them infinitesimal for most pairs of points and making the $n \times n$ affinity matrix effectively sparse. Efficient implementations of t-SNE (van der Maaten, 2014; Linderman et al., 2019) explicitly truncate the affinities and use the k-nearest-neighbor (kNN) graph of the data with $k \ll n$ as the input.

We use the term *neighbor embedding* (NE) to refer to all dimensionality reduction methods that operate on the kNN graph of the data and aim to preserve neighborhood relationships (Yang et al., 2013, 2014). A prominent recent example of this class of algorithms is UMAP (McInnes et al., 2018), which has become popular in applied fields such as single-cell transcriptomics (Becht et al., 2019). It is based on stochastic optimization and typically produces more compact clusters than t-SNE.

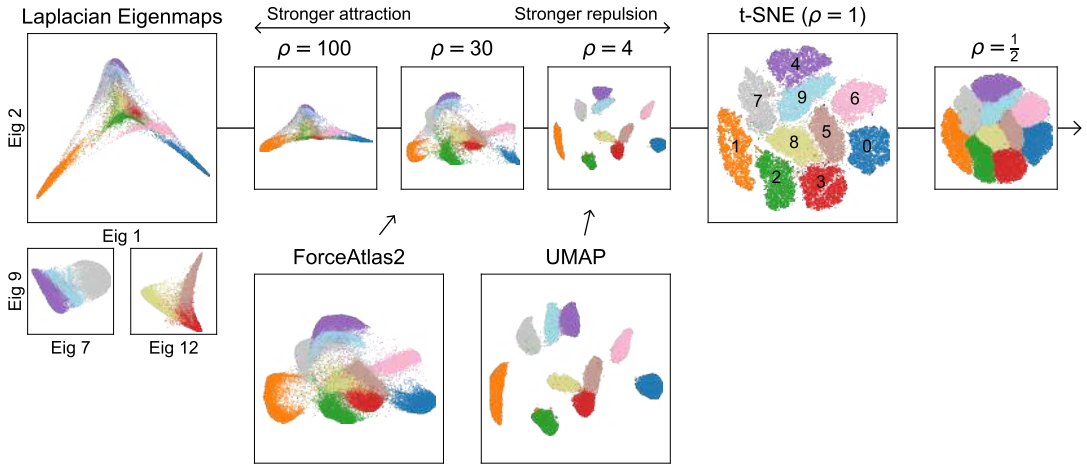

Figure 1: **Attraction-repulsion spectrum for the MNIST data.** Different embeddings of the MNIST data set of hand-written digits ($n = 70\,000$); colors denote digits as shown in the t-SNE panel. Multiplying all attractive forces by an exaggeration factor ρ yields a spectrum of embeddings. Values below 1 yield inflated clusters. Values above 1 yield more compact clusters. Higher values make multiple clusters merge, with $\rho \to \infty$ approximately corresponding to Laplacian eigenmaps (Linderman and Steinerberger, 2019). Insets show two subsets of digits separated in higher eigenvectors. UMAP is similar to $\rho \approx 4$. ForceAtlas2 is similar to $\rho \approx 30$.

Another example of neighbor embeddings are force-directed graph layouts (Noack, 2007, 2009), originally developed for graph drawing. One specific algorithm called ForceAtlas2 (Jacomy et al., 2014) has recently gained popularity in the single-cell transcriptomic community to visualize data sets capturing cells at different stages of development (Weinreb et al., 2018, 2020; Wagner et al., 2018a; Tusi et al., 2018; Kanton et al., 2019; Sharma et al., 2020).

In general, NE algorithms optimize the layout using attractive forces between all pairs of points connected by a kNN graph edge, thus placing them closer in the low-dimensional embedding. In addition, every point feels a repulsive force to every other point, which prevents trivial solutions, such as positioning all points on top of each other. While earlier algorithms took inspiration from physical systems (Fruchterman and Reingold, 1991), similar concepts arise naturally from the loss functions grounded in information theory (see below).

Here we provide a unifying account of NE algorithms. We study the spectrum of t-SNE embeddings that are obtained when increasing/decreasing the attractive forces between kNN graph neighbors, thereby changing the balance between attraction and repulsion. This leads to a trade-off between faithful representations of continuous and discrete structures. Remarkably, we discover that ForceAtlas2 and UMAP can both be accurately positioned on this spectrum (Figure 1). For UMAP, we use mathematical analysis and Barnes–Hut re-implementation to show that increased attraction is due to the negative sampling optimization strategy, and to derive the effective repulsion strength.

All our code is available at `https://github.com/berenslab/ne-spectrum`.

2. Related Work

Various trade-offs in SNE and t-SNE generalizations (Yang et al., 2009; Carreira-Perpiñán, 2010; Kobak et al., 2020; Venna et al., 2010; Amid et al., 2015; Amid and Warmuth, 2019; Narayan et al., 2015; Im et al., 2018) as well as in graph layout algorithms (Noack, 2007; Gansner et al., 2012) have been studied previously, but our work is the first to study the *exaggeration*-induced trade-off in t-SNE. Prior work used 'early exaggeration' only as an optimization trick (van der Maaten and Hinton, 2008) that allows to separate well-defined clusters (Linderman and Steinerberger, 2019; Arora et al., 2018).

Carreira-Perpiñán (2010) introduced the *elastic embedding* algorithm that has an explicit parameter λ controlling the attraction-repulsion balance. However, that paper suggests slowly increasing λ during optimization, as an optimization trick similar to the early exaggeration, and does not discuss trade-offs between high and low values of λ. The same holds for the graph layout model studied by Gansner et al. (2012), who similarly anneal the repulsion strength during optimization.

Our results on UMAP go against the common wisdom regarding what makes UMAP perform as it does (McInnes et al., 2018; Becht et al., 2019). No previous work suggested that negative sampling may have a drastic effect on the resulting embedding. In a follow-up paper, Damrich and Hamprecht (2021) analyzed the effective loss function of UMAP in more detail.

3. Neighbor Embeddings

The standard expositions of t-SNE, UMAP, and ForceAtlas2 (FA2) create the impression that these algorithms have little to do with each other. They use different affinities, different loss functions, different optimization strategies, and different large-sample approximations. They are introduced using different motivations. Importantly, the loss function in t-SNE includes a normalizing term which makes its optimization difficult, whereas the loss functions of UMAP and FA2 do not have such a term.

Despite all these differences, we claim that these algorithms are intimately related (Figure 1). In this section, we cast t-SNE, UMAP, FA2, and Laplacian eigenmaps (LE) in a common mathematical framework, using consistent notation and highlighting the similarities between them. The empirical results will be presented in the following sections. We denote the original high-dimensional points as $\mathbf{x}_i$ and their low-dimensional positions as $\mathbf{y}_i$.

3.1 T-SNE

T-SNE measures similarities between $\mathbf{x}_i$ by *affinities* v_{ij} and *normalized affinities* p_{ij}

$$p_{ij} = \frac{v_{ij}}{n}, \quad v_{ij} = \frac{p_{i|j} + p_{j|i}}{2}, \quad p_{j|i} = \frac{v_{j|i}}{\sum_{k\neq i} v_{k|i}}, \quad v_{j|i} = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2}\right).$$

For fixed i, $p_{j|i}$ is a probability distribution over all points $j \neq i$ (all $p_{i|i}$ are set to zero), and the variance of the Gaussian kernel σ_i^2 is chosen to yield a pre-specified value of the *perplexity* of this probability distribution, $\mathcal{P} = 2^{\mathcal{H}}$, where $\mathcal{H} = -\sum_{j\neq i} p_{j|i} \log_2 p_{j|i}$ is the entropy. The symmetrized affinities v_{ij} are then normalized by n for p_{ij} to form a probability distribution over the set of all pairs of points (i, j). Modern implementations (van der Maaten, 2014; Linderman et al., 2019) construct a kNN graph with $k = 3\mathcal{P}$ neighbors and only consider affinities between connected nodes as non-zero. The default perplexity value in most implementations is $\mathcal{P} = 30$.

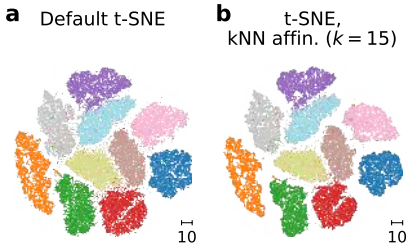

Figure 2: **The role of affinities in t-SNE.** MNIST data set. **(a)** Default t-SNE, Gaussian affinities, perplexity 30. **(b)** t-SNE with binary kNN affinities: all nonzero p_{ij} are the same, and $p_{ij} > 0$ iff point i is among 15 nearest neighbors of point j, or vice versa.

While t-SNE traditionally uses Gaussian affinities, the affinity matrix can be simplified without having a large impact on the resulting layout. In particular, one can use the kNN ($k = 15$) adjacency matrix $\mathbf{A} = [a_{ij}]$ to construct symmetric binary affinities $v_{ij} = a_{ij} \vee a_{ji}$, and then obtain p_{ij} by normalizing the entire matrix to sum to 1. The resulting 'kNN affinities' typically yield t-SNE embeddings that are almost identical to the default ones (Figure 2).

Similarities in the low-dimensional space are defined as

$$q_{ij} = \frac{w_{ij}}{Z}, \quad w_{ij} = \frac{1}{1 + d_{ij}^2}, \quad d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|, \quad Z = \sum_{k \neq l} w_{kl},$$

with all q_{ii} set to 0. The points $\mathbf{y}_i$ are then rearranged in order to minimize the Kullback–Leibler (KL) divergence $\mathcal{D}_{\mathrm{KL}}(\{p_{ij}\} \,\|\, \{q_{ij}\}) = \sum_{i,j} p_{ij} \log (p_{ij}/q_{ij})$ between p_{ij} and q_{ij}

$$\mathcal{L}_{\text{t-SNE}} \sim -\sum_{i,j} p_{ij} \log \frac{w_{ij}}{Z} = -\sum_{i,j} p_{ij} \log w_{ij} + \log \sum_{i,j} w_{ij}, \tag{1}$$

where we drop constant terms and take into account that $\sum p_{ij} = 1$. The first term contributes attractive forces to the gradient while the second term yields repulsive forces. Indeed, using $\partial w_{ij}/\partial \mathbf{y}_i = -2w_{ij}^2(\mathbf{y}_i - \mathbf{y}_j)$, the gradient, up to a constant factor, can be written as

$$\frac{\partial \mathcal{L}_{\text{t-SNE}}}{\partial \mathbf{y}_i} \sim \sum_j v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \frac{n}{Z} \sum_j w_{ij}^2 (\mathbf{y}_i - \mathbf{y}_j). \tag{2}$$

3.2 Exaggeration in t-SNE

A standard optimization trick for t-SNE called *early exaggeration* (van der Maaten and Hinton, 2008; van der Maaten, 2014) is to multiply the first sum in the gradient by a factor $\rho > 1$ during the initial iterations of gradient descent. This increases the attractive forces and allows similar points to gather into clusters more effectively. Modern implementations use $\rho = 12$ for the initial 250 iterations (van der Maaten, 2014) by default. The gradient of t-SNE with exaggeration can be written as

$$\frac{\partial \mathcal{L}_{\text{t-SNE}}(\rho)}{\partial \mathbf{y}_i} \sim \sum_j v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \frac{n}{\rho Z} \sum_j w_{ij}^2 (\mathbf{y}_i - \mathbf{y}_j) \tag{3}$$

and the corresponding loss function can be written in a functional form akin to the KL divergence

$$\mathcal{L}_{\text{t-SNE}}(\rho) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{w_{ij}/Z^{\frac{1}{\rho}}}. \tag{4}$$

However, for $\rho \neq 1$ the values $w_{ij}/Z^{\frac{1}{\rho}}$ in the denominator do not sum to 1 so Eq. (4) is not a KL divergence between probability distributions.

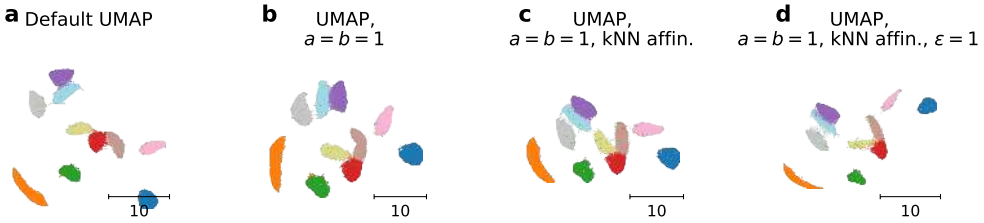

Figure 3: **UMAP with various simplifications.** MNIST data set. **(a)** Default UMAP with $a \approx 1.6$ and $b \approx 0.9$ and LE initialization. **(b)** UMAP with $a = b = 1$ and PCA initialization, the default choice for our experiments. **(c)** The same as in (b), but using binary kNN affinities ($v_{ij} = 1$ iff point i is among 15 nearest neigbors of point j, or vice versa). **(d)** The same as in (c), but with $\epsilon = 1$.

3.3 UMAP

Using the same notation as above, UMAP aims to optimize the cross-entropy loss between v_{ij} and w_{ij}, without normalizing them into probabilities:

$$\mathcal{L}_{\text{UMAP}} = \sum_{i,j}\left[v_{ij}\log\frac{v_{ij}}{w_{ij}} + (1-v_{ij})\log\frac{1-v_{ij}}{1-w_{ij}}\right], \tag{5}$$

where the $1 - v_{ij}$ term is approximated by 1 as most v_{ij} are 0, as done in the original implementation of UMAP (see Sainburg et al., 2021, Section 2.3.2). Note that UMAP differs from t-SNE in how exactly it defines v_{ij} (it uses adaptive Laplacian kernel with $k = 15$ by default), but its result does not change much when using the same binary affinities v_{ij} we introduced above for t-SNE (Figure 3c). Therefore, we believe that the difference in affinities is not what drives the difference in layout between t-SNE and UMAP in practice; see below for the experimental evidence.

Dropping terms that do not depend on $\mathbf{y}_i$, we obtain

$$\mathcal{L}_{\text{UMAP}} \sim -\sum_{i,j} v_{ij}\log w_{ij} - \sum_{i,j}\log(1-w_{ij}), \tag{6}$$

which is the same loss function as the one introduced earlier in LargeVis (Tang et al., 2016). The first term, corresponding to attractive forces, is the same as in t-SNE, but the second, repulsive, term is different. Taking $w_{ij} = 1/(1+d_{ij}^2)$ as in t-SNE, the UMAP gradient is given by

$$\frac{\partial\mathcal{L}_{\text{UMAP}}}{\partial\mathbf{y}_i} \sim \sum_j v_{ij}w_{ij}(\mathbf{y}_i-\mathbf{y}_j) - \sum_j \frac{1}{d_{ij}^2+\epsilon}w_{ij}(\mathbf{y}_i-\mathbf{y}_j), \tag{7}$$

where $\epsilon = 0.001$ is added to the denominator to prevent numerical problems for $d_{ij} \approx 0$. Note that UMAP uses $w_{ij} = 1/(1+ad_{ij}^{2b})$ as an output kernel with $a \approx 1.6$ and $b \approx 0.9$ by default. However, setting $a = b = 1$ does not strongly affect the result (Figure 3b). Moreover, when we modified the UMAP implementation to set $\epsilon = 1$, the resulting embeddings also stayed qualitatively similar (Figure 3d). So here again, we believe that these details are not what drives the difference in layout between t-SNE and UMAP in practice; see below for the experimental evidence.

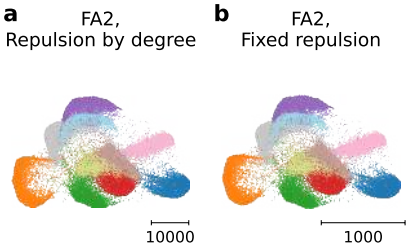

Figure 4: **The effect of edge repulsion in ForceAtlas2.** MNIST data set. **(a)** FA2 with repulsion by degree. **(b)** FA2 without repulsion by degree. Note the difference in scale.

If $\epsilon = 1$, the gradient becomes identical to the t-SNE gradient, up to the n/Z factor in front of the repulsive forces. Moreover, UMAP implementation allows to use an arbitrary γ factor in front of the repulsive forces, which makes it easier to compare the loss functions

$$\frac{\partial \mathcal{L}_{\text{UMAP}}(\gamma)}{\partial \mathbf{y}_i} \sim \sum_j v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \gamma \sum_j \frac{1}{d_{ij}^2 + \epsilon} w_{ij} (\mathbf{y}_i - \mathbf{y}_j). \tag{8}$$

Note that LargeVis used $\gamma = 7$ by default but UMAP sets $\gamma = 1$, as follows from its cross-entropy loss function.

Whereas it is possible to approximate the full repulsive term with the same techniques as used in t-SNE (van der Maaten, 2014; Linderman et al., 2019), UMAP takes a different approach and follows LargeVis in using *negative sampling* (Mikolov et al., 2013) of repulsive forces: on each gradient descent iteration, only a small number m of randomly picked repulsive forces are applied to each point for each of the $\sim k$ attractive forces that it feels. Other repulsive terms are ignored. The default value is $m = 5$. The effect of negative sampling on the resulting embedding has not been studied before.

3.4 ForceAtlas2

Force-directed graph layouts are usually introduced directly via attractive and repulsive forces, even though it is easy to write down a suitable loss function (Noack, 2007). ForceAtlas2 (FA2) has attractive forces proportional to d_{ij} and repulsive forces proportional to $1/d_{ij}$ (Jacomy et al., 2014):

$$\frac{\partial \mathcal{L}_{\text{FA2}}}{\partial \mathbf{y}_i} = \sum_j v_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \sum_j \frac{(h_i + 1)(h_j + 1)}{d_{ij}^2} (\mathbf{y}_i - \mathbf{y}_j), \tag{9}$$

where h_i denotes the degree of node i in the input graph. This is known as *edge repulsion* in the graph layout literature (Noack, 2007, 2009) and is important for embedding graphs that have nodes of very different degrees. However, for symmetrized kNN graphs, assuming that they do not have too many 'hubs' (Radovanovic et al., 2010), $h_i \approx k$, so $(h_i + 1)(h_j + 1)$ term contributes a roughly constant $\sim k^2$ factor to the repulsive forces, and can be compensated by decreasing all distances by a factor of k. Indeed, for the MNIST data set, removing the edge repulsion factor led to a $\sim$15 times decrease in scale but otherwise almost the same embedding (Figure 4).

3.5 Laplacian Eigenmaps

Laplacian eigenmaps (Belkin and Niyogi, 2002; Coifman and Lafon, 2006) is a method for dimensionality reduction that leverages spectral graph theory. Its loss function can be written with a quadratic constraint

$$\mathcal{L}_{\mathrm{LE}} = \sum_{ij} v_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \;\; \text{s.t.} \;\; \mathbf{Y}^\top \mathbf{D} \mathbf{Y} = \mathbf{I}, \tag{10}$$

where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j V_{ij}$ for affinity matrix $\mathbf{V} = [v_{ij}]$, $\mathbf{I}$ is the identity matrix, and $\mathbf{Y}$ is the embedding matrix having $\mathbf{y}_i$ as rows. This loss function can be minimized by solving a generalized eigenvalue problem (Appendix A). The quadratic constraint in some sense serves the role of repulsive forces, preventing collapse of the embedding to a single point.

Carreira-Perpiñán (2010) and Linderman and Steinerberger (2019) noticed that the attractive term in the t-SNE loss function on its own reduces to the unconstrained loss function of Laplacian eigenmaps. Indeed, if $\rho \to \infty$, the relative repulsion strength becomes infinitesimal and the embedding shrinks to a point with all $w_{ij} \to 1$. This means that the gradient from Equation 2 reduces to $\sum_j v_{ij}(\mathbf{y}_i - \mathbf{y}_j)$, which coincides with the gradient of Laplacian eigenmaps (apart from the quadratic constraint). A more detailed analysis in Appendix A shows that when $\rho \to \infty$, the entire embedding shrinks to a single point, but the leading eigenvectors of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{V}$ shrink the slowest. This makes t-SNE with large values of ρ produce embeddings very similar to LE, which computes the leading eigenvectors of the normalized Laplacian (Appendix A).

This theoretical finding immediately suggests that it might be interesting to study t-SNE with exaggeration $\rho > 1$ not only as an optimization trick, but in itself, as an intermediate method between LE and standard t-SNE.

3.6 Implementation

All experiments were performed in Python. We ran all packages with default parameters, unless specified otherwise. We used openTSNE 0.6.0 (Poličar et al., 2019), a Python reimplementation of FIt-SNE (Linderman et al., 2019). When using $\rho < 12$, we used the default early exaggeration with $\rho_{\mathrm{early}} = 12$, and exaggeration ρ for all subsequent iterations. For $\rho \geq 12$ no early exaggeration was used and exaggeration ρ was applied throughout. The learning rate was set to $\eta = n/\max(\rho, \rho_{\mathrm{early}})$ as suggested by Belkina et al. (2019). Note that we used default Gaussian affinities for all experiments.

We used UMAP 0.5.1 with Cauchy similarity kernel (by setting $a = b = 1$). We used default UMAP affinities for all experiments. The Barnes–Hut implementation of UMAP was developed in Cython (Behnel et al., 2011), on top of the `openTSNE` package. We extended the package to leave out the calculation of the normalization constant Z, take into account the ϵ and γ parameters from Equation 8, and load the default UMAP affinities as computed by UMAP itself. For these experiments we also set $a = b = 1$.

For FA2 we used the `fa2` package (Chippada, 2017), which employs a Barnes–Hut approximation to speed up computation of the repulsive forces. We developed a patch that makes it possible to disable the repulsion by degree and applied it on top of the current version 0.3.5. The input to FA2 was the unweighted symmetrized approximate kNN graph $\mathbf{A} \vee \mathbf{A}^\top$, where $\mathbf{A}$ is the kNN adjacency matrix constructed with `Annoy` (Bernhardsson, 2013) with $k = 15$. By default, all algorithms were optimized for 750 iterations.

Unless stated otherwise, we used principal component analysis (PCA) initialization to remove any differences between algorithms due to initialization strategies (Kobak and Linderman, 2021) and

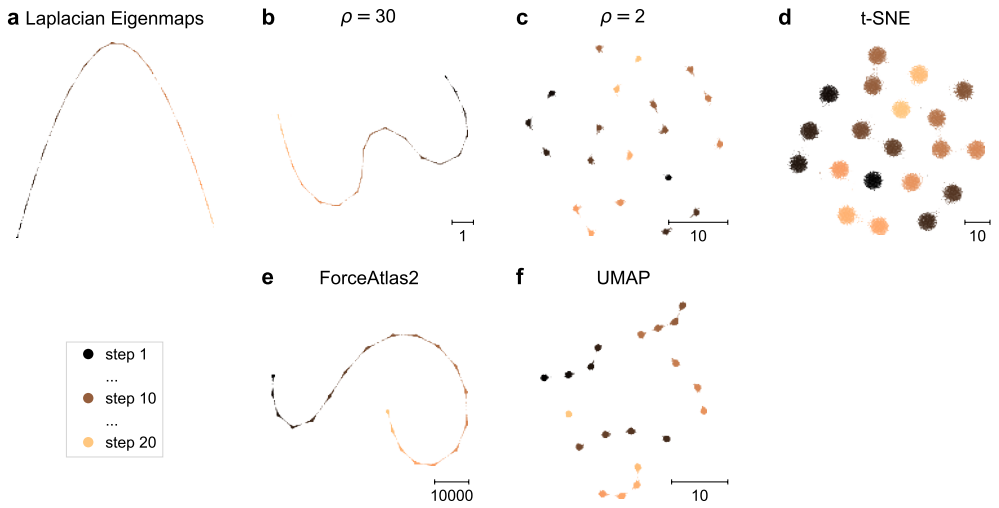

Figure 5: **Simulated data emulating a developmental trajectory.** The points were sampled from 20 isotropic 50-dimensional Gaussians, equally spaced along one axis such that only few inter-cluster edges exist in the kNN graph. Panels (b–f) used a shared random initialization. Panels (b–d) did not use early exaggeration.

to make all embeddings of the same data set visually aligned to each other (Kobak and Berens, 2019). For t-SNE, the initialization was always scaled to have a standard deviation of 0.0001, as suggested by Kobak and Berens (2019) and is default in `openTSNE` (Poličar et al., 2019). For UMAP, the initialization was scaled to have the range of $[-10, 10]$, as is default in the original implementation. For ForceAtlas2, we scaled the initialization to have a standard deviation of 10 000 to approximately match the scale of final ForceAtlas2 embeddings (we experimented with different values and found this setting to work well and avoid convergence problems). Note that Figure 5 is an exception and uses random initialization. By default, UMAP uses LE (and not PCA) initialization, whereas the `fa2` implementation uses random initialization. On the data sets analyzed here, we did not observe any qualitative difference as a result of our non-default (PCA) initialization (cf. Figure 3a,b).

LE was computed using the `scikit-learn` (Pedregosa et al., 2011) implementation, using the `SpectralEmbedding` class and the `PyAMG` solver 4.0 (Olson and Schroder, 2018). The input graph was the same as the input to FA2. No initialization was needed for LE. We flipped the signs of LE eigenvectors to orient them similarly to other embeddings, whenever necessary.

4. The Attraction-Repulsion Spectrum

We first investigated the relationships between the NE algorithms using the MNIST data set of hand-written digits (sample size $n = 70\,000$; dimensionality $28 \times 28 = 784$, reduced to 50 with PCA; Figure 1). T-SNE produced an embedding where all ten digits were clearly separated into clusters with little white space between them, making it difficult to assess relationships between digits. Increasing attraction to $\rho = 4$ shrank the clusters and strongly increased the amount of white space; it also identified two groups of graphically similar digits: “4/7/9” and “3/5/8”. Further increasing

the attraction to $\rho = 30$ made all clusters connect together: cluster "6" connected to "5" and to "0". Even higher exaggeration made the embedding similar to Laplacian eigenmaps, in agreement with the theoretical prediction discussed above (Linderman and Steinerberger, 2019). Here similar digit groups like "4/7/9" were entirely overlapping, and could only be separated using higher eigenvectors (Figure 1, insets). On the other side of the spectrum, exaggeration values $0 < \rho < 1$ resulted in inflated coalescing clusters.

The MNIST example suggests that high attraction emphasizes connections between clusters at the cost of within-cluster structure, whereas high repulsion emphasizes the cluster structure at the expense of between-cluster connections. We interpreted this finding as a *continuity-discreteness trade-off*.

We developed a simple toy example to illustrate this trade-off in more detail (Figure 5). For this, we generated data as draws from 20 standard isotropic Gaussians in 50 dimensions, each shifted by 6 standard deviation units from the previous one along one axis (1000 points per Gaussian, so $n = 20\,000$ overall). For this analysis we used random initialization and turned the early exaggeration off, to isolate the effect of each loss function on the 'unwrapping' of the random initial configuration.

We found that t-SNE with strong exaggeration ($\rho = 30$) recovered the underlying one-dimensional manifold structure of the data almost as well as LE (Figure 5a,b). In both cases, the individual clusters were almost invisible. In contrast, an embedding with weaker attraction and stronger repulsion (t-SNE with exaggeration $\rho = 2$) showed individual clusters but was unable to fully recover the 1-dimensional structure and only found some chunks of it (Figure 5c). Finally, standard t-SNE clearly showed 20 individual clusters but with the continuous structure entirely lost (Figure 5d).

Further, we analyzed a developmental single-cell transcriptomic data set, where cells were collected from human brain organoids at seven time points between 0 days and 4 months into the development (Kanton et al., 2019). In this kind of data, one expects to find rich cluster structure as well as a strong time-dependent trajectory. As in the other data sets, we found that stronger attraction ($\rho = 30$) better represented the developmental trajectory, whereas stronger repulsion (standard t-SNE) better represented the cluster structure (Figure 6). Using much higher k for the kNN graph construction made the developmental trajectory in high-attraction methods even clearer (Figure A1), in agreement with the FA2-based analysis performed in the original publication. We observed the same pattern in a separate data set obtained from chimpanzee brain organoids (Figures A2, A3).

While high exaggeration helps to preserve continuous structures, this comes with a price of distorting local neighborhoods. To quantify this effect, we computed the fraction of $k = 15$ nearest neighbors in high dimensions that remain among the nearest neighbors in the embedding ('kNN recall'). To compute it for a given data point, we found 15 points with the largest affinities in the symmetrized affinity matrix, and determined what fraction of them is among the 15 exact nearest neighbors in the embedding. This was averaged over 10 000 randomly selected points. We found that as ρ increased, the local neighborhood became more and more distorted (Figure 7). For the MNIST data set, the kNN recall of default t-SNE ($\rho = 1$) was 0.34; with $\rho = 4$ it went down to 0.12; with $\rho = 30$ it further dropped to 0.06.

We observed the same fast and monotonic decrease in kNN recall in both brain organoid data sets, as well as in six further data sets (Figure 7): Fashion MNIST (Xiao et al., 2017), Kannada MNIST (Prabhu, 2019), Kuzushiji MNIST (Clanuwat et al., 2018), single-cell data from hydra (Siebert et al., 2019), from zebrafish embryo (Wagner et al., 2018b), and from mouse cortex (Tasic et al., 2018).

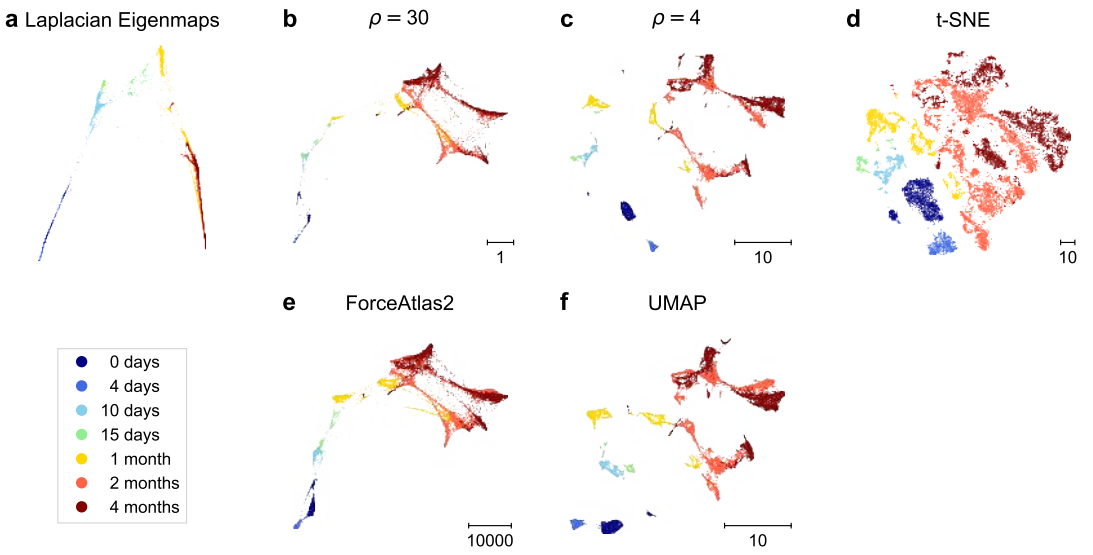

Figure 6: **Neighbor embeddings of the single-cell RNA-seq developmental data.** Cells were sampled from human brain organoids (cell line `409b2`) at seven time points between 0 days and 4 months into the development (Kanton et al., 2019). Sample size $n = 20\,272$. Data were reduced with PCA to 50 dimensions. See Appendix B for transcriptomic data preprocessing steps.

5. UMAP and ForceAtlas2 Can Be Placed on the Attraction-Repulsion Spectrum

Using the MNIST data set, we observed that FA2 produced an embedding very similar to t-SNE with $\rho \approx 30$, while UMAP produced an embedding very similar to t-SNE with $\rho \approx 4$ (Figure 1). The same was true for the simulated data set (Figure 5) and for the brain organoid data set (Figure 6), as well as for the seven further data sets that we analyzed in addition (Figures A2, A4, A5, A6, A7, A8, A9).

To quantify this observation, we computed distance correlations (Szekely et al., 2007) between UMAP & FA2 embeddings and t-SNE embeddings with various values of $\rho \in [1, 100]$ (Figure 8). We found that for most data sets the highest correlation between UMAP and t-SNE layouts was achieved at $4 \leq \rho < 15$ (Figure 8a). For FA2, the highest correlation was typically achieved at $20 < \rho < 80$ (Figure 8b). In both cases, the maximum correlations were above 0.94, indicating very similar layouts. Whereas the exact value of ρ yielding the maximum correlation varied between data sets, the correlation values at $\rho = 4$ for UMAP and at $\rho = 30$ for FA2 were always high and close to the maximum correlations. We also observed that $\rho = 4$ yielded a good match to UMAP independent of the sample size (Figure A10).

Note that for all three algorithms we used all default parameters (apart from always using the same PCA initialization and fixing $a = b = 1$ in UMAP), confirming that the differences between t-SNE and UMAP in affinities and in the value of ϵ in the loss function do not play a large role, at least for our data sets. We used the shared PCA initialization to roughly align the embeddings and make distance correlations more meaningful, but all conclusions stayed the same if we used default UMAP/FA2 initializations (see Section 3.6).

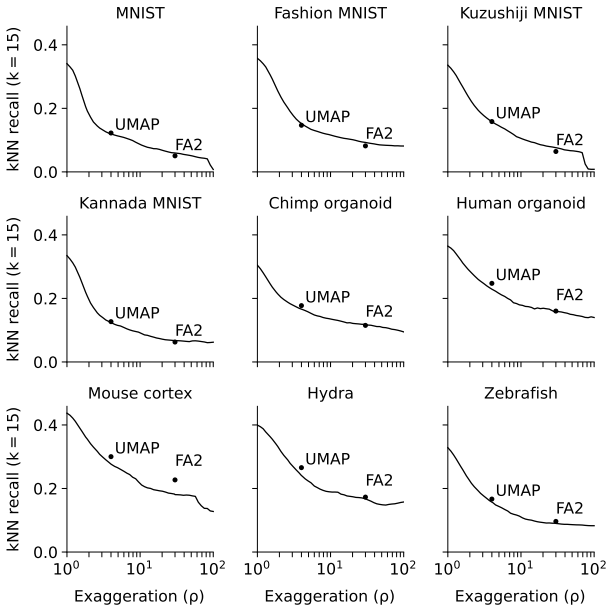

Figure 7: **Nearest neighbors recall as a function of ρ.** The fraction of $k = 15$ nearest neighbors in high dimensions that remain among the nearest neighbors in the embedding (average over 10 000 randomly selected points; see text). The values for UMAP and FA2 are shown at $\rho = 4$ and $\rho = 30$.

A caveat here is that distance correlation metric can be strongly affected by the exact placement of the clusters, and does not always capture the intuitive notion of 'similarity'. For example, both correlation curves for the Kannada MNIST data set (Figure 8, red lines) peak at almost the same value of ρ, but visual inspection of the embeddings (Figure A5) suggests that $\rho = 4$ is qualitatively closer to UMAP, while $\rho = 30$ is qualitatively closer to FA2, in agreement with all other data sets.

The kNN recall of UMAP and FA2 was also similar to the kNN recall of t-SNE with exaggeration set to $\rho = 4$ and $\rho = 30$ respectively, across all analyzed data sets (Figure 7). This suggests that not only the general layout, as measured by the distance correlation, but also the local structure of the embedding was similar between UMAP/FA2 and t-SNE with appropriate exaggeration.

6. Increased Attraction in UMAP Due to Negative Sampling

As shown above, the gradient of UMAP (Eq. 7) is very similar to the gradient of t-SNE (Eq. 2) but does not contain the 'normalizing' n/Z term in front of the repulsive forces. What are the typical values of this coefficient? The normalization term Z in t-SNE evolves during optimization: it starts at $Z \approx n^2$ due to all $d_{ij} \approx 0$ at initialization and decreases as the embedding expands. For a perfect embedding with all $p_{ij} = q_{ij}$ and $v_{ij} = w_{ij}$, Z would be equal to n; in reality Z usually still exceeds n. We found that for all nine data sets analyzed here, the value of Z in the end of optimization with $\rho = 1$ was in the range $[50n, 120n]$ (Figure 9a). For MNIST, the final Z value was $\sim 100n$, corresponding to the final $n/Z \approx 0.01$ (Figure 9b). Increasing the exaggeration shrinks the embedding and increases the final Z; it also changes the repulsive factor to $n/(\rho Z)$ (Eq. 3). Across all data sets, the final Z value with $\rho = 4$ was in the $[400n, 2300n]$ range (Figure 9a). For MNIST, it was $\sim 2100n$, corresponding to the final $n/(\rho Z) \approx 0.0001$. This means that UMAP matched t-SNE results with the repulsive factor 0.0001 better than it matched t-SNE results with the repulsive factor 0.01, even though UMAP itself uses repulsive factor $\gamma = 1$ (Eq. 7). How is this possible?

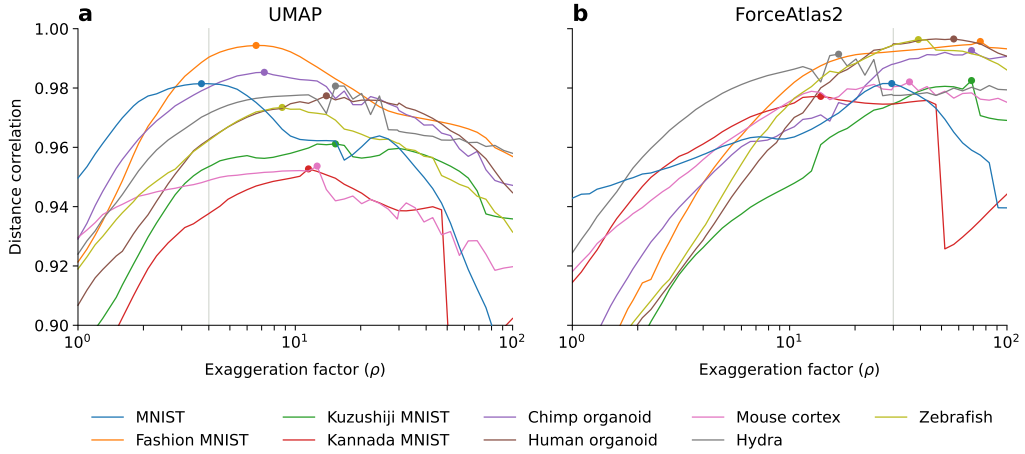

Figure 8: **Distance correlations between UMAP/FA2 and t-SNE.** Exaggeration values $\rho \in [1, 100]$ were evenly distributed on a log-scale, with $\rho = 4$ and $\rho = 30$ added explicitly; 52 points in total. Distance correlation (Szekely et al., 2007) was computed using `dcor` package (Carreño, 2017) on a random subset ($n = 5\,000$) of the data. Dots mark the maximum of each curve. **(a)** Distance correlation between UMAP and t-SNE. **(b)** Distance correlation between FA2 and t-SNE.

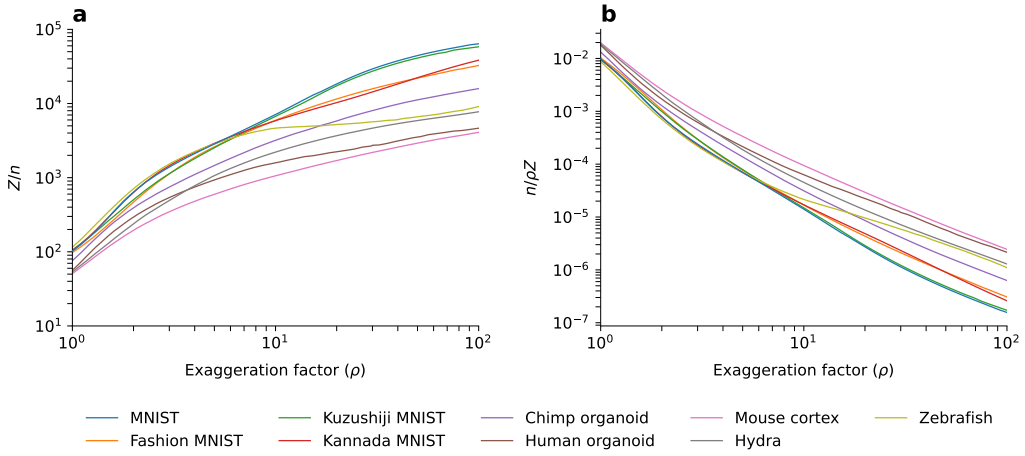

Figure 9: **(a)** The normalization term Z/n computed for all data sets considered in the manuscript, as a function of ρ. **(b)** The term $n/(\rho Z)$ computed for all data sets considered in the manuscript.

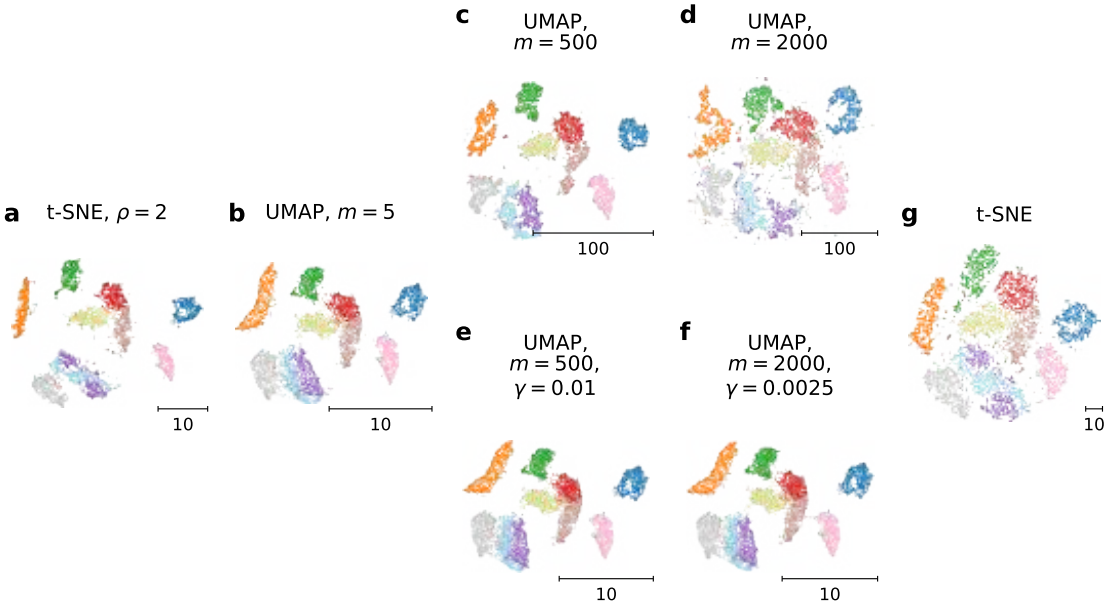

Figure 10: **The effect of negative sampling rate on UMAP embeddings.** MNIST subsample with $n = 6000$. We used a subsample of MNIST because the runtime scales as $O(mn)$, making it impractical to use $m \approx n$ for large n. UMAP (panels b–f) was run for 3000 epochs to ensure convergence, as large m introduce noise in the optimization. **(a)** T-SNE embedding with $\rho = 2$. **(b–d)** UMAP embeddings with $m \in \{5, 500, 2000\}$. Panels (c) and (d) were initialized with the standard UMAP embedding ($m = 5$, 750 epochs) to simulate early exaggeration. **(e–f)** UMAP embeddings with $m \in \{500, 2000\}$, while keeping the product $\gamma \cdot m$ constant. **(g)** Standard t-SNE of the same data.

We hypothesized that this mismatch arises because the UMAP implementation is based on negative sampling and does not in fact optimize its stated loss (Eq. 5). Instead, the negative sampling decreases the repulsion strength, creating an effective $\gamma_{\text{eff}}(m) \ll 1$. We verified that increasing the value of m increased the repulsion strength in UMAP (Figure 10): embeddings grew in size and the amount of between-cluster white space decreased. But when we decreased the γ factor together with increasing m so that their product $\gamma \cdot m$ stayed constant, the embedding did not change at all (Figure 10e,f), confirming that the negative sampling rate m directly controls the repulsion strength.

The repulsion strength in UMAP can also be explicitly controlled by the γ parameter. Decreasing the γ value had the same effect as increasing the ρ value in t-SNE, and moved the UMAP result towards the LE part of the attraction-repulsion spectrum (Figure A11). We found it not possible to increase the repulsion strength by setting $\gamma \gg 1$, likely due to convergence problems.

We can approximately compute the effective repulsion coefficient $\gamma_{\text{eff}}(m)$ arising in UMAP through the negative sampling as follows. The number of repulsive forces per one attractive force is $\sim n/k$ in the full gradient but m with negative sampling. This suggests that $\gamma_{\text{eff}} \approx km/n$ and should decrease with the sample size as $O(1/n)$.

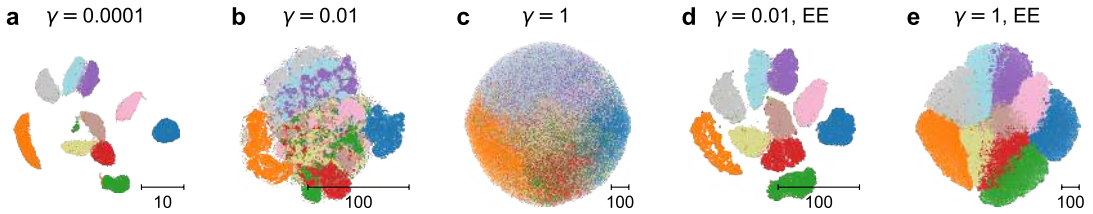

Figure 11: **Barnes–Hut UMAP without negative sampling.** **(a–c)** Embeddings with $\gamma \in \{0.0001, 0.01, 1\}$. **(d–e)** Embeddings with gamma values $\gamma \in \{0.01, 1\}$ initialized with the embedding with $\gamma = 0.0001$ [panel (a)], in analogy to early exaggeration in t-SNE.

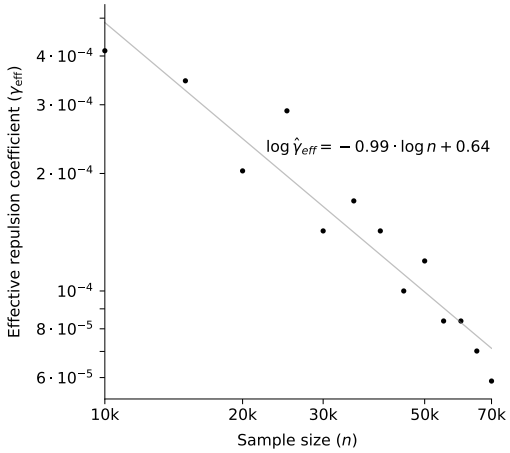

Figure 12: **Estimating the effective repulsion in UMAP.** We ran Barnes–Hut UMAP for subsets of MNIST with sample sizes from 10 000 to 70 000 in the increments of 5000, using a grid of γ values. For each sample size, we found the γ value matching the size of negative-sampling-based UMAP. The grey line shows the linear regression fit to the log-transformed data. Sample sizes below 10 000 led to noisy estimates and were excluded.

To confirm our interpretation, we developed a Barnes–Hut UMAP implementation that optimizes the full UMAP loss without any negative sampling (see Section 3.6). On full MNIST, $\gamma = 0.0001$ yielded an embedding that resembled the standard (negative-sampling-based) UMAP (Figure 11a), while larger values of γ yielded over-repulsed embeddings (Figure 11b,c) and required early exaggeration to produce meaningful results (Figure 11d,e), with $\gamma = 0.01$ resembling t-SNE and $\gamma = 1$ being over-repulsed compared to t-SNE. This suggests that directly optimizing the cross-entropy loss (Eq. 5) leads to an embedding where the repulsive forces strongly dominate visual appearance (Figure 11c,e).

Furthermore, we can use our Barnes–Hut implementation to directly estimate γ_{eff}. To do that, we ran the Barnes–Hut implementation for a range of sample sizes and a range of γ values (40 values of γ evenly log-spaced between 0.01 and 10^{-5}). For each sample size, we found the γ value giving the best match to the size ($\max \mathbf{Y} - \min \mathbf{Y}$) of the negative-sampling-based UMAP (Figure 12). The resulting $\hat{\gamma}_{\text{eff}}$ estimates decreased with the sample size as $1/n$, with regression slope being -0.99 on the log-log plot (Figure 12), confirming our prediction. In a follow-up work, Damrich and Hamprecht (2021) analyzed the UMAP sampling procedure in more detail, showing that $\gamma_{\text{eff}} \approx \log k \cdot m/n$.

In popular expositions (Coenen and Pearce, 2019; Oskolkov, 2019), the success of UMAP and its visually appealing embeddings have been attributed to its cross-entropy loss function and its topological foundations. However, our conclusion is that the more condensed clusters typically

observed in UMAP compared to t-SNE are a serendipitous consequence of UMAP's negative sampling strategy. The mathematical framework and the cross-entropy loss function developed in the original paper (McInnes et al., 2018) would lead to very different and suboptimal embeddings (Figure 11c,e), if not for the negative sampling. Note that we are not criticizing the actual UMAP embeddings here; our statement is that UMAP's stated and effective loss functions are qualitatively different.

7. Increased Attraction in FA2 Due to Non-Decaying Attractive Forces

The attractive forces in t-SNE scale as $d_{ij}/(1 + d_{ij}^2)$. When all d_{ij} are small, this becomes an approximately linear dependency on d_{ij}, which is the reason why t-SNE with high exaggeration $\rho \gg 1$ replicates Laplacian eigenmaps (see Section 3.5 and Appendix A). For large distances d_{ij}, attractive forces in t-SNE decay to zero, making default t-SNE very different from LE. In contrast, in FA2, attractive forces always scale as d_{ij}. Thus, the larger the embedding distance between two points, the stronger the attractive force between them. This strong non-decaying attractive force moves FA2 towards Laplacian eigenmaps on the attraction-repulsion spectrum.

While the attractive forces in FA2 are the same as in Laplacian eigenmaps, FA2 has repulsive forces instead of the quadratic constraint of LE. This moves FA2 somewhat away from LE on the attraction-repulsion spectrum. These arguments provide a qualitative explanation for why FA2 behaves similar to t-SNE with strong exaggeration ($\rho \approx 30$, as we empirically showed above), but more quantitative analysis remains for future work. In addition, our arguments suggest that the exact scaling law of the repulsive forces ($1/d_{ij}^2$ or $1/d_{ij}$) may have little qualitative influence on the resulting embedding as long as the attractive forces remain linear in d_{ij}. We leave it for future work to investigate this.

Note that it is not possible to move FA2 embeddings along the attraction-repulsion spectrum by multiplying the attractive or repulsive forces by a constant factor (such as γ in UMAP or ρ in t-SNE). Multiplying attractive forces by any factor a or repulsive forces by any factor $1/a$ only leads to rescaling of the embedding by $1/\sqrt{a}$. Indeed, if all forces are in equilibrium before such multiplication and rescaling, they will stay in equilibrium afterwards. This is a general property of force-directed layouts where both attractive and repulsive forces scale as powers of the embedding distance d_{ij}.

8. Discussion

We showed that changing the balance between attractive and repulsive forces in t-SNE directly affects the trade-off between preserving continuous/global or discrete/local structures. Increasingly strong repulsion 'brings out' information from higher Laplacian eigenvectors into the two embedding dimensions (Figure 1). It is remarkable that the repulsive forces, which are data-agnostic and do not depend on the input data (Carreira-Perpiñán, 2010), have so much qualitative influence.

While we only considered the exaggeration factor ρ here, other parameters of t-SNE can also qualitatively affect the resulting embedding. In particular, the tail-heaviness of the low-dimensional similarity kernel (Yang et al., 2009) controls the emphasis put on the fine cluster structure of the data (Kobak et al., 2020). There is thus non-trivial interaction between the exaggeration ρ and the tail-heaviness parameter α, which we illustrate using the MNIST data set in Figure A12, but leave more detailed exploration of this two-dimensional parameter space for future work.

Our results suggest that it is beneficial for high-repulsion embeddings to begin optimization with lower repulsion strength, in order to better preserve global structure. This explains how UMAP benefits from its default initialization with Laplacian eigenmaps (Kobak and Linderman, 2021) and how t-SNE benefits from early exaggeration (Linderman and Steinerberger, 2019) (Figure A13 demonstrates the importance of early exaggeration in t-SNE). Similarly, Carreira-Perpiñán (2010) suggested to gradually increase repulsion strength during optimization of elastic embedding. A promising approach to t-SNE optimization would be to use Laplacian eigenmaps for initialization and replace the early exaggeration phase with gradual annealing of the exaggeration factor ρ from 'infinity' down to its final desired value.

Our treatment provides a unified perspective on several well-known NE algorithms that have scalable implementations and that have been shown to successfully embed data sets such as MNIST without coarse-graining the kNN graph. Methods based on coarse-graining, such as PHATE (Moon et al., 2019) or latent variable NE method in Saul (2020) may behave differently. We believe that our treatment may allow to position other NE algorithms on the same spectrum. For example, a recently suggested TriMap algorithm (Amid and Warmuth, 2019), which uses negative sampling similar to UMAP, appears to have stronger attractive forces than UMAP (cf. Figure 5 in the original paper), with some TriMap embeddings, e.g. of the Fashion MNIST data set, looking similar to the ForceAtlas2 embeddings shown in our work. It remains for future work to investigate if and how some of the more recent NE algorithms based on negative sampling fit on the attraction-repulsion spectrum. This includes, for example, IHVD (Minch et al., 2020) and MDE (Agrawal et al., 2021). The latter work developed a flexible NE framework that can combine various attractive and repulsive forces optimized using negative sampling, with the quadratic constraint of Laplacian eigenmaps, resulting in a rich family of embeddings.

We argue that negative sampling (Mikolov et al., 2013), used by LargeVis/UMAP, strongly lowers the effective repulsion, compared to the stated cross-entropy loss function. In a follow-up work, Damrich and Hamprecht (2021) have developed a more formal analysis of negative sampling in UMAP and confirmed our findings.

Negative sampling exhibits some similarity to stochastic gradient descent (SGD), where the gradient is repeatedly computed on small random subsets of the data, known as mini-batches. However, we believe that this analogy is not helpful. SGD iterates over the entire training set, partitioned in mini-batches. Small mini-batches increase the variance of the gradient estimates but do not introduce any bias. Negative sampling, on the other hand, only samples a small subset of the repulsive forces for each attractive force, introducing a systematic bias into the gradient computation.

Negative sampling is closely related to the *noise-contrastive estimation* (NCE) framework (Gutmann and Hyvärinen, 2012). NCE was recently applied to t-SNE under the name of NCVis (Artemenkov and Panov, 2020), and the general NCE theory asserts that it should be asymptotically equivalent to optimizing the full gradient (Gutmann and Hyvärinen, 2012). We consider it an interesting research direction to study the relationship between negative sampling and NCE and their effect on 2D embeddings as well as on higher-dimensional embeddings used in methods like `word2vec` (Mikolov et al., 2013).

The practical takeaway from our work is not that one of the considered algorithms is the 'best'. All three algorithms discussed in this manuscript (t-SNE, UMAP, ForceAtlas2) are widely used in several academic fields, like single-cell biology (Becht et al., 2019; Kobak and Berens, 2019) or population genomics (Diaz-Papkovich et al., 2019; Karczewski et al., 2020). But the choice of the method is often done without a solid understanding of why the results may be different or what

trade-offs are at play. Our work highlights that which algorithm is more appropriate may depend on the question one wants to answer.

If the goal of the analysis is to explore the fine clusters and the local neighborhood structure, one should use t-SNE with default exaggeration parameter $\rho = 1$ (Figure 7). If the interest rather lies in exploring the global structure of the data, such as continuous or temporal trajectories, then it may be helpful to increase the exaggeration in t-SNE or to use UMAP or ForceAtlas2. All things considered, t-SNE with exaggeration is able to cover the entire attraction-repulsion spectrum (Figure 1), whereas UMAP with negative sampling is only able to produce visualizations with relatively high attraction. ForceAtlas2 is even more limited in that it cannot move along the spectrum and corresponds to high attraction. In terms of the running speed, modern implementations of t-SNE (Linderman et al., 2019; Artemenkov and Panov, 2020) and UMAP are comparable.

We hope that the treatment developed here will allow researchers to make an informed choice between algorithms in practical applications.

Acknowledgments

We thank Sebastian Damrich, George Linderman, Stefan Steinerberger, James Melville, and Ulrike von Luxburg for helpful discussions; Pavlin Poličar for discussions and `openTSNE` support; and He Zhisong for the help with loading the organoid transcriptomic data.

This research was funded by the Cyber Valley Research Fund (D.30.28739), the Deutsche Forschungsgemeinschaft through a Heisenberg Professorship (BE5601/4-1), the Excellence Cluster 2064 "Machine Learning: New Perspectives for Science" (390727645), the National Institute Of Mental Health of the National Institutes of Health (U19MH114830), and the German Ministry of Education and Research (01IS18039A, 01GQ1601). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. The authors declare no financial conflict of interest and did not receive any donations/funding from industry with relationship to this project.

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Jan Niklas Böhm.

Appendix A. Relationship to Laplacian Eigenmaps

Laplacian eigenmaps Let a $n \times n$ symmetric matrix $\mathbf{V}$ contain pairwise affinities between n points (or edge weights between nodes in an undirected graph). Let the diagonal matrix $\mathbf{D}$ contain row (or, equivalently, column) sums of $\mathbf{V}$, that is, $D_{ii} = \sum_j V_{ij}$. Then $\mathbf{L} = \mathbf{D} - \mathbf{V}$ is known as the (unnormalized) graph Laplacian. Laplacian eigenmaps (Belkin and Niyogi, 2002) can then be formulated as solving the generalized eigenvector problem

$$\mathbf{La} = \lambda \mathbf{Da}$$

and taking the eigenvectors corresponding to the *smallest* eigenvalues (after discarding the trivial eigenvector $[1, 1, \dots, 1]^\top$ with eigenvalue zero). By multiplying both sides of this equation by $\mathbf{D}^{-1}$, the problem can be reformulated as finding the eigenvectors of $\mathbf{D}^{-1}\mathbf{V}$ corresponding to the *largest* eigenvectors:

$$\mathbf{D}^{-1}\mathbf{Va} = (1 - \lambda)\mathbf{a}.$$

The matrix $\mathbf{D}^{-1}\mathbf{V}$ is not symmetric and has rows normalized to 1. It can be interpreted as a diffusion operator on the graph, making Laplacian eigenmaps equivalent to *Diffusion maps* (Coifman and Lafon, 2006). Another equivalent way to rewrite it, is to define normalized Laplacian $\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ and solve an eigenvector problem $\mathbf{L}_{\text{norm}}\mathbf{b} = \lambda\mathbf{b}$, where $\mathbf{b} = \mathbf{D}^{1/2}\mathbf{a}$.

t-SNE without repulsion In the limit of $\rho \to \infty$, the repulsive term in the t-SNE gradient can be dropped, all $w_{ij} \to 1$, and hence the gradient descent update rule becomes (Linderman and Steinerberger, 2019)

$$\mathbf{y}_i^{t+1} = \mathbf{y}_i^t - \eta \sum_j v_{ij}(\mathbf{y}_i^t - \mathbf{y}_j^t),$$

where t indexes the iteration number and η is the learning rate (including all constant factors in the gradient). Denoting by $\mathbf{Y}$ the $n \times 2$ matrix of the embedding coordinates, this can be rewritten as

$$\begin{aligned}\mathbf{Y}^{t+1} &= (\mathbf{I} - \eta\mathbf{D} + \eta\mathbf{V})\mathbf{Y}^t \\ &= \mathbf{M}\mathbf{Y}^t.\end{aligned}$$

$\mathbf{M}$ is the transition matrix of this Markov chain (note that it is symmetric and its rows and columns sum to 1; its values are all non-negative for small enough η). According to the general theory of Markov chains, the largest eigenvalue of $\mathbf{M}$ is 1, and the corresponding eigenvector is $[1, 1, \dots, 1]^\top$, meaning that the embedding shrinks to a single point (as expected without repulsion). The slowest shrinking eigenvectors correspond to the next eigenvalues. This means that when ρ becomes very large, the embedding will resemble leading nontrivial eigenvectors of $\mathbf{M}$ (as ρ grows, the embedding will become smaller and smaller, but here we ignore its overall size; eigenvectors can have arbitrary length so overall scale of the embedding is not important here). This becomes equivalent to a power iteration algorithm. The eigenvectors of $\mathbf{M}$ are the same as of $\mathbf{L} = \mathbf{D} - \mathbf{V}$, which is the unnormalized graph Laplacian of the symmetric affinity matrix.

Note that this is not precisely what LE computes: as explained above, LE finds eigenvectors of the *normalized* graph Laplacian (von Luxburg et al., 2008). However, for t-SNE affinities, $\mathbf{D}$ is approximately proportional to the identity matrix, because $\mathbf{V}$ is obtained via symmetrization of directed affinities, and those have rows summing to 1 by construction. We can therefore expect that the leading eigenvectors of $\mathbf{L}$ and of $\mathbf{L}_{\text{norm}}$ are very close. We verified that for MNIST data they were almost exactly the same. A more abstract perspective is provided by von Luxburg et al. (2008): the unnormalized spectral embedding corresponds to minimizing a graph cut with respect to the number of vertices, whereas LE takes vertex degrees into account. Since all vertices in a kNN or perplexity-based graph have an almost equal degree, the eigenvectors of the normalized and unnormalized Laplacian will be very close.

Note also that nothing prevents different columns of $\mathbf{Y}$ to converge to the same leading eigenvector: each column independently follows its Markov chain. Indeed, we observed that for large enough values of ρ and large enough number of gradient descent iterations, the embedding collapsed to one dimension. This is the expected limiting behaviour when $\rho \to \infty$. However, for moderate values of ρ (as shown in this manuscript), this typically does not happen, and columns of $\mathbf{Y}$ resemble the two leading non-trivial eigenvectors of the Laplacian. The repulsive force prevents the embedding from collapsing to the leading Laplacian eigenvector. At the same time, a weak repulsive force will only be able to 'bring out' the second LE eigenvector. The stronger the contribution of repulsive forces, the more LE eigenvectors it would be able to 'bring out' (remember that the attractive force acts stronger on the higher eigenvectors).

Loss function of LE and quadratic constraint The original Laplacian eigenmaps paper (Belkin and Niyogi, 2002) motivates the eigenvector problem by considering

$$\mathcal{L}_{\mathrm{LE}} = \sum_{i,j} v_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 = 2\,\mathrm{Tr}(\mathbf{Y}^\top \mathbf{L}\mathbf{Y}).$$

This expression can be trivially minimized by setting all $\mathbf{y}_i = \mathbf{0}$, so the authors introduce a quadratic constraint $\mathbf{Y}^\top\mathbf{D}\mathbf{Y} = \mathbf{I}$, yielding the generalized eigenvector problem. We note that a different quadratic constraint $\mathbf{Y}^\top\mathbf{Y} = \mathbf{I}$ would yield a simple eigenvector problem for $\mathbf{L}$. In any case, the constraint plays the role of the repulsion in t-SNE framework.

Appendix B. Data Sources and Transcriptomic Data Preprocessing

The brain organoid data sets (Kanton et al., 2019) were downloaded from `https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-7552/` in form of UMI counts and metadata tables. The metadata table for the chimpanzee data set was taken from the supplementary materials of the original publication. We used gene counts mapped to the consensus genome, and selected all cells that passed quality control by the original authors (`in_FullLineage=TRUE` in metadata tables). For human organoid data, we only used cells from the `409b2` cell line, to simplify the analysis (the original publication combined cells from two cell lines and needed to perform batch correction).

The hydra data set (Figure A7; Siebert et al., 2019) was downloaded in form of UMI counts from `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE121617`.

The zebrafish data set (Figure A8; Wagner et al., 2018b) was downloaded in form of UMI counts from `https://kleintools.hms.harvard.edu/paper_websites/wagner_zebrafish_timecourse2018/WagnerScience2018.h5ad`.

The adult mouse cortex data set (Figure A9; Tasic et al., 2018) was downloaded in form of read counts from `http://celltypes.brain-map.org/api/v2/well_known_file_download/694413985` and `http://celltypes.brain-map.org/api/v2/well_known_file_download/694413179` for the VISp and ALM cortical areas, respectively. Only exon counts were used here. The cluster labels and cluster colors were retrieved from `http://celltypes.brain-map.org/rnaseq/mouse/v1-alm`.

To preprocess each data set, we selected the 1000 most variable genes using the procedure from Kobak and Berens (2019) with default parameters (for the mouse cortex data set we used 3000 genes and `threshold=32`; Kobak and Berens, 2019) and followed the preprocessing pipeline from the same paper: normalized all counts by cell sequencing depth (sum of gene counts in each cell), multiplied by the median cell depth (or 1 million in case of mouse cortex data), applied $\log_2(x+1)$ transformation, did PCA, and retained 50 leading PCs.

The data sets shown in Figures A4, A5, and A6 have been published explicitly to function as drop-in replacements for the hand-written MNIST data set. The data set variants that we used here all consist of a total $n = 70\,000$ images of 28×28 pixels, in 10 balanced classes. The input was preprocessed like the original MNIST data set by reducing the dimensions to 50 via PCA. Fashion and Kuzushiji MNIST were downloaded via OpenML with the keys `Fashion-MNIST` (`https://www.openml.org/d/40996`) and `Kuzushiji-MNIST` (`https://www.openml.org/d/41982`), respectively. Kannada MNIST was downloaded from `https://github.com/vinayprabhu/Kannada_MNIST`.

Appendix C. Supporting Experiments

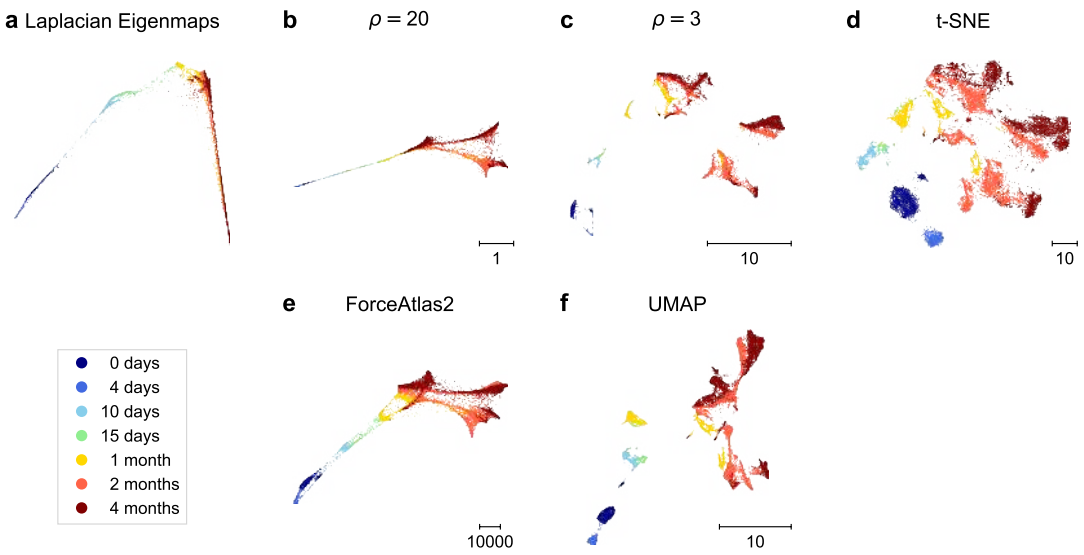

Figure A1: **Neighbor embeddings of the single-cell RNA-seq developmental data (human, high k).** The same as Figure 6, but LE, FA2, and UMAP used $k = 150$ (instead of our default $k = 15$), while t-SNE used perplexity 300 (instead of our default 30).

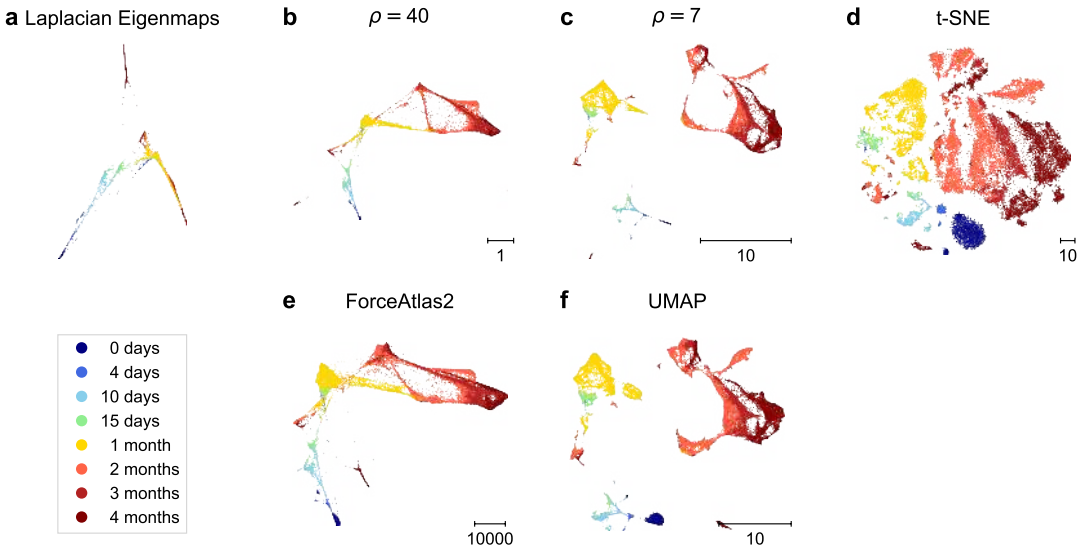

Figure A2: **Neighbor embeddings of the single-cell RNA-seq developmental data (chimpanzee).** Cells were sampled from chimpanzee brain organoids at eight time points between 0 days and 4 months into the development (Kanton et al., 2019). Sample size $n = 36\,884$. Data were reduced with PCA to 50 dimensions. See Appendix B for transcriptomic data preprocessing steps.

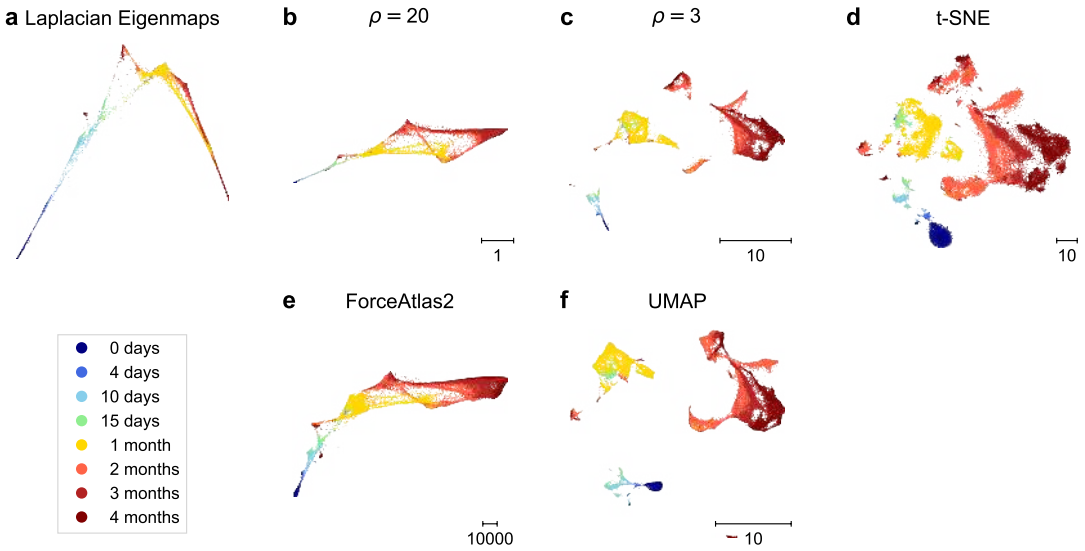

Figure A3: **Neighbor embeddings of the single-cell RNA-seq developmental data (chimpanzee, high k).** The same as Figure A2, but LE, FA2, and UMAP used $k = 150$ (instead of our default $k = 15$), while t-SNE used perplexity 300 (instead of our default 30).

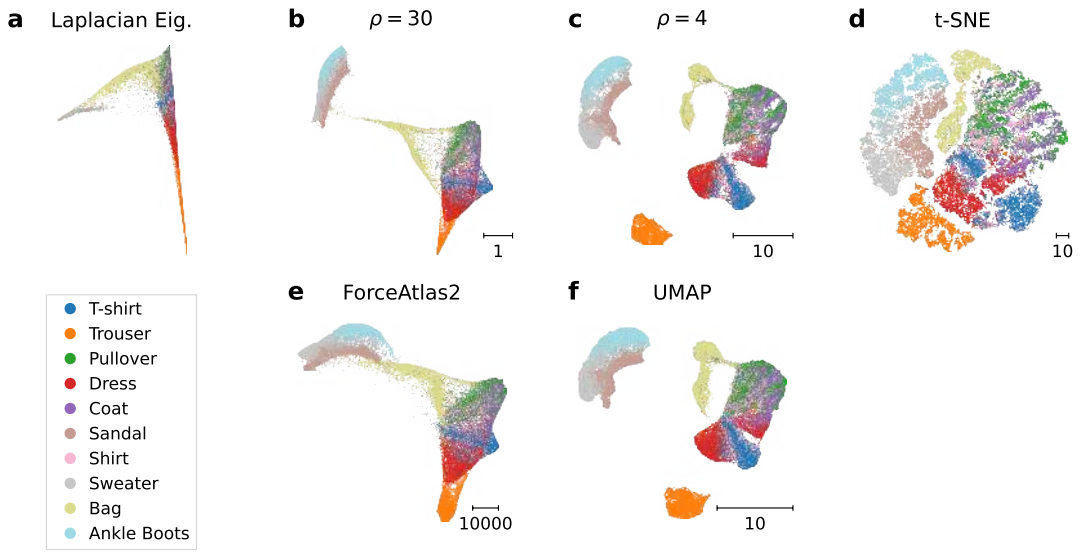

Figure A4: **Fashion MNIST data set (Xiao et al., 2017)**. Sample size $n = 70\,000$. Dimensionality was reduced to 50 with PCA. Colors correspond to 10 classes, see legend.

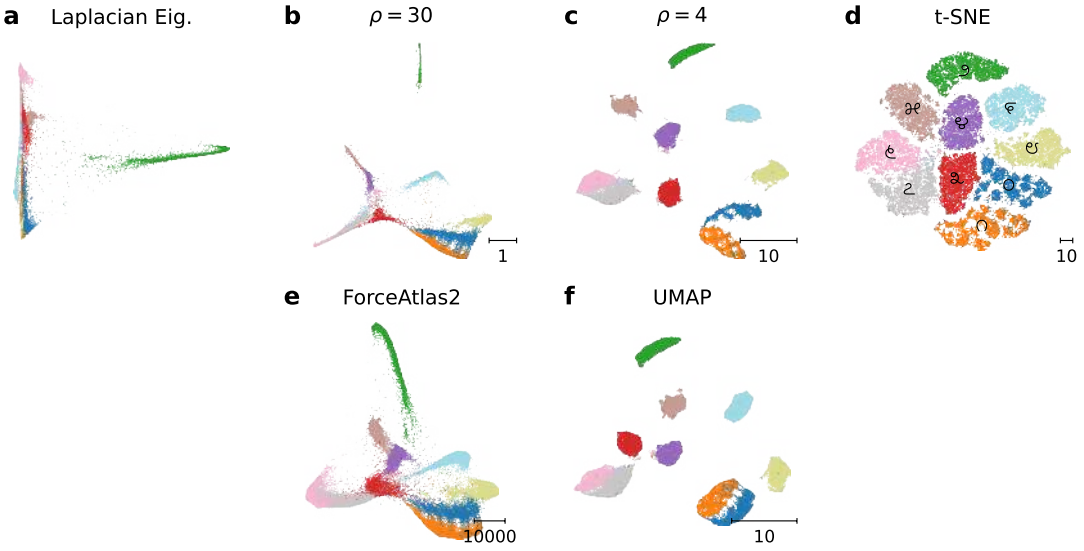

Figure A5: **Kannada MNIST data set (Prabhu, 2019).** Sample size $n = 70\,000$. Dimensionality was reduced to 50 with PCA. Colors correspond to 10 Kannada digits shown in panel (d).

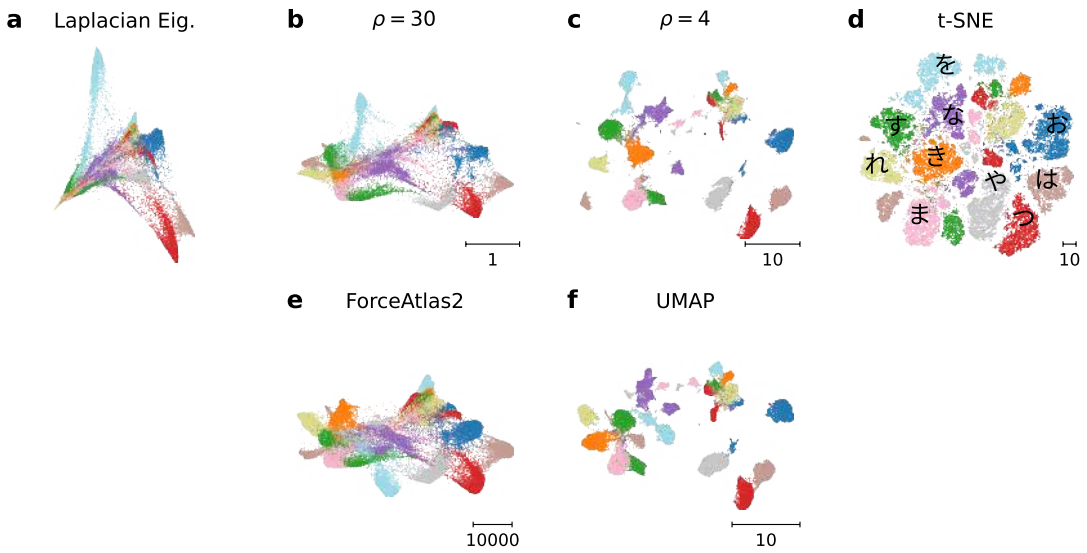

Figure A6: **Kuzushiji MNIST data set (Clanuwat et al., 2018).** Sample size $n = 70\,000$. Dimensionality was reduced to 50 with PCA. Colors correspond to 10 Kanji characters shown in panel (d).

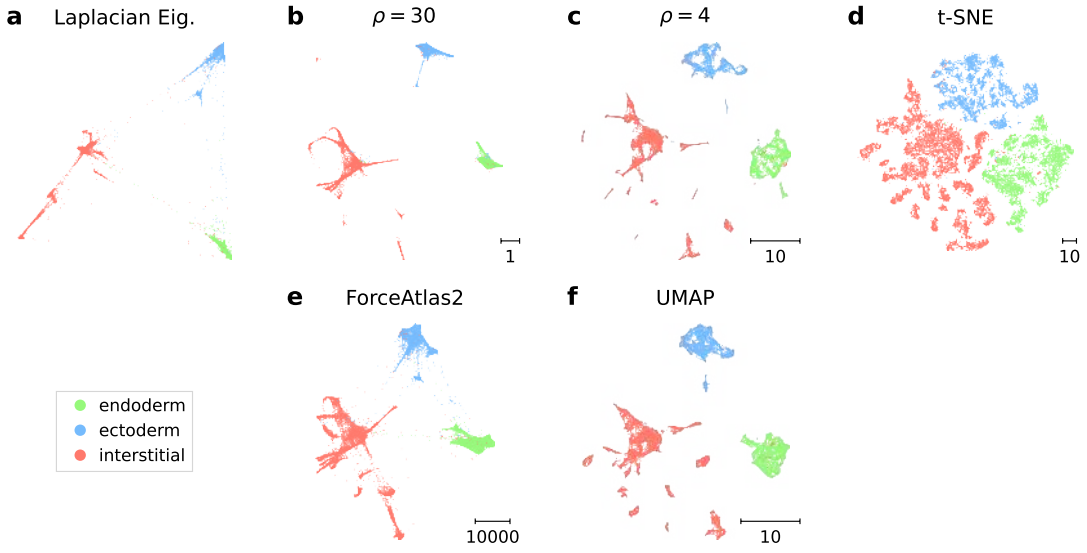

Figure A7: **Single-cell RNA-seq data of a hydra (Siebert et al., 2019).** Sample size $n = 24\,985$. Dimensionality was reduced to 50 with PCA. See Appendix B for transcriptomic data preprocessing steps. Color corresponds to cell classes.

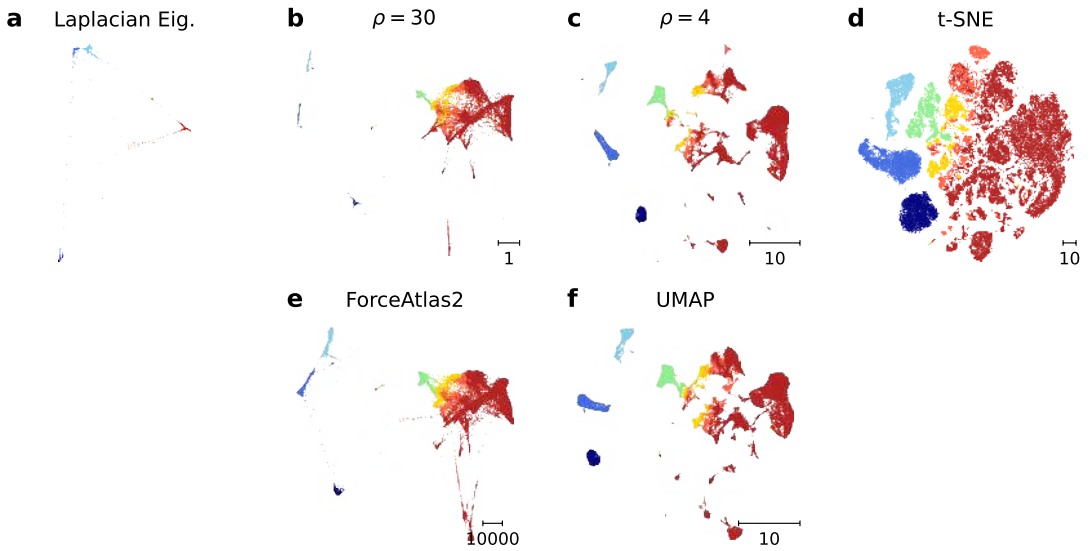

Figure A8: **Single-cell RNA-seq data of a zebrafish embryo (Wagner et al., 2018b).** Sample size $n = 63\,530$. Dimensionality was reduced to 50 with PCA. See Appendix B for transcriptomic data preprocessing steps. Color corresponds to the developmental stage, indicating the hours post fertilization (hpf).

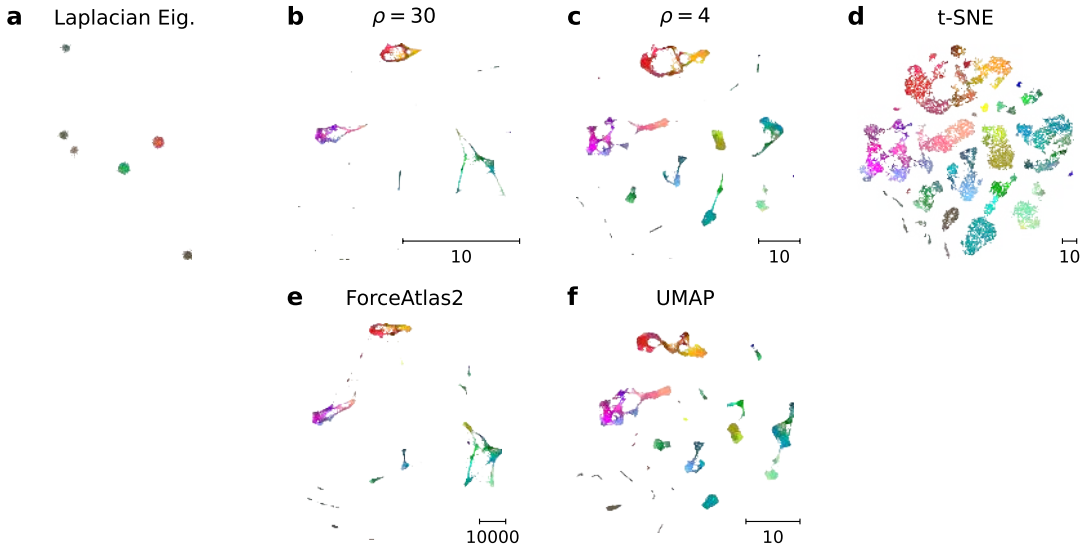

Figure A9: **Single-cell RNA-seq data of adult mouse cortex (Tasic et al., 2018).** Sample size $n = 23\,822$. Dimensionality was reduced to 50 with PCA. See Appendix B for transcriptomic data preprocessing steps. Colors are taken from the original publication (warm colors: inhibitory neurons; cold colors: excitatory neurons; grey/brown: non-neural cells). We added Gaussian noise to the LE embedding in panel (a) to make the clusters more visible. In this data set, the kNN graph is disconnected and has 6 components, resulting in 6 distinct points in the LE embedding.

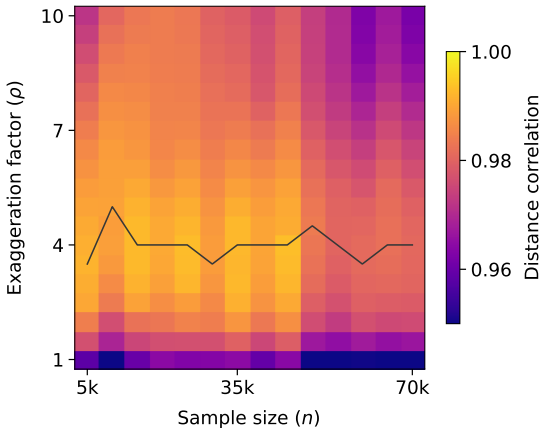

Figure A10: Distance correlations between t-SNE with $\rho \in [1, 10]$ and UMAP depending on the sample size, for MNIST subsets of size $n \in [5\,000, 70\,000]$. Black line indicates best matching ρ values.

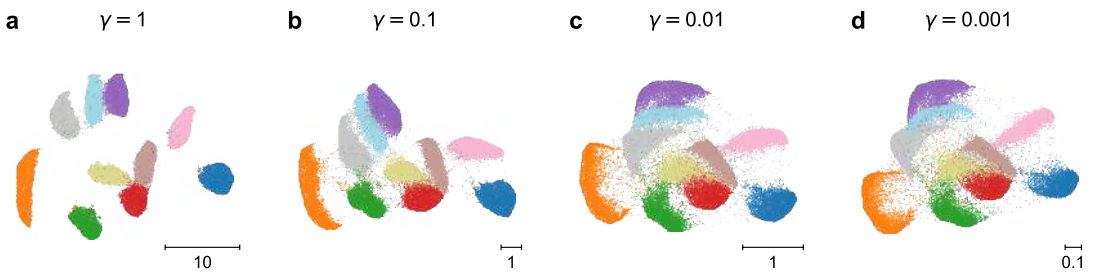

Figure A11: **Decreasing the repulsion in UMAP**. **(a)** UMAP embedding of MNIST with $\gamma = 1$ (default). **(b–d)** Decreasing γ produces the same effect as increasing the exaggeration ρ in t-SNE. Values $\gamma > 1$ are not shown because we could not achieve a well-converged embedding for $\gamma \gg 1$.

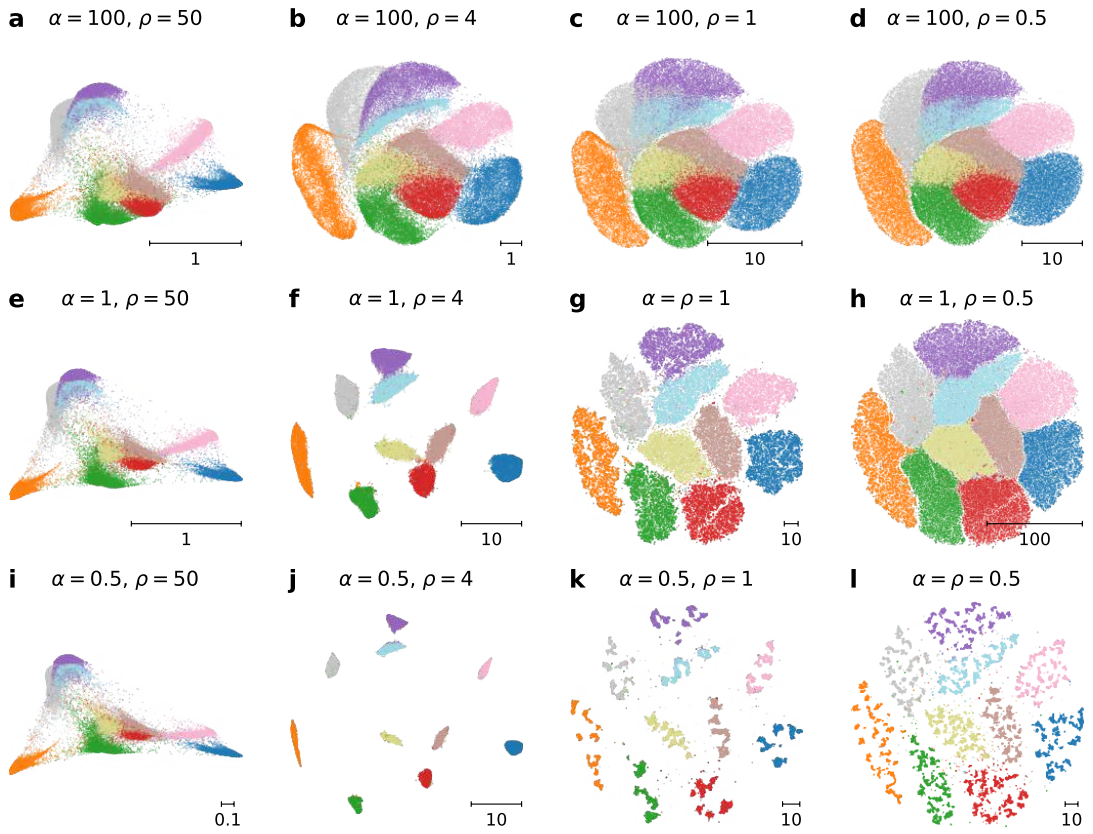

Figure A12: **Varying the tail-heaviness and exaggeration**. Changes in the layout for t-SNE of the MNIST data set when varying the tail-heaviness (Kobak et al., 2020; Yang et al., 2009) $\alpha \in \{100, 1, 0.5\}$ and the exaggeration factor $\rho \in \{50, 4, 1, 0.5\}$.

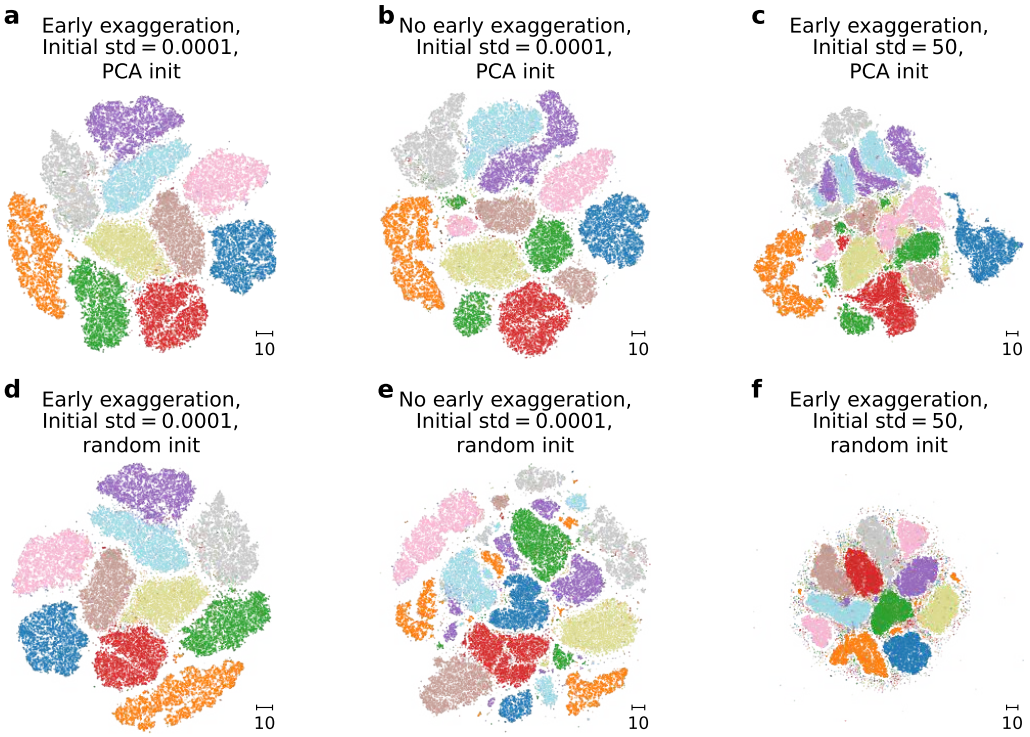

Figure A13: **The effect of early exaggeration on t-SNE**. **(a)** Default t-SNE embedding of MNIST. This uses early exaggeration and sets the standard deviation of PCA initialization to 0.0001. **(b)** T-SNE embedding without early exaggeration. This embedding is stuck in a suboptimal local minimum with some clusters split into multiple parts. **(c)** T-SNE embedding with early exaggeration, but with initial standard deviation set to 50. The attractive forces are too weak to pull the clusters together during the early exaggeration phase. **(d)** Default t-SNE with random initialization. The cluster structure is recovered, but the placement of the clusters is different from (a). **(e)** Same experiment as in (b), but with random initialization. The clusters are more fragmented due to less structure in the initialization and the lack of early exaggeration. **(f)** Same experiment as (c), but with random initialization. Here again, the attractive forces are too weak to pull the clusters together, and in addition there are points on the periphery that got stuck there due to the large initial distances.

UNSUPERVISED VISUALIZATION OF IMAGE DATASETS USING CONTRASTIVE LEARNING

Jan Niklas Böhm, Philipp Berens & Dmitry Kobak
University of Tübingen, Germany
{jan-niklas.boehm,philipp.berens,dmitry.kobak}@uni-tuebingen.de

ABSTRACT

Visualization methods based on the nearest neighbor graph, such as t-SNE or UMAP, are widely used for visualizing high-dimensional data. Yet, these approaches only produce meaningful results if the nearest neighbors themselves are meaningful. For images represented in pixel space this is not the case, as distances in pixel space are often not capturing our sense of similarity and therefore neighbors are not semantically close. This problem can be circumvented by self-supervised approaches based on contrastive learning, such as SimCLR, relying on data augmentation to generate implicit neighbors, but these methods do not produce two-dimensional embeddings suitable for visualization. Here, we present a new method, called t-SimCNE, for unsupervised visualization of image data. t-SimCNE combines ideas from contrastive learning and neighbor embeddings, and trains a parametric mapping from the high-dimensional pixel space into two dimensions. We show that the resulting 2D embeddings achieve classification accuracy comparable to the state-of-the-art high-dimensional SimCLR representations, thus faithfully capturing semantic relationships. Using t-SimCNE, we obtain informative visualizations of the CIFAR-10 and CIFAR-100 datasets, showing rich cluster structure and highlighting artifacts and outliers.

1 INTRODUCTION

As many research fields are producing ever larger and more complex datasets, data visualization methods have become important in many scientific and practical applications (Becht et al., 2019; Diaz-Papkovich et al., 2019; Kobak & Berens, 2019; Schmidt, 2018). Such methods allow a concise summary of the entire dataset, displaying a high-dimensional dataset as a 2D *embedding*. This low-dimensional representation is often convenient for data exploration, highlighting clusters and relationships between them. In practice, most useful are *neighbor embedding* methods, such as t-SNE (van der Maaten & Hinton, 2008) and UMAP (McInnes et al., 2018), that aim to preserve nearest neighbors from the high-dimensional space when optimizing the layout in 2D.

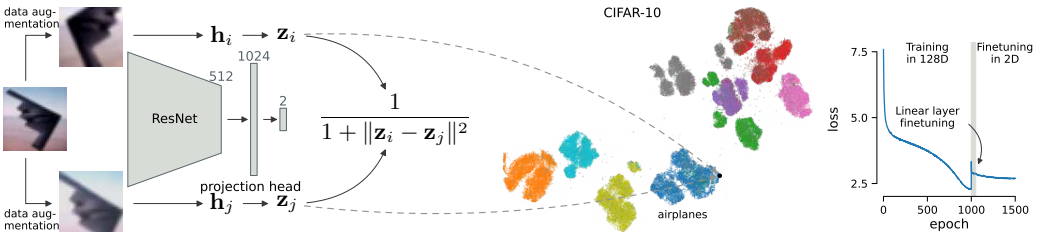

Figure 1: *Left: t-SimCNE.* Two augmentations of the same image are fed through the same ResNet and fully-connected projection head to get representations $\mathbf{z}_i$ and $\mathbf{z}_j$. The loss function pushes $\mathbf{z}_i$ and $\mathbf{z}_j$ together to maximize their Cauchy similarity. *Middle: Embedding of CIFAR-10.* The dashed arrows point to the locations of $\mathbf{z}_i$ and $\mathbf{z}_j$ from the left. *Right: Training loss.* The optimization consists of three stages: (1) pre-training with a 128D output for 1000 epochs; (2) fine-tuning only the 2D readout layer for 50 epochs; and (3) fine-tuning the entire network for 450 epochs.

Unfortunately, for image datasets, nearest neighbors computed using the Euclidean metric in pixel space are typically not worth preserving. Although t-SNE works well on very simple image datasets such as MNIST (van der Maaten & Hinton, 2008, Figure 2a), the approach fails when considering more natural image datasets such as CIFAR-10/100 (Supp. Fig. A.1). To create 2D embeddings for images, new visualization approaches are required, which use different notions of similarity.

Here, we provide such a method based on the contrastive learning framework. Contrastive learning is currently the state-of-the-art approach to unsupervised learning in computer vision (Hadsell et al., 2006). The contrastive learning method SimCLR (Chen et al., 2020) uses image transformations to create two views of each image and then optimizes a convolutional neural network so that the two views always stay close together in the resulting representation. While this method performs very well in benchmarks — such as linear or kNN classification accuracy, — the computed representation is typically high-dimensional (e.g. 128-dimensional), hence not suitable for visualization.

We extend the SimCLR framework to directly optimize a 2D embedding. Taking inspiration from t-SNE, we use the Euclidean distance and the Cauchy (t-distribution) kernel to measure similarity in 2D. While using 2D instead of 128D output may not seem like a big step, we show that optimizing the resulting architecture is challenging. We develop an efficient training strategy to overcome these challenges, and only then are able to achieve satisfactory visualizations. We call the resulting method t-SimCNE (Fig. 1) and show that it yields meaningful and useful embeddings of CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009).

Our code is available at `github.com/berenslab/t-simcne` (see `iclr2023` branch).

2 Related work

Neighbor embeddings (NE) have a rich history dating back to locally linear embedding (Roweis & Saul, 2000) and stochastic neighbor embedding (SNE; Hinton & Roweis, 2003). They became widely used after the introduction of the Cauchy kernel into the SNE framework (van der Maaten & Hinton, 2008) and after efficient approximations became available (van der Maaten, 2014; Linderman et al., 2019). A number of algorithms based on that framework, such as LargeVis (Tang et al., 2016), UMAP (McInnes et al., 2018), and TriMap (Amid & Warmuth, 2019) have been developed and got widespread adoption in recent years in a variety of application fields. All of them are closely related to SNE (Böhm et al., 2022; Damrich et al., 2023) and rely on the kNN graph of the data.

NE algorithms have been used to visualize latent representations of neural networks trained in a supervised setting (e.g. Karpathy, 2014; Mnih et al., 2015). This approach is, however, unsuitable for data exploration as it is supervised. NE algorithms can also be applied to an unsupervised (also known as self-supervised) representation of a dataset obtained with SimCLR, or to a representation obtained with a neural network pre-trained on a generic image classification task such as ImageNet (e.g. Narayan et al., 2015). The downside is that these approaches would not yield a parametric mapping to 2D. In this work, we are interested in an unsupervised but parametric mapping that allows embedding out-of-sample points. See Discussion for further considerations.

Conceptual similarity between SimCLR and t-SNE has recently been pointed out by Damrich et al. (2023). The authors suggest interpreting kNN graph edges as data augmentations, and show that t-SNE can also be optimized using the InfoNCE loss (van den Oord et al., 2018) used by SimCLR, and/or using a parametric mapping. Equivalently, one can think of SimCLR as a parametric SNE that samples edges from an unobservable neighbor graph. We were motivated by this connection when developing t-SimCNE. Further motivation comes from a recently described phenomenon called *dimensional collapse* (Jing et al., 2022; Tian, 2022), which suggests that there is redundant information in the output of SimCLR. Hence we reasoned that it should be possible to achieve a good representation even with drastically reduced output dimensionality.

Two closely related works appeared during preparation of this manuscript: Zang et al. (2022) suggest an architecture similar to SimCLR for 2D embeddings, but use a more complicated setup to impose 'local flatness' and, judging from their figures, obtain qualitatively worse embeddings of CIFAR datasets than we do (we were unable to quantitatively benchmark their method). Hu et al. (2023) suggest to use the Cauchy kernel in the SimCLR framework (calling it t-SimCLR), but in terms of 2D visualization, obtain worse results than we do (Hu et al., 2023, Fig. B.11 shows CIFAR-10, reported kNN accuracy 57% vs. our 89%).

3 CONTRASTIVE LEARNING FOR VISUALIZATION

3.1 SIMCLR OVERVIEW

SimCLR relies on creating two views of each data sample using random data augmentations (Fig. 1). For image data, this means that the image is randomly cropped, flipped, and so on. The method feeds a mini-batch containing pairs of augmented images through a ResNet to obtain a representation of the images. The parameters of the network are optimized such that the paired images are placed close to each other in the output space, while keeping all other images further apart.

In mathematical terms, the loss function used by SimCLR, known as InfoNCE loss (van den Oord et al., 2018), is defined as

$$\ell_{\text{SimCLR}}(i,j) = -\log \frac{\exp\big(\operatorname{sim}(\mathbf{z}_i,\mathbf{z}_j)/\tau\big)}{\sum_{k\neq i}^{2b}\exp\big(\operatorname{sim}(\mathbf{z}_i,\mathbf{z}_k)/\tau\big)} \tag{1}$$

$$= -\operatorname{sim}(\mathbf{z}_i,\mathbf{z}_j)/\tau + \log\sum_{k\neq i}^{2b}\exp\big(\operatorname{sim}(\mathbf{z}_i,\mathbf{z}_k)/\tau\big)\,. \tag{2}$$

Here, indices i and j correspond to two data augmentations of the same original image, and $\mathbf{z}$ denotes network output. For batch size b, there are $2b$ samples in the batch because each sample is augmented twice. The similarity function function used by SimCLR is the cosine similarity: $\operatorname{sim}(\mathbf{x},\mathbf{y}) = \mathbf{x}^\top\cdot\mathbf{y}/\big(\|\mathbf{x}\|\cdot\|\mathbf{y}\|\big)$. We follow Chen et al. (2020) in using $\tau = 0.5$.

The SimCLR cost function is intimately related to the SNE loss. Note that the cosine distance $\operatorname{cosdist}(\mathbf{x},\mathbf{y}) = 1-\operatorname{sim}(\mathbf{x},\mathbf{y})$ is equal to the half of the squared Euclidean distance between unit-normalized $\mathbf{x}$ and $\mathbf{y}$. Let $\tilde{\mathbf{a}} = \mathbf{a}/\|\mathbf{a}\|$. Then

$$\exp\big(\operatorname{sim}(\mathbf{z}_i,\mathbf{z}_j)/\tau\big) = \exp\left(\frac{1-\operatorname{cosdist}(\mathbf{z}_i,\mathbf{z}_j)}{\tau}\right) = \text{const}\cdot\exp\left(-\frac{\|\tilde{\mathbf{z}}_i-\tilde{\mathbf{z}}_j\|^2}{2\tau}\right), \tag{3}$$

which is precisely the expression used by SNE (a Gaussian kernel of the Euclidean distance) to measure similarity between embedding vectors (Hinton & Roweis, 2003). As shown by Damrich et al. (2023), SNE can also be optimized using the InfoNCE loss and mini-batch training; in the SNE setup, i and j are two points connected by a kNN graph edge.

The loss functions of both SimCLR and t-SNE give rise to an attractive force between similar points i and j, enforced by the InfoNCE numerator in Eq. (1), and a repulsion between all points, enforced by the denominator. Wang & Isola (2020) called this the *alignment* and the *uniformity* aspects of the loss. In the neighbor embedding literature, this is referred to as attractive and repulsive forces acting on the embedding points (Böhm et al., 2022; Damrich & Hamprecht, 2021; Wang et al., 2021).

3.2 t-SIMCNE LOSS FUNCTION FOR 2D VISUALIZATION

To achieve our goal of adapting SimCLR to create 2D embeddings of images, we reduce the dimensionality of the linear output layer from 128 to 2 (Fig. 1). This change, however, requires changing the similarity function as well, as using cosine similarity would effectively constrain the embedding to the unit circle S^1, which is not suitable for data visualization.

Instead, we remove the normalization from the last expression in Eq. (3), allowing the output $\mathbf{z}$ vectors to lie anywhere in $\mathbb{R}^2$. In other words, we replace the cosine distance with the Euclidean distance, which is a natural choice for measuring distance between points in a 2D embedding. While one could still use Gaussian kernel to transform Euclidean distance into a similarity as in Eq. (3), we follow t-SNE's example (van der Maaten & Hinton, 2008) and use the Cauchy (t-distribution with one degree of freedom) kernel instead, as the heavy-tailed Cauchy kernel reduces the 'crowding problem' that SNE had with the Gaussian kernel (van der Maaten & Hinton, 2008). We call the resulting method t-SimCNE.

Table 1: Model performance on the CIFAR-10 dataset. The columns are: model type (see text), dimensionality of Z, the total number of training epochs, linear classification accuracy in H, kNN classification accuracy in Z ($k = 15$; see Fig. A.3 for $k \in [1, 30]$), the final loss value, training time. Standard deviations correspond to three runs with different random seeds (for the loss, the standard deviation was always smaller than 0.05). Each experiment was run on a single GeForce RTX 2080 Ti GPU (the 5000 epoch experiment was run on a V100 GPU).

#	Model	dim	Epochs	Linear in H	kNN in Z	Loss	Time (hr.)
1	Cosine (SimCLR)	128	1000	$93.1 \pm 0.1\%$	$91.1 \pm 0.2\%$	5.8	12.0 ± 0.1
2	Cosine	3	1000	$87.6 \pm 0.3\%$	$74.0 \pm 1.4\%$	6.3	12.3 ± 0.4
3	Euclidean	128	1000	$90.7 \pm 0.3\%$	$90.1 \pm 0.2\%$	2.3	12.9 ± 1.6
4	Euclidean	2	1000	$84.6 \pm 0.3\%$	$80.0 \pm 0.2\%$	3.7	11.9 ± 0.1
5	Euclidean	2	5000	$88.7 \pm 0.2\%$	$87.0 \pm 0.5\%$	2.9	65.4 ± 0.2
6	Cos.→Euclidean	2	1500	$93.0 \pm 0.3\%$	$90.1 \pm 0.4\%$	3.2	17.5 ± 0.1
7	**Eucl.→Euclidean**	**2**	**1500**	$\mathbf{90.6 \pm 0.2\%}$	$\mathbf{89.4 \pm 0.4\%}$	**2.7**	$\mathbf{18.9 \pm 2.3}$
8	Eucl.→Euclidean	2	1000	$90.3 \pm 0.3\%$	$88.3 \pm 0.4\%$	2.9	11.8 ± 0
9	Eucl.→Euclidean	2	500	$88.7 \pm 0.1\%$	$85.8 \pm 0.1\%$	3.3	6.0 ± 0

We denote the Euclidean distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ as $d_{ij} = \|\mathbf{z}_i - \mathbf{z}_j\|$. The corresponding Cauchy similarity is $1/(1 + d_{ij}^2)$. With that, we define the t-SimCNE loss as

$$\ell_{t\text{-SimCNE}}(i,j) = -\log \frac{1/(1+d_{ij}^2)}{\sum_{k\neq i}^{2b} 1/(1+d_{ik}^2)} \tag{4}$$

$$= -\log \frac{1}{1+d_{ij}^2} + \log \sum_{k\neq i}^{2b} \frac{1}{1+d_{ik}^2}\,. \tag{5}$$

We will refer to the SimCLR loss as 'cosine' loss and to the t-SimCNE loss as 'Euclidean' loss. Note that the numerical loss values between them are not directly comparable, because the minimum of the Euclidean loss is zero, whereas the cosine loss, Eq. (2), is bounded from below by

$$\ell^*_{\text{SimCLR}}(i,j) = -1/\tau + \log\Big(\exp(1/\tau) + \sum_{k\neq i}^{2b-1} \exp(-1/\tau)\Big)$$

$$= -1/\tau + \log\big(\exp(1/\tau) + (2b-2)\cdot\exp(-1/\tau)\big)\,. \tag{6}$$

which is $\sim$3.65 for $b = 1024$ and $\tau = 0.5$.

3.3 Implementation, datasets, and performance metrics

We used our own PyTorch (Paszke et al., 2019, version 1.12.1) implementation of SimCLR. As the backbone, we used a ResNet18 (He et al., 2016), which has 512-dimensional output. We reduced the kernel size in the first convolutional layer of the ResNet18 from 7×7 to 3×3 as in Chen et al. (2020). For the fully-connected projection head we used one hidden ReLU layer with 1024 units, and linear output layer with 128 units. We optimized the network for 1000 epochs using SGD with momentum 0.9. The initial learning rate was $0.03 \cdot b/256 = 0.12$, with linear warm-up over ten epochs (from 0 to 0.12) and cosine annealing (Loshchilov & Hutter, 2017) down to 0 for the remaining epochs. We used batch size $b = 1024$ and the same set of data augmentations as in Chen et al. (2020).

We used CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009) for all our experiments. Each dataset consists of $n = 60\,000$ colored and labeled 32×32 images. CIFAR-10 has 10 classes (Fig. A.2), while CIFAR-100 has 100 classes grouped into 20 superclasses. We used the dataset classes `CIFAR10` and `CIFAR100` provided by the `torchvision` package.

To quantitatively assess the embedding quality, we not only report loss values but also the test-set kNN accuracy in the projection head output space Z as our main performance metric. Note that in the contrastive learning literature (e.g. Chen et al., 2020), representation quality is usually assessed

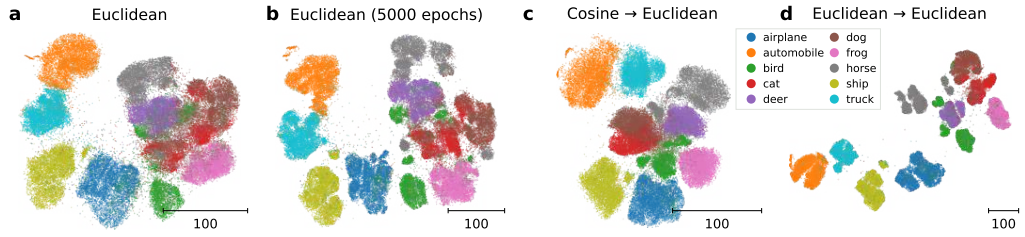

Figure 2: Different training strategies for t-SimCNE on CIFAR-10. *(a)* Optimizing the 2D Euclidean loss for 1000 epochs. *(b)* Optimizing the 2D Euclidean loss for 5000 epochs. *(c)* Pretraining with cosine loss in 128D and fine-tuning with Euclidean loss in 2D. *(d)* Pretraining with Euclidean loss in 128D and fine-tuning with Euclidean loss in 2D.

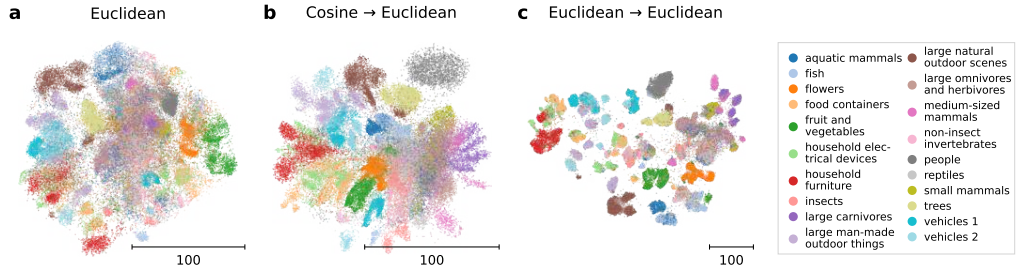

Figure 3: Different training strategies for t-SimCNE on CIFAR-100. The colors correspond to 20 superclasses and not to the fine-grained labels. *(a)* Optimizing the 2D Euclidean loss for 1000 epochs. *(b)* Pretraining with cosine loss in 128D and fine-tuning with Euclidean loss in 2D. *(c)* Pretraining with Euclidean loss in 128D and fine-tuning with Euclidean loss in 2D.

via linear classification accuracy in the ResNet output space H. In the Z space, we prefer to use the kNN classifier, as the 2D embedding can be considered good even if classes are separable but not linearly separable. For kNN accuracy, we used the scikit-learn (Pedregosa et al., 2011) implementation with $k = 15$ (any $k \in [1, 30]$ gave qualitatively the same results, Fig. A.3). For the linear accuracy, we used the `LogisticRegression` class from scikit-learn, with the SAGA solver (Defazio et al., 2014) and no penalty. For CIFAR-10 (CIFAR-100), our SimCLR implementation achieved 93% (67%) test-set linear accuracy in H (Tables 1 and A.1, line 1) which is state of the art for SimCLR with ResNet18 (e.g. Chen & He, 2021, Appendix D and da Costa et al., 2022, Table 1).

We heavily used Matplotlib 3.6.0 (Hunter, 2007), NumPy 1.23.1 (Harris et al., 2020), and openTSNE 0.6.2 (Poličar et al., 2019), which, in turn, uses Annoy (Bernhardsson, 2013).

3.4 t-SimCNE requires dimensionality annealing for optimal results

We used the setup described in the previous section to train t-SimCNE on the entire CIFAR-10 dataset. However, we found that naïve training of 2D t-SimCNE resulted in a suboptimal embedding layout (Fig. 2a) and achieved only 80% kNN accuracy (Table 1, line 4).

When we increased the number of training epochs from 1000 to 5000, the embedding improved (Fig. 2b), with kNN accuracy reaching 87% (Table 1, line 5). However, 5000 epochs took a long time (almost three GPU days) so this approach is not very practical. Moreover, we felt the embedding was still suboptimal, with many clusters seemingly unable to break apart from each other.

We noticed that in both cases above, the linear accuracy in H was markedly lower than with standard SimCLR (85% and 89% vs. 93%, cf. Table 1, lines 4, 5, and 1). We reasoned that it may be beneficial to pretrain the model with 128D output, achieving high-quality representation in H, and then fine-tune the model with the 2D output. This can be seen as 'dimensionality annealing'. Moving from 128D output to 2D requires changing the linear output layer in the projection head. For this, we

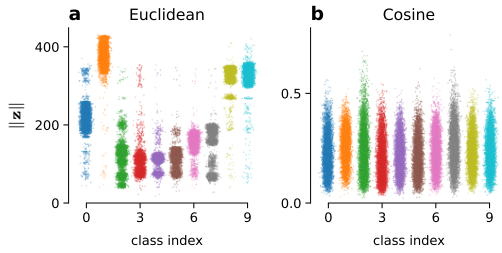

Figure 4: L_2 norms of the representation in 128-dimensional Z space after training on CIFAR-10 with the Euclidean loss (a) and with the cosine loss (b). Standard SimCLR uses the cosine loss. Colors as in Fig. 2. There was a similar difference in the H space, but less pronounced.

initialized the new 2D output layer randomly, froze the entire network apart from this layer, and trained the last layer for 50 epochs (learning rate 0.12, no warm-up, no annealing). This ensured that the new output layer was reasonably aligned with the rest of the projection head. Afterwards, the entire model was unfrozen and trained for another 450 epochs (initial learning rate 0.12/1000, warm-up for 10 epochs, cosine annealing down to 0).

We experimented with two different options for 128D pre-training, either using standard SimCLR with the cosine loss, or using the Euclidean loss. We found that using Euclidean similarity for pretraining resulted in the final t-SimCNE loss 2.7 (Table 1, line 7), vs. 3.2 when using the cosine similarity (Table 1, line 6). Euclidean loss also resulted in visually more pleasing embedding with crisp and well-separated clusters (Fig. 2d vs. Fig. 2c). As the difference in the final loss was large, we believe that this training strategy is the optimal one (even though the final kNN accuracy was slightly lower compared to the cosine pretraining: 89% vs. 90%).

We confirmed this conclusion by applying t-SimCNE to CIFAR-100 (Fig. 3). We found that without any pretraining, kNN accuracy on the superclass level was 50% (Fig. 3a); with SimCLR pretraining using cosine loss, it was 65% (Fig. 3b); and with Euclidean pretraining, it was 68% (Fig. 3c). On the level of classes, the kNN accuracy was 33%, 47%, and 51%, respectively (Table A.1, lines 4–6).

Even though the cosine similarity led to a higher quality of the H representation for both CIFAR-10 and CIFAR-100 (93% vs. 91% on CIFAR-10; Table 1, lines 1 vs. 3) compared to the Euclidean similarity; and even though the spherical constraint has theoretically appealing properties (Wang & Isola, 2020), the Euclidean similarity worked better for t-SimCNE pretraining. This may have to do with the norm distribution of the embedding vectors. Standard SimCLR with cosine loss has no discernible structure in the embedding norms (Fig. 4b), which is not surprising as the $\mathbf{z} \in \mathbb{R}^{128}$ vectors are normalized when computing the cosine similarity. But with Euclidean loss, the norms of $\mathbf{z}$ vectors strongly differ between classes (Fig. 4a). This leads to a reasonable 2D embedding even *before* the linear fine-tuning (Fig. A.4a and Fig. A.5a), and fine-tuning is able to work more effectively (Fig. A.4b,c and Fig. A.5b,c). Regarding dimensional collapse, it was less pronounced with Euclidean loss compared to the cosine loss (Fig. A.6).

3.5 Additional experiments

We explored the effect of training budget on the resulting representations. The training strategy described above totaled 1500 epochs and took almost 20 hours on our hardware. Even when the training length was drastically reduced, pretrained models achieved a better performance than end-to-end training in 2D. With the total budget of 500 epochs ($400 + 25 + 75$), the final kNN accuracy was 86% (Table 1, line 9); with 1000 epochs in total ($775+25+200$), it was 89% (Table 1, line 8) — better than end-to-end training for 5000 epochs. Visually, all three budgets resulted in qualitatively similar embeddings (Fig. A.7).

To test the stability of t-SimCNE, we repeated the training procedure three times with different random seeds. We found that while the exact layout and cluster arrangement differed between runs, qualitatively the embeddings were very similar (Fig. A.8).

SimCLR itself could in principle be used for data visualization if the output dimensionality is set to 3, so that the embedding lies on the 2-sphere S^2. However, we found that this setup only resulted in kNN accuracy of 74% (Table 1, line 2) and the embedding itself looked poor (Fig. A.9).

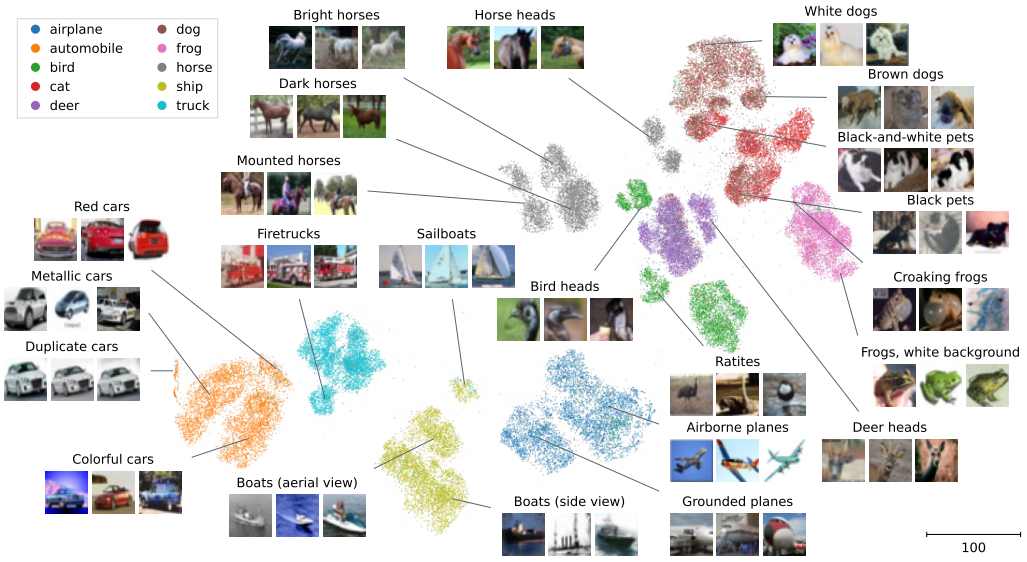

Figure 5: Annotated t-SimCNE embedding of the CIFAR-10 dataset. We manually annotated some of the prominent clusters by inspecting the images. Shown images are a random selection from the 15 nearest neighbors of the line tip.

4 Exploratory analysis shows the power of t-SimCNE

The ultimate goal of t-SimCNE is to be a tool for exploratory data analysis. In this section we demonstrate its power using the same CIFAR datasets as above.

4.1 t-SimCNE reveals subclass structure in CIFAR-10

We manually annotated the t-SimCNE embedding of CIFAR-10 for additional structure beyond the 10 originally defined classes (Fig. 5). We found that the embedding exhibited rich cluster structure, with many of the original classes (colors) splitting into several distinct clusters, or 'islands'.

We inspected the images within these clusters and found that they corresponded to meaningful semantic concepts. For example, within the 'ship' class, sailboats and other boat types differed clearly, such that sailboats formed an isolated cluster. Within the main 'ship' cluster, aerial view and sea level photographs varied systematically in their position in the 2D embedding. Similarly, the 'birds' class split into three well-separated islands: bird heads, ratites[1], and small birds. In the 'horse' class, horse heads and mounted horses formed separate clusters, and the remaining horse images were separated by color. Importantly, this within-class structure was not possible to infer from the original class labels alone, and we did not expect to find it when we started the project. Therefore, t-SimCNE helped us to discover additional latent structure in the dataset.

Although neighbor embedding methods are notoriously unreliable in preserving global structure of the data (Wattenberg et al., 2016), we found that t-SimCNE adequately represented some of the between-class structure. For example, the 'automobile' class was split into three main clusters: metallic cars, more colorful cars, and mostly bright red cars. The latter formed a separate cluster in the upper part of the orange island, next to one of the 'truck' clusters, consisting of bright red firetrucks.

One feature of the embedding that catches the eye, is a distinct, stripe-like orange cluster on the left (Fig. 5). Its oddly elongated and suspiciously dense shape suggests that it may be an artifact. Indeed, we found that this cluster consisted of near-duplicate pictures of the same three cars, such that three pictures appeared multiple times with only minor variations. A previous study focused specifically on near-duplicates in CIFAR datasets (Barz & Denzler, 2020) and identified 55 near-

[1]Large running birds such as ostriches and emus.

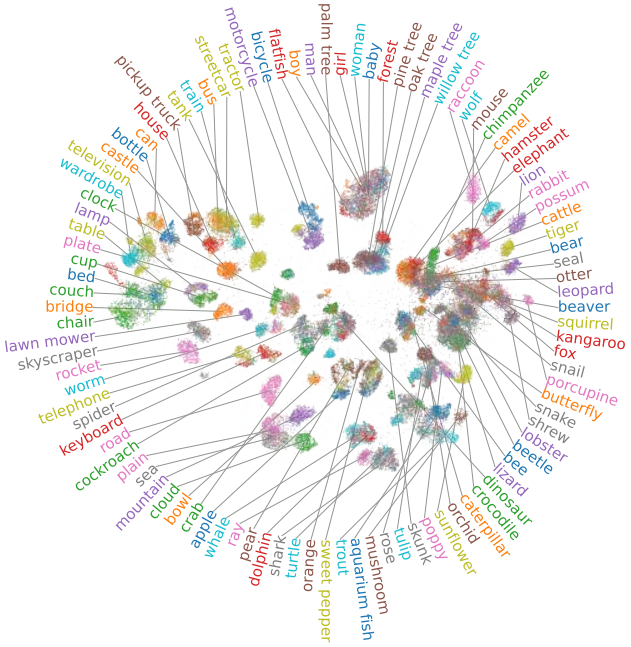

Figure 6: Annotated t-SimCNE embedding of the CIFAR-100 dataset. Class labels were positioned on the periphery in the order of $\mathrm{atan2}(y, x)$ where (x, y) is the mode of the kernel density estimate of embedding coordinates within each class.

duplicates in the 'automobile' class (Barz & Denzler, 2020, Table 1). With our exploratory analysis we found 163 almost identical images in the small orange island (Fig. A.10). Another study on labeling errors in standard computer vision datasets including CIFAR-10 missed the near-duplicates (Northcutt et al., 2021). This suggests that t-SimCNE is more intuitive and more sensitive than other, more conventional, ways to explore image datasets, and can be useful for quality control in actual practice.

The t-SimCNE visualization also highlighted what parts of the dataset SimCLR-like models struggle with. For example, the model seemed to be confused between dogs and cats as parts of these two classes were merged into one (upper-right corner: black-and-white pets, black pets). It is less noticeable, but within the cluster of airborne planes there were some images of birds. Those were pictures of birds flying in the sky, which exhibit a lot of visual similarity to airborne planes, hence the model's confusion.

While overall the embedding exhibited a pronounced cluster structure, there were a few outlier points located amidst the white space, which did not seem to belong to any one cluster. We investigated these images and found some obvious examples of outlier images. For example, one image, labeled as 'ship', depicts a person riding a jet ski (Fig. A.11c). Another image is labeled as 'truck', but we were unable to make out what was actually depicted (Fig. A.11f). These examples would be hard to find within a dataset of 60 000 images, but they clearly stick out in the 2D t-SimCNE visualization.

4.2 t-SimCNE elucidates inter-class relations in CIFAR-100

Similar observations apply to the t-SimCNE embedding of the CIFAR-100 dataset with its much more heterogeneous structure compared to CIFAR-10 (Fig. 6). In terms of the large-scale organisation, different animal species were placed mostly on the right side of the embedding, while man-made objects could be found on the left side. In terms of the fine-scale organization, the superclasses were separated very well in the embedding (Fig. 3c), as were many individual classes within a single superclass. For example, in the superclass 'flowers', the embedding clearly distinguished between orchids, sunflowers, roses, tulips, and poppy. Similarly, in the superclass 'large natural outdoor scenes', images of plain, sea, mountain, cloud, and forest were mostly non-overlapping.

On the other hand, in some other superclasses, individual classes were mixed together. For example, in the superclass 'people' (the topmost cluster), all five classes (boy, man, girl, woman, and baby) were intermingled, suggesting that the model struggled in this region of the feature space. Curiously, the same cluster also contained the 'flatfish' class. We found that the 'flatfish' images often show fishermen holding their catch, explaining their appearance in the 'people' island (Fig. A.12).

There were further examples of classes that t-SimCNE positioned next to a seemingly wrong superclass. The 'forest' class was separated from other members of the 'large natural outdoor scenes' superclass and placed next to the pine, oak, maple, and willow tree classes. This makes sense, as forest is close to trees both semantically and in terms of the image statistics.

These examples suggest that the class and the superclass definitions in the CIFAR-100 dataset do not always form a reliable ontology capturing all aspects of an image.

5 Discussion

We developed a new self-supervised method, t-SimCNE, to visualize image datasets in 2D, and showed that it yields meaningful and interpretable visualizations of CIFAR-10 (Fig. 5) and CIFAR-100 (Fig. 6) datasets (kNN classification accuracy 89% and 51% respectively). We are not aware of any other unsupervised parametric methods that could yield comparably good 2D embeddings:

- t-SNE embeddings in pixel space are typically very poor as the Euclidean metric is not meaningful for image data (Fig. A.1). Any other neighbor embedding method, be it UMAP (McInnes et al., 2018), LargeVis (Tang et al., 2016), TriMap (Amid & Warmuth, 2019), etc., or any of their parametric versions (van der Maaten, 2009; Cho et al., 2018; Ding et al., 2018; Szubert et al., 2019; Kalantidis et al., 2022; Sainburg et al., 2021; Damrich et al., 2023), would have the same issue.
- t-SNE embeddings of the trained SimCLR representation are reasonably good (Fig. A.13), however (i) we found them qualitatively to be less interpretable than the t-SimCNE embeddings, as outliers, artifacts, and subclass structure were less noticeable; and (ii) they are not parametric, i.e. do not allow positioning out-of-sample images into an existing embedding, which is often relevant in applications.
- SimCLR with 3D output yields an embedding on 2-sphere S^2 but it was outperformed by t-SimCNE in terms of kNN accuracy and visual class separation, and also is cumbersome to visualize on a 2D plane as it requires a map projection (Fig. A.9).
- t-SNE embeddings of a trained ResNet-based classifier representation (Fig. A.14) are not unsupervised and not useful for data exploration, as they do not show much structure within each of the pre-defined classes.
- Finally, using a readily available off-the-shelf ResNet-based classifier pretrained on ImageNet and visualizing its representation with t-SNE (or UMAP, TriMap, etc.) is a very fast alternative approach to t-SimCNE. It is unsupervised in a sense that it does not use dataset labels (but is based on supervised ImageNet pretraining). While it is, again, not a parametric method, we found the resulting embeddings to have good kNN accuracy, especially for larger ResNets, such as ResNet-152 (Figs. A.15 and A.16). As it is much faster than t-SimCNE, this approach can make sense for initial exploration of the data, however we found that it is strongly outperformed by t-SimCNE in terms of visual class separation, as measured by the clustering-based adjusted Rand index (Fig. A.17).

Optimizing our architecture with 2D output was challenging and required a carefully tuned training setup. Our training scheme yielded good 2D embeddings, but led to the worse representation in the H space compared to standard SimCLR. This suggests that it may be possible to further improve the training protocol and the fine-tuning schedule, which we leave for future work.

In the future it will be interesting to apply t-SimCNE to larger and more complex datasets, such as ImageNet (Russakovsky et al., 2015) or its subsets, e.g. Tiny ImageNet. It will also be interesting to apply it to real-life scientific data, e.g. in a biomedical domain. Our hope is that methods like t-SimCNE will be able to aid scientific exploration and discovery.

APPENDIX A SUPPLEMENTARY TABLES AND FIGURES

Table A.1: Model performance on the CIFAR-100 dataset. See Table 1 for description.

#	Model	dim	Epochs	Linear in H	kNN in Z	Loss	Time (hr.)
1	Cosine (SimCLR)	128	1000	$66.7 \pm 0.5\%$	$58.0 \pm 0.2\%$	5.8	12.0 ± 0.1
2	Cosine	3	1000	$52.4 \pm 0.1\%$	$15.7 \pm 0.2\%$	6.3	11.9 ± 0.0
3	Euclidean	128	1000	$59.5 \pm 0.2\%$	$54.5 \pm 0.1\%$	2.5	11.9 ± 0.1
4	Euclidean	2	1000	$49.4 \pm 0.8\%$	$33.2 \pm 0.2\%$	4.3	11.7 ± 0.1
5	Cos.→Euclidean	2	1500	$65.1 \pm 0.1\%$	$46.7 \pm 0.6\%$	3.6	17.5 ± 0.2
6	**Eucl.→Euclidean**	**2**	**1500**	$\mathbf{59.3 \pm 0.4\%}$	$\mathbf{51.1 \pm 0.3\%}$	**2.9**	$\mathbf{17.4 \pm 0.1}$

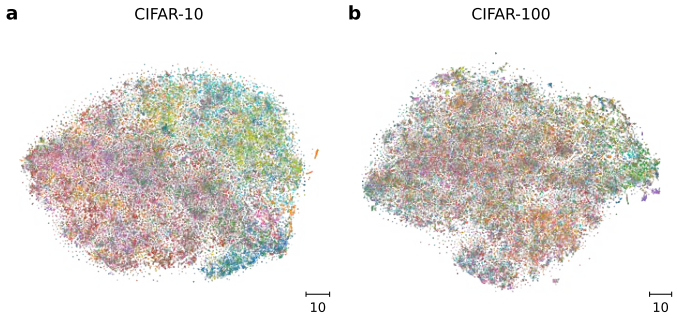

Figure A.1: t-SNE embedding of the CIFAR-10 (a) and CIFAR-100 (b) datasets in pixel space. Each image is 32×32 with three color channels, corresponding to vectors of dimensionality $32 \cdot 32 \cdot 3 = 3072$. They were embedded using openTSNE (Poličar et al., 2019) with default parameters. Colors correspond to classes as in Figs. 2 and 6. kNN accuracies were 33% and 13% (CIFAR-100 classes) in the embedding, and 33% and 15% when measured directly in the 3072-dimensional pixel space. For reference, our method t-SimCNE obtained 89% and 51% respectively.

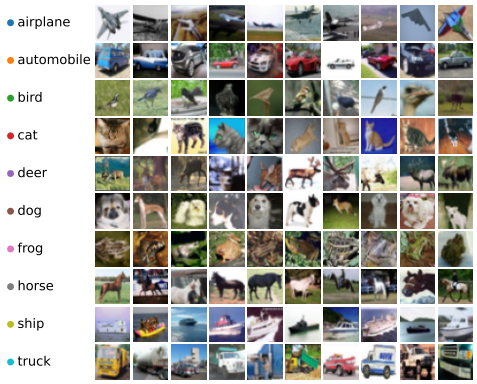

Figure A.2: Ten random images from each class of the CIFAR-10 dataset.

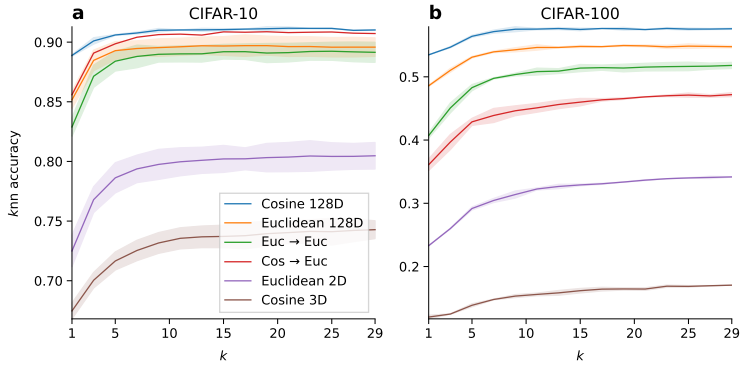

Figure A.3: kNN accuracy in the Z space of various models listed in Tables 1 and A.1 as a function of k. Shading shows standard deviations across three runs. (a) CIFAR-10. (b) CIFAR-100.

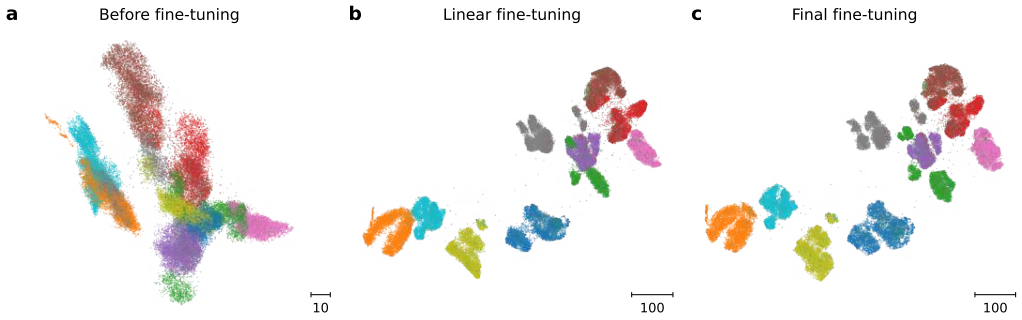

Figure A.4: Different stages of fine-tuning when training t-SimCNE on CIFAR-10. *(a)* The embedding after random initialization of the 2D output layer. *(b)* The embedding after training the output layer for 50 epochs, while the rest of the network stays frozen. *(c)* The final embedding after fine-tuning the entire network for 450 epochs. Colors as in Fig. 2.

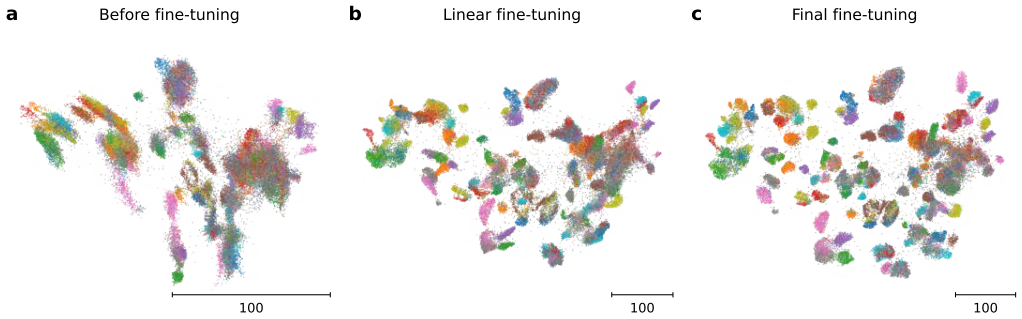

Figure A.5: Different stages of fine-tuning when training t-SimCNE on CIFAR-100. *(a)* The embedding after random initialization of the 2D output layer. *(b)* The embedding after training the output layer for 50 epochs, while the rest of the network stays frozen. *(c)* The final embedding after fine-tuning the entire network for 450 epochs. Colors as in Fig. 6.

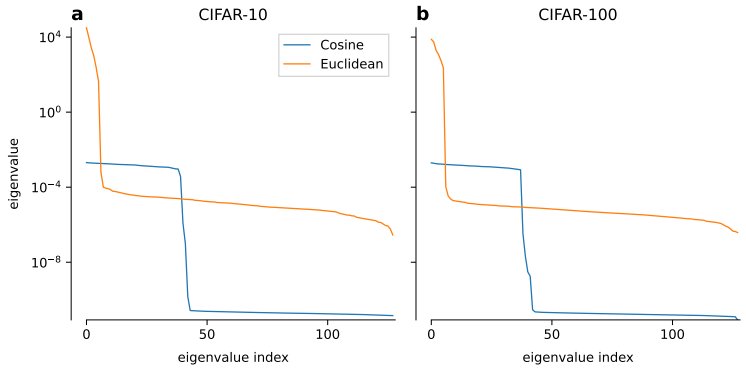

Figure A.6: Eigenvalue spectrum of the covariance matrix in the Z space for 128-dimensional models trained with the cosine and with the Euclidean losses. (a) CIFAR-10. (b) CIFAR-100.

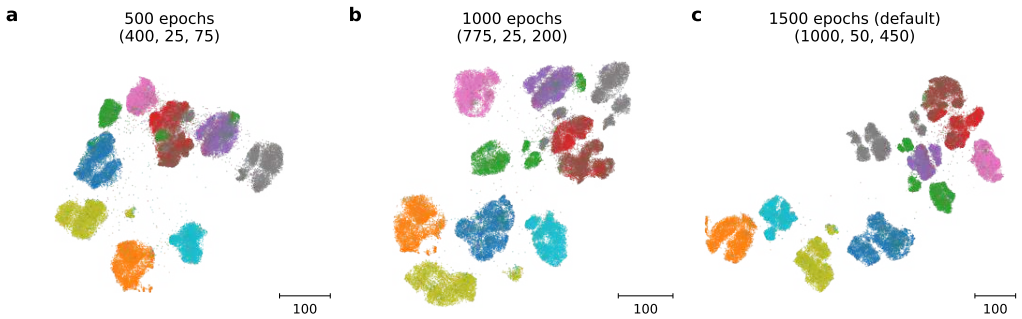

Figure A.7: t-SimCNE embeddings of CIFAR-10 dataset with different runtime budgets. *(a)* 500 epochs in total (400 for pretraining, 25 for the readout training, 75 for fine-tuning). *(b)* 1000 epochs in total (775 for pretraining, 25 for the readout training, 200 for fine-tuning). *(a)* 1500 epochs in total (1000 for pretraining, 50 for the readout training, 450 for fine-tuning). Embeddings were flipped along the x- and/or y-axis to maximize the alignment. Colors as in Fig. 2.

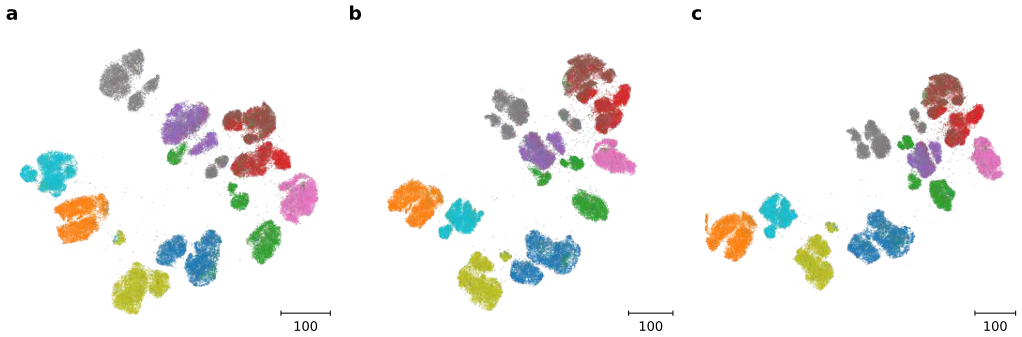

Figure A.8: t-SimCNE embeddings of CIFAR-10 dataset, trained from scratch with three different random seeds. Embeddings were flipped along the x- and/or y-axis to maximize the alignment. Colors as in Fig. 2.

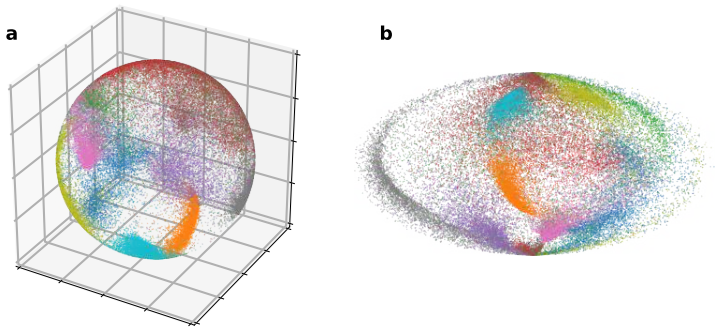

Figure A.9: 3D visualization of CIFAR-10 obtained with SimCLR with 3D output. *(a)* Embedding vectors on the unit sphere. *(b)* Hammer projection (equal-area) to two dimensions. Colors as in Fig. 2.

Figure A.10: Near-duplicates in CIFAR-10. A zoom-in into the t-SimCNE embedding. There are three distinct images, duplicated many times with small variations: (1) front view of a car, (2) rear view of a car, and (3) side view of a car. Colors as in Fig. 2. For this image the aspect ratio of the embedding has not been preserved.

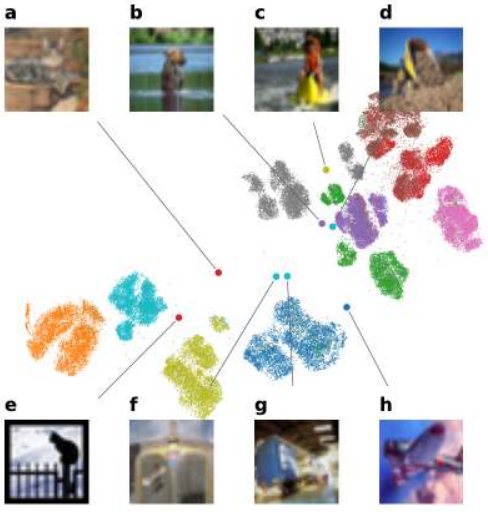

Figure A.11: Outliers in the t-SimCNE embedding of CIFAR-10. *(a)* A cat, almost blurred into the background. *(b)* An animal, labeled as 'deer', emerging from the water. *(c)* A person on a jet ski, labeled as 'ship'. *(d)* Mud in the foreground, and a truck unloading it in the background. *(e)* A cat silhouette. *(f)* Unknown, labeled as 'truck'. *(g)* Unknown, labeled as 'truck'. *(h)* Unknown, labeled as 'plane'. Colors as in Fig. 2.

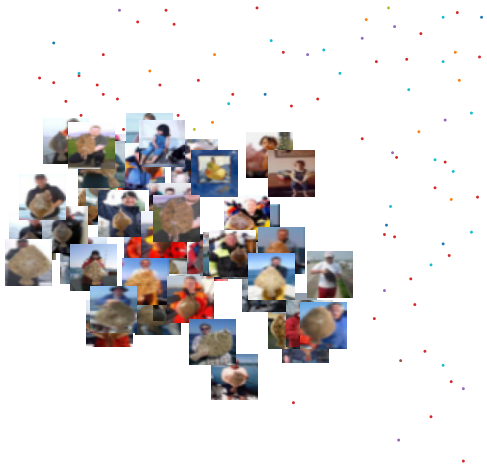

Figure A.12: Images of the class 'flatfish' in CIFAR-100 located next to the 'people' superclass in t-SimCNE embedding. Colors as in Fig. 6. For this image the aspect ratio of the embedding has not been preserved.

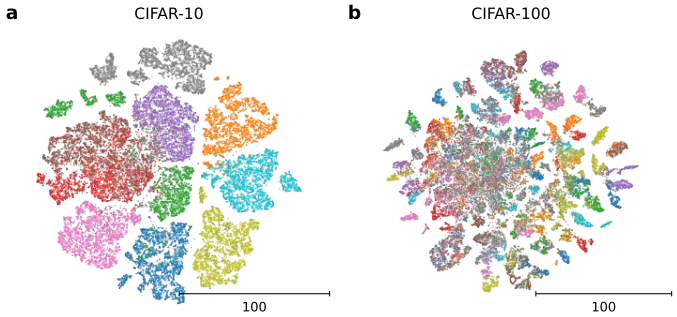

Figure A.13: t-SNE of the 512-dimensional standard (with cosine loss) SimCLR representation of CIFAR-10 (a) and CIFAR-100 (b). We used openTSNE (Poličar et al., 2019) with default parameters. Colors correspond to classes as in Figs. 2 and 6. The kNN accuracies were 91% for CIFAR-10 (a) and 57% for CIFAR-100 classes (b).

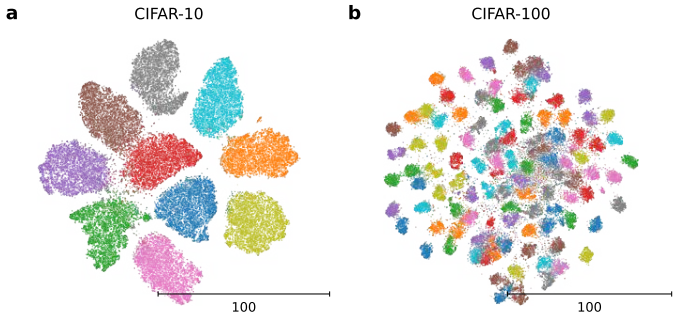

Figure A.14: t-SNE embedding of the CIFAR-10 (a) and CIFAR-100 (b) representations obtained with a ResNet-18 trained with the supervised cross-entropy loss on the same CIFAR data. The network architecture was the same as for the SimCLR experiments, but the projection head was mapping to ten (for CIFAR-10) or 100 (for CIFAR-100) softmax dimensions. We trained each network for 100 epochs, with initial learning rate 0.1 and linear annealing down to 0. We used the same set of data augmentations as we used for contrastive learning (Chen et al., 2020). Only training images were used for training; the test set accuracy after training was 92.8% for CIFAR-10 and 72.4% for CIFAR-100 classes. The ResNet output layer H (see Fig. 1) was used as the input to t-SNE (both training and test images together). We used openTSNE (Poličar et al., 2019) with default parameters. Colors correspond to classes as in Figs. 2 and 6.

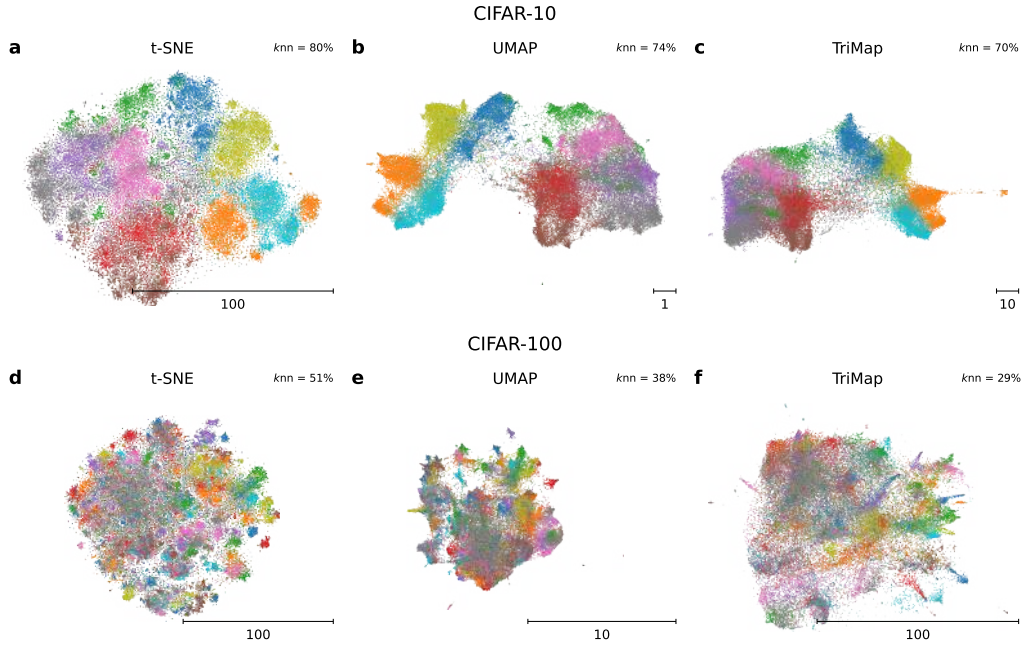

Figure A.15: Embeddings of the CIFAR-10 (a–c) and CIFAR-100 (d–f) representations obtained with a ResNet-18, pretrained on the ImageNet classification task. We used pretrained weights available in PyTorch (Paszke et al., 2019). The input images were first resized to 256×256 pixels and then center-cropped to 224×224 pixels, following He et al. (2016). The ResNet output layer H (see Fig. 1) was used as the input to the visualization algorithms. We used openTSNE (Poličar et al., 2019), UMAP (McInnes et al., 2018), and TriMap (Amid & Warmuth, 2019) with default parameters. Colors correspond to classes as in Figs. 2 and 6. kNN classification accuracies are indicated in the corner of each panel (for CIFAR-100, it is the accuracy on the class level).

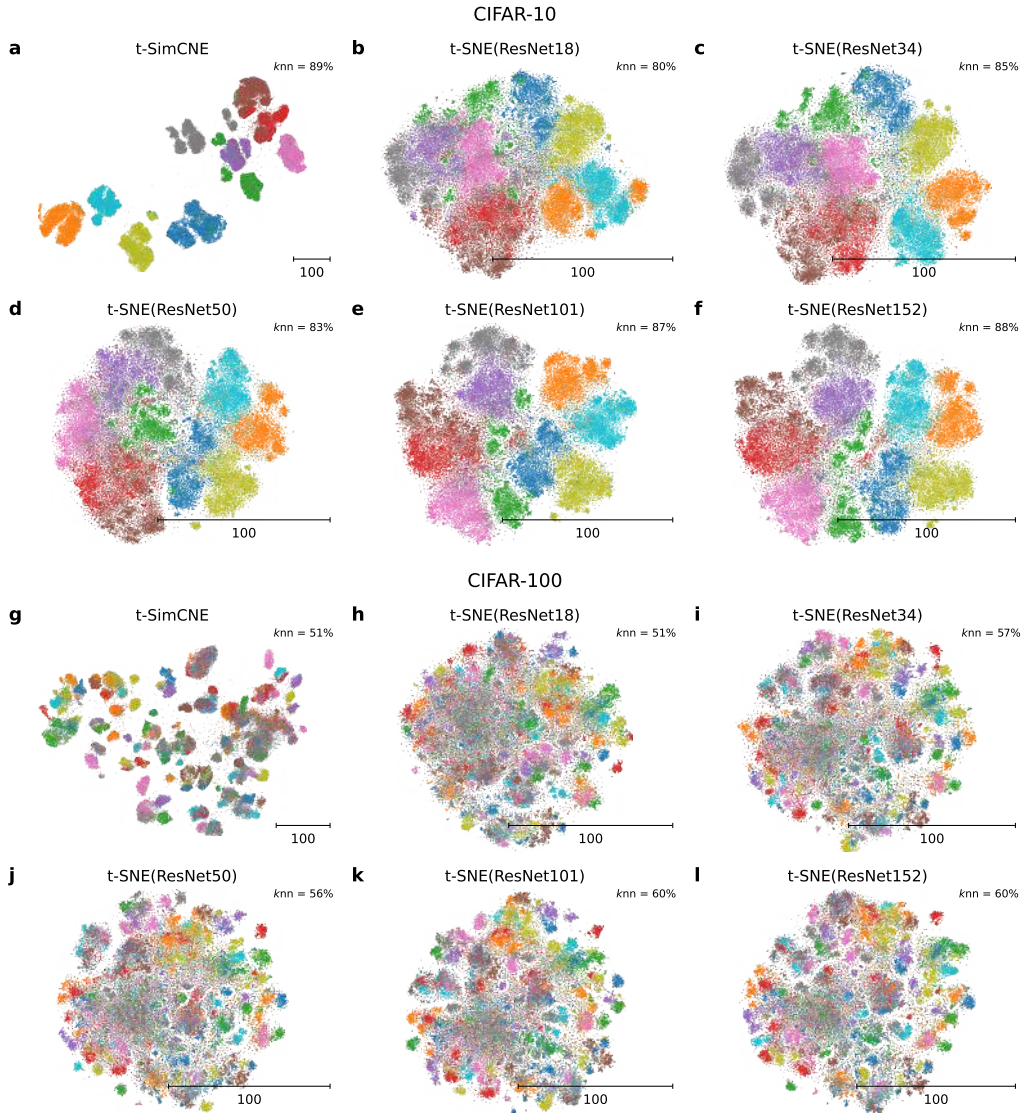

Figure A.16: Visualizations of t-SimCNE and of t-SNE applied to the representation obtained using ImageNet-pretrained ResNet networks of different size. *(a–f)* CIFAR-10. *(g–l)* CIFAR-100.

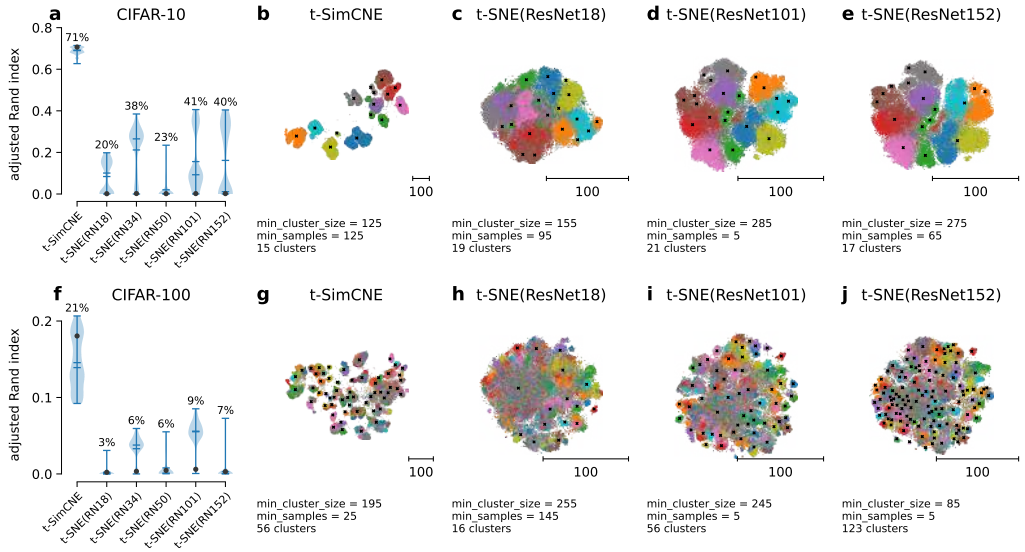

Figure A.17: Class separability comparison between t-SimCNE and t-SNE applied to the representation obtained using ImageNet-pretrained ResNet networks. We used clustering to assess how strongly classes were separated from each other in 2D. *(a)* The adjusted Rand index (ARI; Hubert & Arabie, 1985) between the clusters and the class labels on CIFAR-10. The clusters were obtained with HDBSCAN (McInnes et al., 2017) using the parameter grid $\texttt{min\_cluster\_size} \in \{5, 15, \ldots, 295\} \times \texttt{min\_samples} \in \{5, 15, \ldots, 145\}$ with the constraint $\texttt{min\_samples} \leq \texttt{min\_cluster\_size}$. The violin plots (Hintze & Nelson, 1998) show the distribution of the ARI values across different HDBSCAN parameter combinations, with the best value indicated above. The black dot denotes the ARI of the default HDBSCAN parameters ($\texttt{min\_cluster\_size} = \texttt{min\_samples} = 5$). We used ResNet weights that are available in PyTorch. *(b–e)* The 2D visualizations of CIFAR-10 using t-SimCNE and t-SNE of pretrained ResNet representations. The cluster centroids corresponding to the best clustering (in terms of ARI) are shown as black cross marks, and the corresponding HDBSCAN parameters are given below. *(f–j)* The same for CIFAR-100.

Node Embeddings via Neighbor Embeddings

Jan Niklas Böhm [*1] **Marius Keute** [*1] **Alica Guzmán** [1] **Sebastian Damrich** [1] **Andrew Draganov** [2] **Dmitry Kobak** [1]

Abstract

Graph layouts and node embeddings are two distinct paradigms for non-parametric graph representation learning. In the former, nodes are embedded into 2D space for visualization purposes. In the latter, nodes are embedded into a high-dimensional vector space for downstream processing. State-of-the-art algorithms for these two paradigms, force-directed layouts and random-walk-based contrastive learning (such as DeepWalk and node2vec), have little in common. In this work, we show that both paradigms can be approached with a single coherent framework based on established neighbor embedding methods. Specifically, we introduce *graph t-SNE*, a neighbor embedding method for two-dimensional graph layouts, and *graph CNE*, a contrastive neighbor embedding method that produces high-dimensional node representations by optimizing the InfoNCE objective. We show that both graph t-SNE and graph CNE strongly outperform state-of-the-art algorithms in terms of local structure preservation, while being conceptually simpler.

1. Introduction

Many real-world datasets, ranging from molecule structures to citation networks, come in the form of graphs. Graphs are abstract objects consisting of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ which do not inherently belong to any specific metric space. Therefore, the field of graph representation learning has emerged with the goal of embedding the nodes into a metric space $\mathbb{R}^d$ so that neighborhoods are well-preserved. Traditionally, a distinction has been made between (i) *graph layout* (or graph drawing) methods which embed nodes into $\mathbb{R}^2$ for visualization purposes and (ii) *node embedding* methods like DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) which embed nodes into higher-dimensional spaces more suitable for downstream analysis, such as classification or clustering. Graph layout methods typically obtain the embeddings by pulling together connected nodes, whereas node embedding methods are typically based on contrastive learning and random-walk notions of node similarity. Both approaches are *non-parametric* and do not require any node features.

Recent work has revealed connections between graph layouts and dimensionality reduction techniques like t-SNE (van der Maaten & Hinton, 2008), leading to improved graph layout algorithms (Kruiger et al., 2017; Pitsianis et al., 2019; Zhu et al., 2020a; Zhong et al., 2023). Concurrently, researchers have unified various dimensionality reduction algorithms based on the idea of embedding high-dimensional neighbors near each other (Damrich & Hamprecht, 2021; Böhm et al., 2022) and established connections between neighbor embeddings and contrastive learning (Damrich et al., 2023; Böhm et al., 2023; Hu et al., 2023). This led to implementations that allow optimizing neighbor embeddings in high-dimensional embedding spaces (McInnes et al., 2018; Damrich et al., 2023). Despite these links between graph layouts, data visualization, and contrastive learning, state-of-the-art 2D and high-D graph representation methods remain fundamentally distinct, overlooking the potential of these theoretical connections. This raises a natural question: is there a single, principled approach which unifies these seemingly disparate methods while also achieving competitive performance across both domains, 2D graph layouts and high-D node embeddings?

In this work, we show that such an approach indeed exists. First, we introduce a neighbor-embedding-based graph layout algorithm, graph t-SNE (Figure 1), and show that it performs better than existing methods. Second, we introduce a contrastive neighbor embedding algorithm for node embeddings, graph CNE (Figure 1), and show that it outperforms DeepWalk and node2vec while being conceptually simpler and without requiring costly hyperparameter tuning. Importantly, both of our proposed techniques emerge naturally from the same underlying principle of neighbor embeddings, demonstrating that a single set of core ideas can yield state-of-the-art performance across these traditionally separate domains.

[*]Equal contribution [1]Hertie Institute for AI in Brain Health, University of Tübingen, Germany [2]Aarhus University, Denmark. Correspondence to: Jan Niklas Böhm <jan-niklas.boehm@uni-tuebingen.de>, Dmitry Kobak <dmitry.kobak@uni-tuebingen.de>.

Code is available at github.com/berenslab/contrastive-graphs.

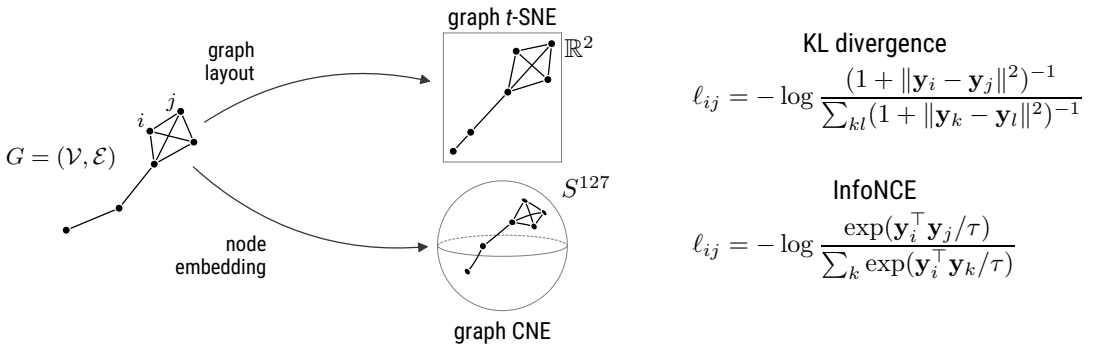

Figure 1. Abstract graph G gets embedded into $\mathbb{R}^2$ with graph t-SNE or into S^{127} with graph CNE.

2. Related Work

Graph layouts Graph layout algorithms have traditionally been based on spring models, where every connected pair of nodes feels a distance-dependent attractive force F_a and all pairs of nodes feel a distance-dependent repulsive force F_r (*force-directed graph layouts*). Many algorithms can be written as $F_a = d_{ij}^a$ and $F_r = d_{ij}^r$ (Noack, 2007), where d_{ij}^a (resp. d_{ij}^r) is the embedding distance between nodes i and j raised to the a-th (resp. r-th) power. For example, the Fruchterman–Reingold algorithm uses $a = 2, r = -1$ (Fruchterman & Reingold, 1991); ForceAtlas2 uses $a = 1$, $r = -1$ (Jacomy et al., 2014); LinLog uses $a = 0, r = -1$ (Noack, 2007). Efficient implementations can be based on Barnes–Hut approximation of the repulsive forces, as in SFDP (Hu, 2005). The relationship of ForceAtlas2 to neighbor embeddings was discussed by Böhm et al. (2022).

Several recent graph layout algorithms have been inspired by neighbor embeddings. tsNET (Kruiger et al., 2017) applied a modified version of t-SNE to the pairwise shortest path distances between all nodes. DRGraph (Zhu et al., 2020a) made tsNET faster by using negative sampling (Mikolov et al., 2013). t-FDP (Zhong et al., 2023) suggested custom F_a and F_r forces inspired by t-SNE and adopted the interpolation-based approximation of Linderman et al. (2019). In Section 7.1 we will show that our graph t-SNE outperforms both DRGraph and t-FDP.

SGtSNEpi (Pitsianis et al., 2019) is the closest method to our proposed graph t-SNE algorithm. It applies t-SNE optimization to affinities derived from the graph G, but derives these affinities in a more complex way than we do, and with additional hyperparameters (Section 4.2). In our experiments, our graph t-SNE outperformed SGtSNEpi.

Node embeddings The popular DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) algorithms optimize node placement in a high-dimensional target space based on random walks on graph G. These walks treat nodes as analogous to words and random walk paths as sentences, enabling the application of word embedding techniques to learn the representation. Specifically, DeepWalk achieves this by performing random walks from each starting node and then using the word2vec algorithm (Mikolov et al., 2013) to ensure that nodes which co-occur often in these random walks are represented near one another in the embedding space. The node2vec algorithm similarly obtains node embeddings by giving graph traversals to the word2vec algorithm, but it differs from DeepWalk by defining two parameters which control the depth-first vs. breadth-first nature of the random walk. These parameters (p and q) provide an additional level of control over the community structure uncovered by the walks, with DeepWalk being a specific instantiation of node2vec when these parameters are both set to 1.

Both DeepWalk and node2vec have been widely adopted for graph-based machine learning applications, including classification and link-prediction tasks (Khosla et al., 2019). Although connections have been drawn between Word2vec and contrastive learning (Arora et al., 2019), we emphasize that the DeepWalk and node2vec algorithms are often regarded as separate from standard contrastive techniques (Grohe, 2020).

Parametric embeddings and node-level graph contrastive learning Our paper is about *non-parametric* embeddings that only use the structure of the graph $G = (\mathcal{V}, \mathcal{E})$. In contrast, *parametric* graph contrastive learning (GCL) methods use node feature vectors and employ a neural network, usually a graph convolutional network (GCN; Kipf & Welling, 2017), to transform features into embedding vectors.

The basic principle behind contrastive learning is to learn data representation by contrasting pairs of observations that are similar to each other (positive pairs) with those that are dissimilar to each other (negative pairs). In computer vision, positive pairs are generated via data augmentation, e.g. in SimCLR (Chen et al., 2020). GCL can be graph-

level or node-level, depending on whether representations are obtained for a set of graphs or for the set of nodes of a single graph. Many graph-level (e.g. You et al., 2020) and node-level GCL algorithms (Velickovic et al., 2019; Zhu et al., 2020b; Hassani & Khasahmadi, 2020; Thakoor et al., 2021; Zhang et al., 2021; Zhu et al., 2021) are also based on graph augmentations, such as node dropping or edge perturbation. A general problem with domain-agnostic graph augmentations is that they can have unpredictable effects on graph semantics (Trivedi et al., 2022). This motivated development of augmentation-free node-level GCL methods, where positive pairs are pairs of nodes that are located close to each other in terms of graph distance (Lee et al., 2022; Li et al., 2023; Zhang et al., 2022). Recent work argued that GCL methods effectively pull connected nodes together, sometimes explicitly through their loss function, but also implicitly through the GCN architecture (Trivedi et al., 2022; Wang et al., 2023; Guo et al., 2023).

Note that Leow et al. (2019) also suggested an algorithm called 'graph t-SNE', that uses a GCN to build a parametric mapping optimizing a combination of t-SNE losses on node features and on shortest graph distances; it has almost no relation to our proposed graph t-SNE algorithm, which is non-parametric, and does not use node features or GCNs.

3. Background: Neighbor Embedding Framework

3.1. Neighbor Embeddings

Neighbor embeddings are a family of dimensionality reduction methods aiming to embed n observations from some high-dimensional metric space $\mathcal{X}$ into a lower-dimensional (usually two-dimensional) Euclidean space $\mathbb{R}^d$, such that neighborhood relationships between observations are preserved in the embedding space. Typically, $\mathcal{X}$ is another real-valued space $\mathbb{R}^p$, with $d \ll p$. We denote the original vectors as $\mathbf{x}_i \in \mathbb{R}^p$ and the embedding vectors as $\mathbf{y}_i \in \mathbb{R}^d$.

One of the most popular neighbor embedding methods, t-distributed stochastic neighbor embedding (t-SNE; van der Maaten & Hinton, 2008) is an extension of the earlier SNE (Hinton & Roweis, 2002). t-SNE minimizes the Kullback–Leibler divergence between the high-dimensional and low-dimensional *affinities* p_{ij} and q_{ij}:

$$\mathcal{L} = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} = \text{const} - \sum_{ij} p_{ij} \log q_{ij}. \quad (1)$$

Both affinity matrices are defined to be symmetric, positive, and to sum to 1. The high-dimensional affinities $\mathbf{P}$ are computed using adaptive Gaussian kernels such that the distribution of p_{ij} values for any fixed i has a given perplexity (a hyperparameter controlling the effective neighborhood size). Low-dimensional affinities $\mathbf{Q}$ are defined in t-SNE using a t-distribution kernel with one degree of freedom, also known as the Cauchy kernel:

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

In practice, t-SNE optimization can be accelerated by an approximation of the repulsive force field based on the Barnes–Hut algorithm (van der Maaten, 2014; Yang et al., 2013), on interpolation (Linderman et al., 2019), or on sampling (Artemenkov & Panov, 2020; Damrich et al., 2023; Draganov et al., 2023; Yang et al., 2023).

3.2. Contrastive Neighbor Embeddings

The contrastive neighbor embedding (CNE) algorithm (Damrich et al., 2023) is a flexible dimensionality reduction framework that replaces t-SNE's Kullback–Leibler divergence loss with contrastive losses. It considers three different loss functions: NCE (noise-contrastive estimation) (Gutmann & Hyvärinen, 2010), InfoNCE (Jozefowicz et al., 2016; Oord et al., 2018), and negative sampling (Mikolov et al., 2013). These loss functions are called *contrastive* because they are based on contrasting pairs of k-nearest neighbors and non-neighbors in the same mini-batch, and do not require a global normalization like in Equation 2. Using NCE and InfoNCE in CNE approximates t-SNE.

In this work we will only use the InfoNCE loss function. The InfoNCE loss is defined for one pair of k-nearest neighbors ij (*positive pair*) with affinity p_{ij} as

$$\ell(i,j) = -p_{ij} \log \frac{w_{ij}}{w_{ij} + \sum_{k=1}^{m} w_{ik}}, \quad (3)$$

where w_{ij} are non-normalized low-dimensional affinities and the sum in the denominator is over m *negative pairs* ik where k can be drawn from all points in the same mini-batch apart from i and j. One mini-batch consists of b pairs of neighbors, and hence contains $2b$ points. Therefore, for a given batch size b, the maximal value of m is $2b - 2$. The larger the number of negative samples m, the better is the approximation to t-SNE (Damrich et al., 2023). The InfoNCE loss aims to make w_{ij} large, i.e. place embeddings $\mathbf{y}_i$ and $\mathbf{y}_j$ nearby, if ij is a positive pair, and small if it is a negative one.

The w_{ij} affinities do not need to be normalized. When embedding into $\mathbb{R}^2$, they can be defined simply as

$$w_{ij} = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}. \quad (4)$$

When using a high-dimensional embedding space, e.g. $d = 128$ instead of $d = 2$, embedding vectors are usually projected to lie on the unit sphere. For points on the unit sphere, the cosine distance and the squared Euclidean

distance differ only by a constant, making the following definitions of w_{ij} equivalent:

$$w_{ij} = \exp\big(\mathbf{y}_i^\top \mathbf{y}_j / (\|\mathbf{y}_i\| \cdot \|\mathbf{y}_j\|)/\tau\big) \tag{5}$$

$$= \text{const} \cdot \exp\Big(-\Big\|\frac{\mathbf{y}_i}{\|\mathbf{y}_i\|} - \frac{\mathbf{y}_j}{\|\mathbf{y}_j\|}\Big\|^2 \Big/ (2\tau)\Big), \tag{6}$$

where τ is called the *temperature* (by default, $\tau = 0.5$). Together with Equation 3, this gives the same loss function as in SimCLR (Chen et al., 2020), a popular contrastive learning algorithm in computer vision. Note that instead of nearest neighbors, SimCLR uses pairs of augmented images as positive pairs.

4. Applying the Neighbor Embedding Framework to Graphs

4.1. General Approach

Standard t-SNE employs Gaussian high-dimensional affinities with most $p_{ij} \approx 0$. This can be seen as a generalization of discrete nearest neighbors: if p_{ij} is close to 0, then the points are effectively dissimilar. However, almost the same visualizations can be obtained using hard nearest neighbors, i.e. simply by normalizing and symmetrizing the kNN graph adjacency matrix $\mathbf{A}$ directly (Böhm et al., 2022):

$$\mathbf{P} = \frac{\mathbf{A}/k + \mathbf{A}^\top/k}{2n}. \tag{7}$$

Here $\mathbf{A}$ has element $a_{ij} = 1$ if $\mathbf{x}_j$ is within the k nearest neighbors of $\mathbf{x}_i$. Reasonable values of k typically lie between 10 and 100, corresponding to the effective neighborhood size for typical perplexity values.

Similarly, contrastive neighbor embeddings often directly choose the edges of the kNN graph as high-dimensional affinities, $\mathbf{P} = \mathbf{A}/\sum_{ij} A_{ij}$ (Artemenkov & Panov, 2020; Damrich et al., 2023). This is equivalent to simply leaving out p_{ij} from Equation (3).

Thus, even though neither t-SNE nor CNE are usually presented as such, they can be thought of as graph layout algorithms, specifically applied to kNN graphs. During optimization, neighboring nodes (sharing a kNN edge) feel attraction, whereas all nodes feel repulsion, arising through the normalization in Equations 2 and 3.

This suggests a simple strategy for applying the neighbor embedding framework to a general graph G: obtain affinities directly from G instead of a kNN graph of some data, and then run the t-SNE or CNE embedding optimization on these affinities (Figure 1).

4.2. Graph Layouts via Graph t-SNE

Given an unweighted graph $G = (\mathcal{V}, \mathcal{E})$, its adjacency matrix $\mathbf{A}$ has elements $A_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and $A_{ij} = 0$ otherwise. Since all graphs considered in this study are undirected, the adjacency matrix is a binary, symmetric square $n \times n$ matrix. In order to convert it into an affinity matrix suitable for t-SNE, we followed the strategy of Böhm et al. (2022) in Eq. 7: divide each row by the sum of its elements, symmetrize the resulting matrix, and then normalize to sum to 1:

$$\mathbf{P} = \frac{\tilde{\mathbf{A}} + \tilde{\mathbf{A}}^\top}{2n}, \text{ where } \tilde{A}_{ij} = A_{ij} \Big/ \sum_{k=1}^{n} A_{ik}. \tag{8}$$

For optimization, we used `openTSNE` (Poličar et al., 2019) with default parameters. It uses Laplacian Eigenmaps (Belkin & Niyogi, 2003) for initialization (Kobak & Linderman, 2021), sets the learning rate to n to achieve good convergence (Linderman & Steinerberger, 2019; Belkina et al., 2019), and employs the FIt-SNE algorithm that has linear $\mathcal{O}(n)$ runtime (Linderman et al., 2019).

We have also experimented with an alternative way to convert the adjacency matrix into the affinity matrix: namely, to divide $\mathbf{A}$ by the sum of its elements: $\mathbf{P} = \mathbf{A}/\sum_{ij} A_{ij}$. This approach resulted in lower neighbor recall and kNN accuracy values, and gave visually unpleasing embeddings, with low-degree nodes pushed out to the periphery (Figure S2c,f). Furthermore, we experimented with various initialization schemes and found that on our graphs, random initialization performed very similar to the default Laplacian Eigenmaps initialization (Figure S2b,e).

SGtSNEpi (Pitsianis et al., 2019) derives the affinity matrix $\mathbf{P}$ from the adjacency matrix $\mathbf{A}$ in a more complicated way (Pitsianis et al., 2024, Supplementary). Non-zero elements A_{ij} are first weighted by the Jaccard similarity of the sets of neighbors of nodes i and j, then power-transformed to match a pre-specified row sum λ, and finally divided by λ to yield $\tilde{\mathbf{A}}$. By default, $\lambda = 10$.

Table 1. Benchmark datasets. Columns: number of nodes in the largest connected component, number of undirected edges, number of node classes, and the average number of edges per node.

Dataset	Nodes	Edges	Classes	E/N
Citeseer	2 120	7 358	6	3.5
Cora	2 485	10 138	7	4.1
PubMed	19 717	88 648	3	4.5
Photo	7 487	238 086	8	31.8
Computer	13 381	491 556	10	36.7
MNIST kNN	70 000	1 501 392	10	21.4
arXiv	169 343	2 315 598	40	13.7
MAG	726 664	10 778 888	349	14.8

4.3. Node Embeddings via Graph CNE

As in graph t-SNE, graph CNE directly obtains the affinities from a graph G, instead of a kNN graph. Following CNE, we compute them as $\mathbf{P} = \mathbf{A}/\sum_{ij} A_{ij}$. Then, graph CNE optimizes the embedding using a contrastive loss function such as InfoNCE to make neighbors be close in the embedding (Section 3.2).

For all experiments with CNE we used the output dimensionality $d = 128$ and the InfoNCE loss with the cosine distance. We set the batch size to $\min\{8192, |\mathcal{V}|/10\}$ (in pilot experiments we noticed that small graphs required smaller batch sizes for good convergence) and used full-batch repulsion ($m = 2b - 2$). The number of epochs was set to 100. We used the Adam optimizer (Kingma & Ba, 2015) with learning rate 0.001. Graph CNE was initialized with 128-dimensional Laplacian Eigenmaps as well, although, again, there was almost no difference when using random initialization.

Note that our method is conceptually much simpler than DeepWalk and node2vec. In both of these algorithms, random walks are used to implicitly estimate node similarity by their co-occurence, and then word2vec is employed to train the embedding. Furthermore, node2vec requires per-graph hyperparameter tuning so that its random walk distributions appropriately model the input graph (Grover & Leskovec, 2016). In our graph CNE method, all nodes connected by an edge attract each other, exactly as in graph t-SNE, and no random walks are needed. The main difference between graph t-SNE & CNE is the contrastive loss for efficient optimization in 128 dimensions.

5. Experimental Setup

Datasets We used eight publicly available graph datasets (Table 1). The first five datasets were retrieved from the Deep Graph Library (Wang et al., 2019). The arXiv and MAG dataset were retrieved from the Open Graph Benchmark (Hu et al., 2020). The MNIST kNN dataset was obtained by computing the kNN graph with $k = 15$ on top of the 50 principal components of the MNIST digit dataset (Lecun et al., 1998). Each dataset was treated as an unweighted and undirected graph, where each node has a class label, used only for evaluation. We restricted ourselves to graphs with labeled nodes in order to use classification accuracy as one of the performance metrics. In all datasets we used only the largest connected component and excluded all self-loops if present, using `NetworkX` (Hagberg et al., 2008) functions `connected_components` and `selfloop_edges`.

Performance metrics We evaluated the performance using three metrics: neighbor recall, kNN classification accuracy, and linear classification accuracy.

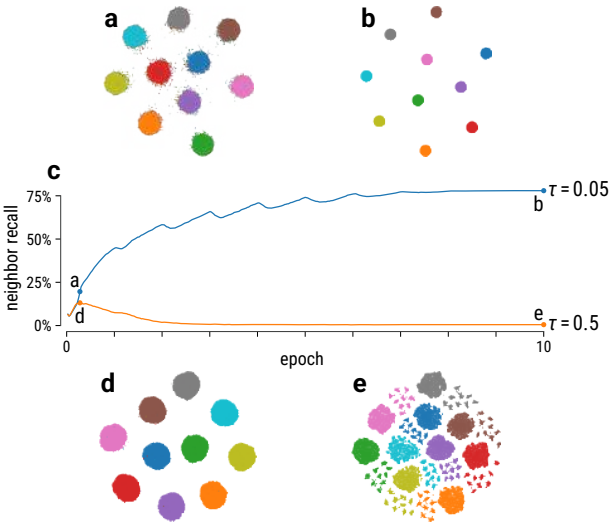

Figure 2. Learning dynamics of the 128-dimensional CNE embeddings of nodes in a stochastic block model graph with 10 blocks. **(a, b)** t-SNE visualizations of the CNE embeddings with $\tau = 0.05$, in the middle of the first epoch and after ten epochs. **(c)** The neighbor recall as a function of the training epoch, for $\tau = 0.05$ and for $\tau = 0.5$. Points correspond to t-SNE visualizations above/below. **(d, e)** Same as (a, b), but for $\tau = 0.5$.

The neighbor recall quantifies how well local node neighborhoods are preserved in the embedding. We defined it as the average fraction of each node's graph neighbors that are among the node's nearest neighbors in the embedding:

$$\text{Recall} = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \frac{\big|N_G[i] \cap N_{E,k_i}[i]\big|}{k_i}, \tag{9}$$

where $|\mathcal{V}|$ is the number of nodes in the graph, $N_G[i]$ is the set of node i's graph neighbors, $N_{E,k}[i]$ denotes the set of node i's k Euclidean nearest neighbors in the embedding space, and $k_i = |N_G[i]|$ is the number of node i's graph neighbors. This metric does not require ground truth classes and is similar to what is commonly used in the literature to benchmark graph layout algorithms (Kruiger et al., 2017; Zhu et al., 2020a; Zhong et al., 2023). Therefore, we use this as our primary metric for measuring graph layout quality.

The kNN classification accuracy quantifies local class separation in the embedding. To calculate kNN accuracy, we randomly split all nodes into a training (90% of all nodes) and a test set (10%), and used the `KNeighborsClassifier` from scikit-learn with $k = 10$ (Pedregosa et al., 2011).

We used the Euclidean distance to calculate all kNN evaluations (recall and accuracy) in $d = 2$, and the cosine similarity for evaluations in $d = 128$. CNE uses the cosine metric

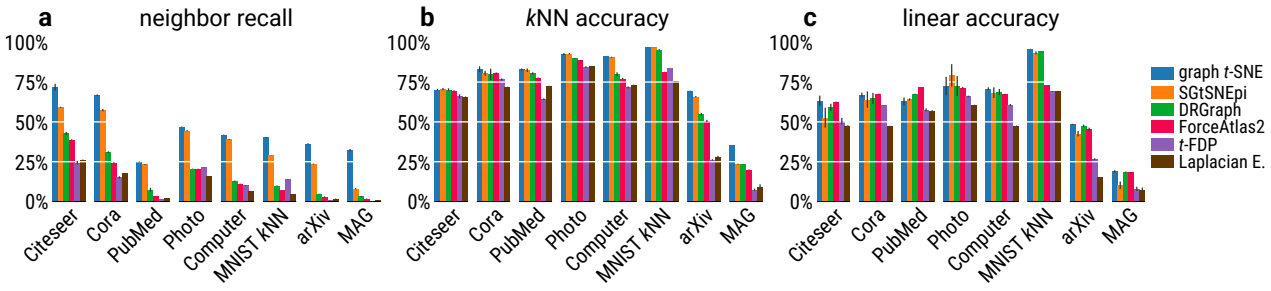

Figure 3. Performance metrics for graph layouts: **(a)** neighbor recall, **(b)** kNN accuracy, and **(c)** linear accuracy. Datasets are ordered by the number of edges. See Figures S4 and 5 for the corresponding layouts.

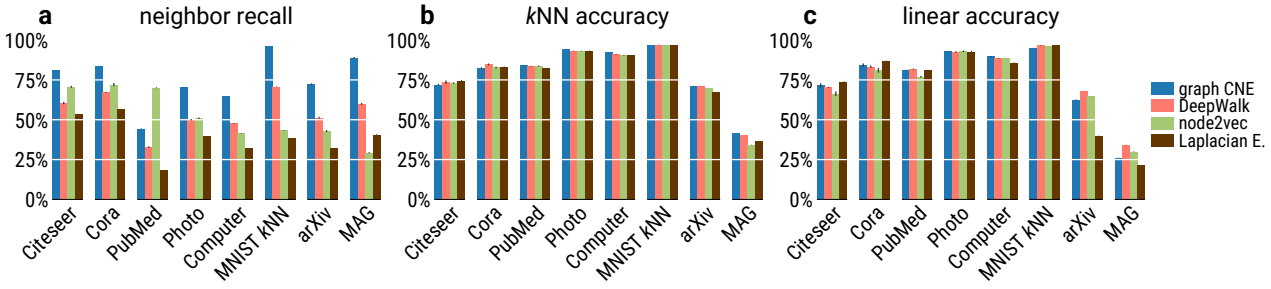

Figure 4. Performance metrics for node embeddings: **(a)** neighbor recall, **(b)** kNN accuracy, **(c)** linear accuracy. Datasets are ordered by the number of edges. For node2vec we did a grid search over $p, q \in \{0.25, 0.5, 1, 2, 4\}$ and show results with the highest neighbor recall.

in its loss function (Equation 6), so only cosine neighbors make sense for evaluation. DeepWalk and node2vec rely on word2vec which uses dot product similarity in the loss function, and the original papers also used cosine metric for evaluation (in our experiments cosine evaluation led to better results on average).

For linear accuracy we used `LogisticRegression` from scikit-learn with no regularization (`penalty=None`), SAGA solver (Defazio et al., 2014) with `tol=0.01`, and the same train/test split. We standardized all features based on the training set.

6. Graph CNE Requires Low Temperature

In pilot experiments, we noticed that the performance of graph CNE was strongly affected by the temperature parameter τ. To investigate it further, we synthesized a graph following the stochastic block model (SBM; Holland et al., 1983). The generated graph had 80 000 nodes in 10 clusters, with any two nodes from the same cluster having probability $2.5 \cdot 10^{-3}$ to be connected by an edge, and any two nodes from two different clusters having probability $5 \cdot 10^{-6}$ to be connected. The resulting graph has a clear community structure that should be easy to recover.

CNE with the default temperature $\tau = 0.5$ achieved near-perfect class separation but failed to retain the neighborhood structure. The neighbor recall, after reaching 13% within the first training epoch, collapsed to below 1% over the next several epochs (Figure 2c, orange line). The t-SNE visualization of the high-dimensional embedding at the point of maximum neighbor recall showed ten compact clusters (Figure 2d), but after convergence it showed nine subclusters for each of the ten classes (Figure 2e). These smaller subclusters corresponded to nodes with an inter-cluster edge to a specific other class. During the optimization, these nodes got 'pulled out' of their class, destroying the local structure of the embedding and leading to near-zero neighbor recall.

In contrast, CNE with a lower temperature $\tau = 0.05$ did not show this behavior. The neighbor recall was almost monotonically increasing during training, reaching 78% after 10 epochs (Figure 2c, blue line). The t-SNE visualization showed ten compact clusters (Figure 2b), without any visible subclusters. Our interpretation is that the InfoNCE loss with low temperature could effectively ignore the noise in form of rare inter-class edges.

In the following experiments, we set the temperature of graph CNE to $\tau = 0.05$ for all datasets. We have also implemented learnable temperature, making τ an additional

Table 2. Neighbor recall for all methods and datasets (in %). All values are mean ± standard deviation across three training runs. The top performing method for each dimensionality is highlighted in bold. Methods in **blue** are ours.

d	Method	Citeseer	Cora	PubMed	Photo	Computer	MNIST	arXiv	MAG
2	**graph t-SNE**	**71.7** ± 2.2	**66.7** ± 0.5	**25.0** ± 0.2	**46.9** ± 0.2	**41.8** ± 0.1	**40.2** ± 0.2	**36.3** ± 0.3	**32.3** ± 0.7
	SGtSNEpi	59.1 ± 0.3	57.4 ± 0.8	23.3 ± 0.3	44.5 ± 0.4	39.0 ± 0.3	29.1 ± 0.1	23.6 ± 0.3	8.1 ± 0.4
	DRGraph	42.8 ± 1.0	31.0 ± 0.5	7.5 ± 1.1	20.5 ± 0.1	12.8 ± 0.4	10.0 ± 0.1	4.7 ± 0.2	3.2 ± 0.3
	ForceAtlas2	38.8 ± 0.3	24.4 ± 0.3	3.2 ± 0.1	20.6 ± 0.1	11.1 ± 0.1	7.0 ± 0.0	2.9 ± 0.3	1.7 ± 0.2
	t-FDP	24.4 ± 1.2	15.2 ± 0.7	1.3 ± 0.1	21.6 ± 0.1	10.2 ± 0.2	13.9 ± 0.1	0.7 ± 0.2	0.3 ± 0.1
	Laplacian E.	26.2 ± 0.1	17.9 ± 0.0	2.0 ± 0.1	16.0 ± 0.0	6.4 ± 0.0	5.0 ± 0.0	1.6 ± 0.3	0.8 ± 0.1
128	**graph CNE**	**81.0** ± 0.1	**83.8** ± 0.0	44.3 ± 0.2	**70.3** ± 0.1	**64.8** ± 0.0	**96.0** ± 0.0	**72.3** ± 0.6	**89.0** ± 0.5
	DeepWalk	60.5 ± 0.9	67.1 ± 0.4	32.9 ± 0.6	50.0 ± 0.5	47.7 ± 0.3	70.8 ± 0.2	51.4 ± 0.6	60.0 ± 0.7
	node2vec	70.7 ± 0.6	72.1 ± 1.0	**70.1** ± 0.3	50.9 ± 0.5	41.3 ± 0.2	43.2 ± 0.2	42.9 ± 0.6	29.3 ± 0.4
	Laplacian E.	53.4 ± 0.0	56.7 ± 0.0	18.3 ± 0.1	39.7 ± 0.0	32.4 ± 0.0	38.5 ± 0.0	32.1 ± 0.1	40.6 ± 0.3

trainable parameter. We found that on all our benchmark datasets, this the temperature converged towards a value in a range of $[0.04, 0.08]$. The evaluation results were also close to the results with fixed $\tau = 0.05$ (Tables S1 to S3).

7. Benchmarking graph t-SNE and CNE

7.1. Graph t-SNE Outperforms Other Graph Layouts

We compared graph t-SNE with five existing graph layout algorithms: SGtSNEpi (Pitsianis et al., 2019), ForceAtlas2 (FA2; Jacomy et al., 2014), Laplacian Eigenmaps (LE; Belkin & Niyogi, 2003), DRGraph (Zhu et al., 2020a), and t-FDP (Zhong et al., 2023). We did not include tsNET (Kruiger et al., 2017), because it cannot embed large graphs and is outperformed by its successor DRGraph. Unless specified otherwise, we used the original implementation of the algorithms and ran them with the default parameters. For FA2 we used the Barnes–Hut implementation by Chippada (2017). For LE we used scikit-learn, which in turn uses LOBPCG (Knyazev et al., 2007) for solving the generalized eigenproblem. Both t-SNE and t-FDP are implemented in Cython, DRGraph and SGtSNEpi are implemented in C++ and offer wrappers in Python and Julia, respectively. For consistency, we used LE initialization for all algorithms unless not possible (SGtSNEpi and DRGraph).

Graph t-SNE showed outstanding performance on all our benchmark datasets. The neighbor recall of graph t-SNE was always the highest, with SGtSNEpi only sometimes coming close (Figure 3a, Table 2). In terms of kNN accuracy, graph t-SNE was either the top performing method or within 1% of the top performing method for all datasets (Figure 3b, Table S2). In terms of linear accuracy the same was true for six out of the eight datasets (Figure 3c, Table S3). Qualitatively, graph t-SNE layouts did well in terms of separating clusters from each other and bringing out sub-cluster details within individual clusters (Figure 5).

7.2. Graph CNE Outperforms Other Node Embeddings

We compared graph CNE with the popular non-parametric node embedding algorithms DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) (all optimizing 128-dimensional embeddings), as well as with 128-dimensional Laplacian Eigenmaps (LE). We used node2vec's implementation from PyTorch Geometric (Fey & Lenssen, 2019) and the DeepWalk implementation from DGL (Wang et al., 2019). We ran both methods with the default parameters for 100 epochs (as we did for CNE, see Figure S1 for runtimes). For node2vec, we ran a sweep over the parameters $p, q \in \{0.25, 0.5, 1, 2, 4\}$, as in the original paper, and report the results with the highest neighbor recall (for all results, see Figure S3).

We found that graph CNE outperformed the other algorithms in terms of neighbor recall on seven datasets out of eight; on the PubMed dataset, it was the runner-up (Figure 4a, Table 2). Across all datasets, the average gap in neighbor recall between graph CNE and the best other method was 13.4 percentage points.

In terms of the classification accuracies, the results on most datasets were very similar across all methods including LE. Graph CNE had slightly lower kNN accuracy on the two smallest datasets (Citeseer and Cora), and was the best or within 1% of the best on all other datasets (Figure 4b, Table S2). In terms of linear accuracy, graph CNE yielded competitive results and lagged only slightly behind other methods for some datasets (Figure 4c, Table S3). Curiously, graph CNE with $\tau = 0.5$ was the best or within 1% of the best on all datasets apart from Cora and MAG, where it was slightly behind (Table S3); but this temperature led to substantially worse neighbor recall (Table S1). This suggests a trade-off between linear classification and neighbor quality.

In summary, results in terms of classification accuracies were all similar, but neighbor recall showed large and pronounced differences with graph CNE performing the best.

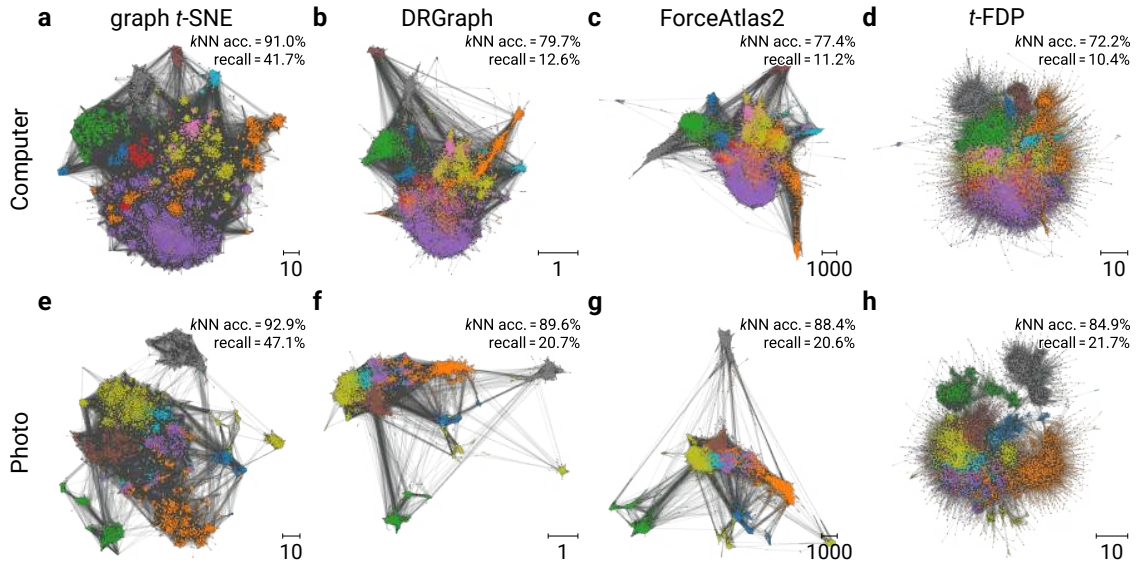

Figure 5. Embeddings of the Computer and Photo dataset obtained using our graph t-SNE, DRGraph, ForceAtlas2, and t-FDP. Embeddings were aligned using Procrustes rotation. See Figure S4 for all datasets and methods.

8. Discussion

Our paper makes three contributions, two practical and one conceptual. First, we suggested a novel graph layout algorithm, *graph t-SNE*, and showed that it outperforms existing competitors in terms of preserving local graph structure. Second, we suggested a novel node embedding algorithm, *graph CNE*, and showed that it outperforms existing competitors (DeepWalk and node2vec) in terms of preserving local graph structure, while being conceptually simpler and not requiring random walks. Third, we established a *conceptual connection* between 2D graph layouts and high-D node embeddings, and showed that both can be efficiently implemented using existing neighbor embedding frameworks.

Both graph t-SNE and graph CNE are remarkably simple, because they use existing t-SNE and CNE machinery out of the box. This is in stark contrast with competing algorithms. For example, many existing graph layout algorithms inspired by t-SNE, such as tsNET (Kruiger et al., 2017), DRGraph (Zhu et al., 2020a), and t-FDP (Zhong et al., 2023), all develop their own machinery, implementation, and approximations, and deviate from t-SNE in many different nontrivial ways (see Section 2). SGtSNEpi (Pitsianis et al., 2019) is one exception: it also runs t-SNE optimization on graph-derived affinities, however its affinities are more complicated than ours, and our results suggest that this is not needed and is often even detrimental. As we demonstrated, simply applying t-SNE optimization to the normalized graph adjacency matrix (i.e. graph t-SNE) leads to the best layout quality.

Similarly, graph CNE outperformed both DeepWalk and node2vec without using any random walks, by simply pulling together connected nodes via contrastive InfoNCE loss function, which is ubiquitous in self-supervised learning in computer vision (Chen et al., 2020) and other domains.

Limitations and future work In this work, we focused on complex real-world graphs and have purposefully not tested our graph t-SNE on simple planar graphs or 3D mesh graphs that are often used for benchmarking graph layout algorithms. Such simple graphs are arguably not an interesting case for high-dimensional embeddings, and we aimed to use the same graphs for 2D and high-D benchmarks.

Our work opens up several directions for future work. First, CNE allows to train parametric embeddings (Damrich et al., 2023), which we have not explored here. How would parametric CNE with a GCN mapping compare to existing GCL methods, in particular augmentation-free methods? Second, we only used t-SNE here, but a similar approach could be implemented using other neighbor embedding algorithms, e.g. UMAP (McInnes et al., 2018). How would graph UMAP (2D and high-D) perform for graph layouts and node embeddings, especially in contrast to DRgraph and DeepWalk/node2vec, which, like UMAP, use negative sampling for optimization? Third, our results pointed to a nontrivial effect that temperature can have on InfoNCE-based embeddings. Further investigation of this phenomenon and its potential relevance for contrastive learning in computer vision and other domains also remains for future work.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Pitsianis, N., Iliopoulos, A.-S., Floros, D., and Sun, X. Spaceland embedding of sparse stochastic graphs. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, 2019.

Pitsianis, N., Iliopoulos, A.-S., Floros, D., and Sun, X. Sg-tsne-π. http://web.archive.org/web/20250124132719/https://t-sne-pi.cs.duke.edu/, 2024. Accessed: 2025-01-25.

Poličar, P. G., Stražar, M., and Zupan, B. openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding. *BioRxiv*, pp. 731877, 2019.

Schönemann, P. H. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 1966.

Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

Trivedi, P., Lubana, E. S., Yan, Y., Yang, Y., and Koutra, D. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *Proceedings of the ACM Web Conference 2022*, pp. 1538–1549, 2022.

van der Maaten, L. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Devon, R. Deep graph infomax. *International Conference for Learning Representations*, 2(3):4, 2019.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Wang, Y., Zhang, Q., Du, T., Yang, J., Lin, Z., and Wang, Y. A message passing perspective on learning dynamics of contrastive learning. *International Conference on Learning Representations*, 2023.

Yang, Z., Peltonen, J., and Kaski, S. Scalable optimization of neighbor embedding for visualization. In *International Conference on Machine Learning*, pp. 127–135. PMLR, 2013.

Yang, Z., Chen, Y., Sedov, D., Kaski, S., and Corander, J. Stochastic cluster embedding. *Statistics and Computing*, 33(1):12, 2023.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33: 5812–5823, 2020.

Zhang, H., Wu, Q., Yan, J., Wipf, D., and Yu, P. S. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021.

Zhang, H., Wu, Q., Wang, Y., Zhang, S., Yan, J., and Yu, P. S. Localized contrastive learning on graphs. *arXiv preprint arXiv:2212.04604*, 2022.

Zhong, F., Xue, M., Zhang, J., Zhang, F., Ban, R., Deussen, O., and Wang, Y. Force-directed graph layouts revisited: a new force based on the t-distribution. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

Zhu, M., Chen, W., Hu, Y., Hou, Y., Liu, L., and Zhang, K. DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1666–1676, 2020a.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020b.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pp. 2069–2080, 2021.

A. Supplementary Figures and Tables

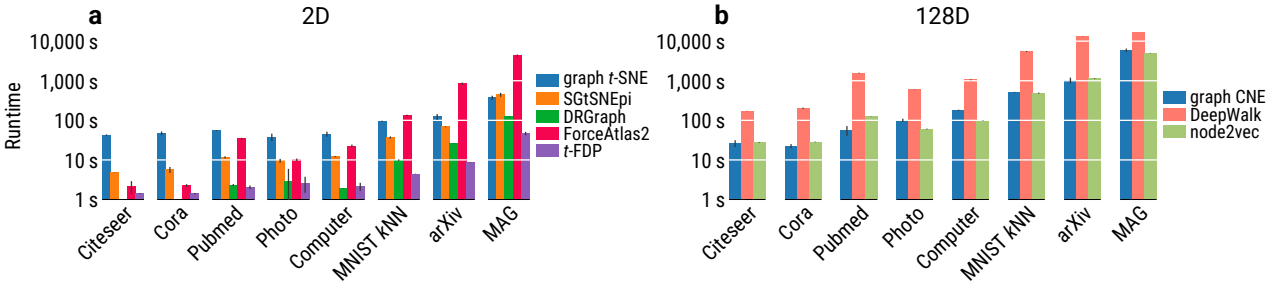

Figure S1. Computation times for graph t-SNE and graph CNE with 2 and 128 output dimensions. All computations were performed on a cluster which isolates the computing resources and removes interference between concurrent computations. All 2D experiments require only CPUs and were ran on 8 cores of an Intel Xeon Gold 6226R. Experiments in 128D ran on a single Nvidia 2080ti GPU card. For node2vec, this shows runtime for $p = 1, q = 1$; we ran 25 parameter combinations, so our actual runtime including hyperparameter tuning was much larger.

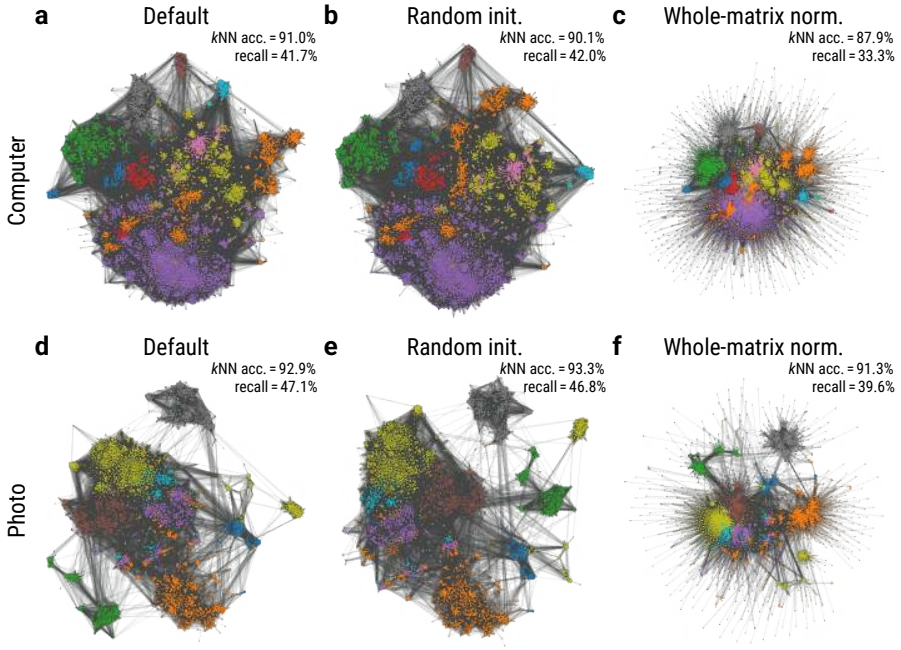

Figure S2. The effect of initialization and normalization on graph t-SNE of Computer and Photo datasets. **(a, d)** Default graph t-SNE with Laplacian Eigenmaps initialization and using per-node normalization of the adjacency matrix. **(b, e)** Graph t-SNE using random initialization. **(c, f)** Graph t-SNE with whole-matrix normalization. Embeddings in each row were aligned using Procrustes rotation.

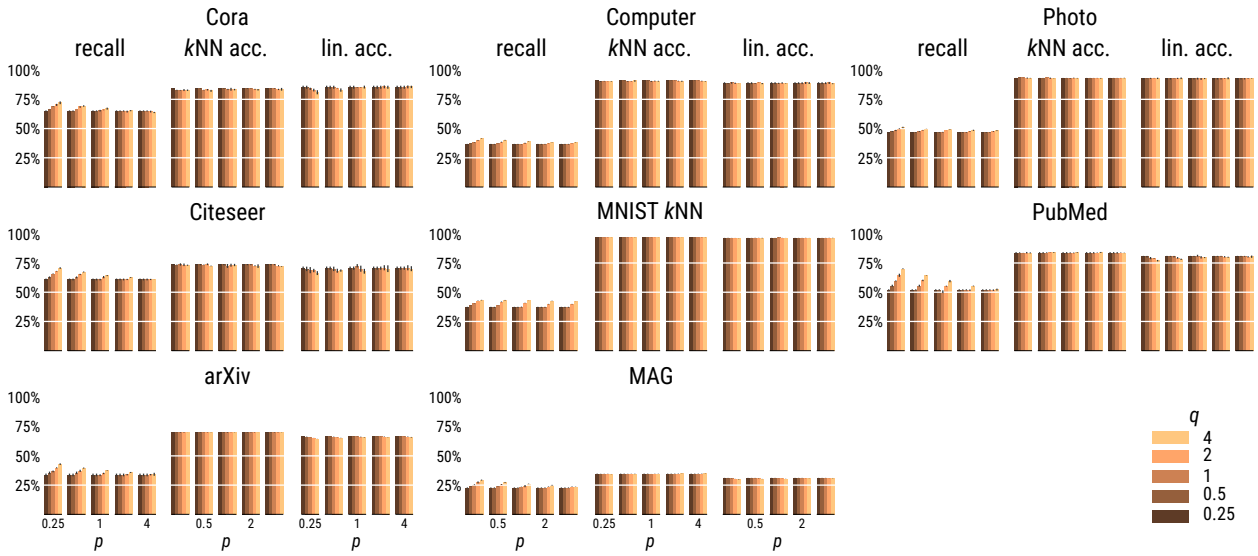

Figure S3. Evaluation for node2vec (Grover & Leskovec, 2016) with the hyperparameters $p, q \in \{0.25, 0.5, 1, 2, 4\}$. These parameter values were taken from the original node2vec paper. The highest neighbor recall was always achieved at $p = 0.25$, $q = 4$. This corresponds to oversampling unseen nodes.

Table S1. Neighbor recall for all methods and datasets (in %). All values are mean ± standard deviation across three training runs and random training/test splits. The top performing method for each dimensionality (and all methods within 1%) is highlighted in bold. Methods in **blue** are ours. "Graph CNEτ" means that the temperature τ was learned as a parameter during training, see Section 6.

d	Method	Citeseer	Cora	PubMed	Photo	Computer	MNIST	arXiv	MAG
2	**graph t-SNE**	**71.7** ± 1.8	**66.7** ± 0.4	**25.0** ± 0.1	**46.9** ± 0.1	**41.8** ± 0.1	**40.2** ± 0.2	**36.3** ± 0.3	**32.3** ± 0.6
	SGtSNEpi	59.1 ± 0.3	57.4 ± 0.6	23.3 ± 0.2	44.5 ± 0.3	39.0 ± 0.2	29.1 ± 0.1	23.6 ± 0.2	8.1 ± 0.3
	DRGraph	42.8 ± 0.8	31.0 ± 0.4	7.5 ± 0.9	20.5 ± 0.1	12.8 ± 0.3	10.0 ± 0.1	4.7 ± 0.1	3.2 ± 0.2
	ForceAtlas2	38.8 ± 0.3	24.4 ± 0.3	3.2 ± 0.1	20.6 ± 0.1	11.1 ± 0.1	7.0 ± 0.0	2.9 ± 0.3	1.7 ± 0.1
	t-FDP	24.4 ± 0.9	15.2 ± 0.6	1.3 ± 0.1	21.6 ± 0.1	10.2 ± 0.2	13.9 ± 0.1	0.7 ± 0.1	0.3 ± 0.1
	Laplacian E.	26.2 ± 0.1	17.9 ± 0.0	2.0 ± 0.1	16.0 ± 0.0	6.4 ± 0.0	5.0 ± 0.0	1.6 ± 0.2	0.8 ± 0.1
128	**graph CNE**	**81.0** ± 0.1	**83.8** ± 0.0	44.3 ± 0.2	**70.3** ± 0.1	**64.8** ± 0.0	**96.0** ± 0.0	**72.3** ± 0.5	89.0 ± 0.4
	graph CNEτ	**80.3** ± 0.0	**82.8** ± 0.1	42.3 ± 0.1	**69.5** ± 0.0	**64.0** ± 0.1	**96.0** ± 0.0	**72.5** ± 0.7	**91.0** ± 0.1
	CNE, $\tau = 0.5$	55.9 ± 0.1	58.1 ± 0.0	21.9 ± 0.3	35.7 ± 0.1	31.9 ± 0.0	39.2 ± 0.0	34.4 ± 0.2	33.2 ± 0.2
	DeepWalk	60.5 ± 0.7	67.1 ± 0.3	32.9 ± 0.5	50.0 ± 0.4	47.7 ± 0.3	70.8 ± 0.2	51.4 ± 0.5	60.0 ± 0.6
	node2vec	70.7 ± 0.5	72.1 ± 1.0	**70.1** ± 0.3	50.9 ± 0.5	41.3 ± 0.2	43.2 ± 0.2	42.9 ± 0.6	29.3 ± 0.3
	Laplacian E.	53.4 ± 0.0	56.7 ± 0.0	18.3 ± 0.1	39.7 ± 0.0	32.4 ± 0.0	38.5 ± 0.0	32.1 ± 0.1	40.6 ± 0.2

Table S2. kNN classification accuracy for all methods and datasets (in %). The same setup as in Table S1 applies.

d	Method	Citeseer	Cora	PubMed	Photo	Computer	MNIST	arXiv	MAG
2	**graph t-SNE**	**70.3** ± 0.4	**83.1** ± 1.5	**82.9** ± 0.5	**92.6** ± 0.5	**91.0** ± 0.0	**96.8** ± 0.1	**69.4** ± 0.2	**35.3** ± 0.0
	SGtSNEpi	**70.6** ± 0.6	80.5 ± 1.3	**82.4** ± 0.9	**92.8** ± 0.4	**90.6** ± 0.3	**96.9** ± 0.1	65.9 ± 0.3	23.5 ± 0.3
	DRGraph	**70.3** ± 0.7	79.8 ± 2.9	80.4 ± 0.6	89.8 ± 0.2	80.1 ± 0.9	95.2 ± 0.3	54.7 ± 0.7	23.3 ± 0.1
	ForceAtlas2	69.3 ± 0.4	80.6 ± 0.3	77.6 ± 0.2	88.6 ± 0.2	77.2 ± 0.4	81.4 ± 0.0	50.6 ± 0.6	20.0 ± 0.2
	t-FDP	66.0 ± 1.0	76.6 ± 0.6	64.2 ± 0.5	84.5 ± 0.4	71.6 ± 0.5	83.9 ± 0.0	26.1 ± 0.4	7.5 ± 0.6
	Laplacian E.	65.6 ± 0.0	71.8 ± 0.0	72.4 ± 0.2	84.8 ± 0.2	73.2 ± 0.2	75.3 ± 0.1	27.8 ± 0.8	9.4 ± 1.2
128	**graph CNE**	72.0 ± 0.4	82.7 ± 0.0	**84.1** ± 0.1	**94.3** ± 0.1	**92.4** ± 0.1	**97.2** ± 0.0	**71.3** ± 0.1	**41.6** ± 0.1
	graph CNEτ	72.2 ± 0.4	83.1 ± 0.3	**83.8** ± 0.3	**94.3** ± 0.1	**92.4** ± 0.2	**97.1** ± 0.0	**71.7** ± 0.1	**41.7** ± 0.1
	CNE, $\tau = 0.5$	72.8 ± 0.2	83.3 ± 0.2	**83.1** ± 0.0	92.6 ± 0.0	91.4 ± 0.0	**96.9** ± 0.0	**71.1** ± 0.1	36.5 ± 0.1
	DeepWalk	**73.6** ± 1.0	**84.7** ± 0.6	**83.7** ± 0.2	93.3 ± 0.1	91.1 ± 0.2	**97.0** ± 0.1	**71.2** ± 0.1	40.5 ± 0.0
	node2vec	73.1 ± 0.4	82.8 ± 0.5	**83.6** ± 0.4	93.1 ± 0.2	90.5 ± 0.2	**96.8** ± 0.0	70.1 ± 0.0	34.4 ± 0.1
	Laplacian E.	**74.5** ± 0.0	83.1 ± 0.0	82.6 ± 0.0	93.0 ± 0.0	90.3 ± 0.0	**96.7** ± 0.0	67.2 ± 0.1	36.5 ± 0.0

Table S3. Linear classification accuracy for all methods and datasets (in %). The same setup as in Table S1 applies.

d	Method	Citeseer	Cora	PubMed	Photo	Computer	MNIST	arXiv	MAG
2	**graph t-SNE**	**63.2** ± 2.7	**66.9** ± 1.2	62.8 ± 2.0	72.5 ± 4.8	**70.9** ± 0.7	**96.0** ± 0.1	**48.4** ± 0.3	**18.9** ± 0.6
	SGtSNEpi	52.7 ± 5.2	64.0 ± 4.2	64.1 ± 0.5	**79.2** ± 5.8	68.3 ± 2.8	93.3 ± 0.7	42.6 ± 1.4	10.2 ± 1.9
	DRGraph	59.1 ± 1.8	64.8 ± 2.8	67.4 ± 0.2	72.5 ± 5.2	68.8 ± 1.5	94.2 ± 0.1	**47.6** ± 0.6	**18.5** ± 0.3
	ForceAtlas2	**62.4** ± 0.2	**67.3** ± 0.0	**71.6** ± 0.0	71.4 ± 0.4	67.3 ± 0.2	73.1 ± 0.0	45.4 ± 0.8	**18.2** ± 0.3
	t-FDP	50.5 ± 1.7	60.3 ± 0.2	57.5 ± 0.8	66.0 ± 0.4	60.6 ± 0.5	69.1 ± 0.2	26.5 ± 0.5	8.0 ± 1.1
	Laplacian E.	47.6 ± 0.0	47.2 ± 0.0	57.1 ± 0.0	60.4 ± 0.0	47.3 ± 0.0	69.4 ± 0.0	15.6 ± 0.0	7.1 ± 1.4
128	**graph CNE**	71.5 ± 1.2	84.3 ± 1.0	80.9 ± 0.2	**93.0** ± 0.2	**89.7** ± 0.2	95.3 ± 0.0	62.6 ± 0.2	26.1 ± 0.1
	graph CNEτ	71.9 ± 1.4	83.6 ± 0.7	80.2 ± 0.7	**93.4** ± 0.4	**89.5** ± 0.4	90.8 ± 0.3	63.6 ± 0.1	27.6 ± 0.1
	CNE, $\tau = 0.5$	**72.8** ± 0.8	84.3 ± 0.7	**83.6** ± 0.3	**92.7** ± 0.2	**89.4** ± 0.2	**97.1** ± 0.0	**67.9** ± 0.2	32.2 ± 0.0
	DeepWalk	70.3 ± 0.0	83.3 ± 0.7	81.9 ± 0.3	**92.5** ± 0.4	88.5 ± 0.2	**96.7** ± 0.1	**68.1** ± 0.0	**34.2** ± 0.1
	node2vec	66.4 ± 1.4	81.0 ± 1.4	77.0 ± 0.5	**93.0** ± 0.5	88.5 ± 0.1	96.1 ± 0.1	64.6 ± 0.0	29.9 ± 0.2
	Laplacian E.	**73.6** ± 0.0	**86.7** ± 0.0	81.4 ± 0.0	**92.8** ± 0.0	85.5 ± 0.0	**97.0** ± 0.0	40.0 ± 0.1	21.3 ± 0.1

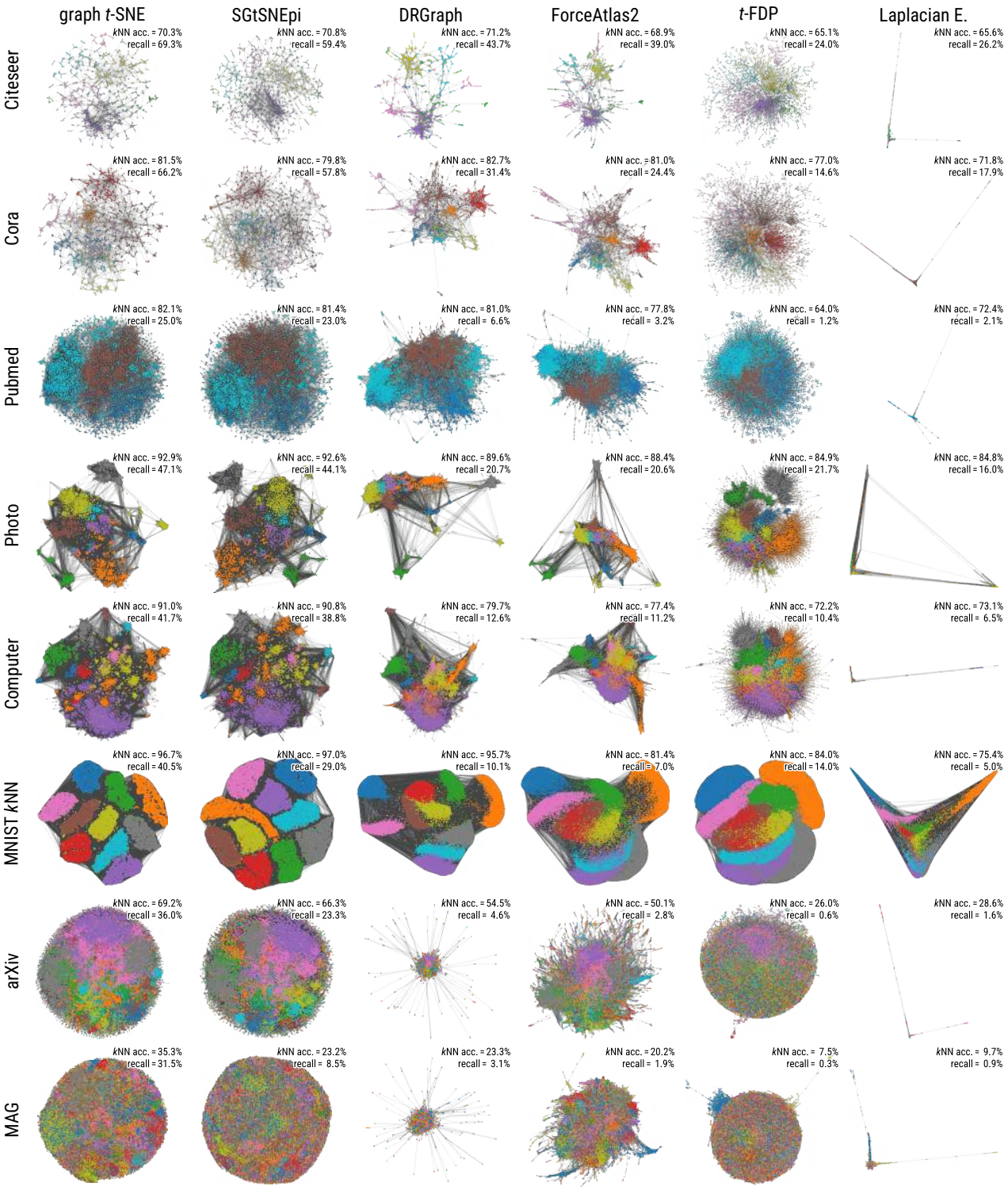

Figure S4. Embeddings of all considered datasets obtained using our graph t-SNE, SGtSNEpi (Pitsianis et al., 2019), DRGraph (Zhu et al., 2020a), ForceAtlas2 (Jacomy et al., 2014), t-FDP (Zhong et al., 2023), and Laplacian Eigenmaps (Belkin & Niyogi, 2003). Embeddings in each row were aligned using orthogonal Procrustes rotation (Schönemann, 1966).

Bibliography

B

Aggarwal, Charu C., Alexander Hinneburg, & Daniel A. Keim (Oct. 2001). "On the Surprising Behavior of Distance Metrics in High Dimensional Space". In: *International Conference on Database Theory*.

All of Us Research Program Genomics Investigators (Feb. 19, 2024). "Genomic data in the All of Us Research Program". In: *Nature* 627, pp. 340–346.

Amid, Ehsan & Manfred Klaus Warmuth (2019). "TriMap: Large-scale Dimensionality Reduction Using Triplets". In: *arXiv*.

Arora, Sanjeev, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, & Nikunj Saunshi (2019). "A theoretical analysis of contrastive unsupervised representation learning". In: *arXiv preprint arXiv:1902.09229*.

Artemenkov, Aleksandr & Maxim Panov (2020). "NCVis: Noise Contrastive Approach for Scalable Visualization". In: *The Web Conference*, pp. 2941–2947.

Ash, Jordan, Surbhi Goel, Akshay Krishnamurthy, & Dipendra Misra (2022). "Investigating the Role of Negatives in Contrastive Representation Learning". In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Proceedings of Machine Learning Research, pp. 7187–7209.

Bachman, Philip, R Devon Hjelm, & William Buchwalter (2019). "Learning Representations by Maximizing Mutual Information across Views". In: *Advances in Neural Information Processing Systems*. Vol. 32, pp. 15535–15545.

Balestriero, Randall & Yann LeCun (2022). "Contrastive and Non-Contrastive Self-Supervised Learning Recover Global and Local Spectral Embedding Methods". In: *Advances in Neural Information Processing Systems*.

Barnes, Josh & Piet Hut (1986). "A hierarchical O(N log N) force-calculation algorithm". In: *Nature* 324.6096, pp. 446–449.

Barz, Björn & Joachim Denzler (2020). "Do We Train on Test Data? Purging CIFAR of Near-Duplicates". In: *Journal of Imaging* 6.6.

Becht, Etienne, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, & Evan W Newell (2019). "Dimensionality reduction for visualizing single-cell data using UMAP". In: *Nature Biotechnology* 37.1, pp. 38–44.

Behnel, Stefan, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, & Kurt Smith (2011). "Cython: The best of both worlds". In: *Computing in Science & Engineering* 13.2, pp. 31–39.

Belkin, Mikhail & Partha Niyogi (2002). "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering". In: *Advances in Neural Information Processing Systems*. Vol. 14, pp. 585–591.

— (2003). "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural Computation* 15.6, pp. 1373–1396.

Belkina, Anna C, Christopher O Ciccolella, Rina Anno, Richard Halpert, Josef Spidlen, & Jennifer E Snyder-Cappione (2019). "Automated optimized param-

eters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets". In: *Nature Communications* 10.1, pp. 1–12.
Bernhardsson, Erik (2013). *Annoy*. https://github.com/spotify/annoy.
Bishop, Christopher Michael (Feb. 2006). *Pattern Recognition and Machine Learning*. Munich, Germany: Springer-Verlag Berlin Heidelberg. ISBN: 0-387-31073-8.
Böhm, Jan Niklas (Sept. 1, 2020). "Dimensionality Reduction with Neighbor Embeddings". MA thesis. University of Tübingen.
Böhm, Jan Niklas, Philipp Berens, & Dmitry Kobak (Mar. 2022). "Attraction-Repulsion Spectrum in Neighbor Embeddings". In: *Journal of Machine Learning Research* 23.95, pp. 1–32.
— (July 2023). "Unsupervised visualization of image datasets using contrastive learning". In: *International Conference on Learning Representations*.
Böhm, Jan Niklas, Marius Keute, Alica Guzmán, Sebastian Damrich, Andrew Draganov, & Dmitry Kobak (Jan. 2025). "Node Embeddings via Neighbor Embeddings". In: *Under review*.
Box, George Edward Pelham (Dec. 1976). "Science and Statistics". In: *Journal of the American Statistical Association* 71.356, pp. 791–799.
do Carmo, Manfredo Perdigão (1992). *Riemannian Geometry*. Trans. by Francis Flaherty. Boston, MA: Birkhäuser. ISBN: 0-8176-3490-8.
Caron, Mathilde, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, & Armand Joulin (2020). "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". In: 33, pp. 9912–9924.
Chan, David M, Roshan Rao, Forrest Huang, & John F Canny (2018). "t-SNE-CUDA: GPU-Accelerated t-SNE and its Applications to Modern Data". In: *2018 30th International Symposium on Computer Architecture and High Performance Computing*. IEEE, pp. 330–338.
Chari, Tara & Lior Pachter (Aug. 2023). "The specious art of single-cell genomics". In: *PLOS Computational Biology* 19.8, pp. 1–20.
Charlier, Benjamin, Jean Feydy, Joan Alexis Glaunès, François-David Collin, & Ghislain Durif (2021). "Kernel Operations on the GPU, with Autodiff, without Memory Overflows". In: *Journal of Machine Learning Research* 22.74, pp. 1–6.
du Châtellier, Paul (1901). "Les pierres gravées de Penhoat, en Saint-Coulitz et de Sanct-Bélec, en Leuhan". In: *Bulletin de la Société Archéologique du Finistère* 28.
Chen, Ting, Simon Kornblith, Mohammad Norouzi, & Geoffrey Everest Hinton (2020). "A simple framework for contrastive learning of visual representations". In: *International Conference on Machine Learning*, pp. 1597–1607.
Chippada, Bhargav (2017). *forceatlas2: Fastest Gephi's ForceAtlas2 graph layout algorithm implemented for Python and NetworkX*. https://github.com/bhargavchippada/forceatlas2.
Cho, Hyunghoon, Bonnie Berger, & Jian Peng (2018). "Generalizable and scalable visualization of single-cell data using neural networks". In: *Cell Systems* 7.2, pp. 185–191.
Coenen, Andy & Adam Pearce (2022). *A deeper dive into UMAP theory*. https://pair-code.github.io/understanding-umap/supplement.html. Accessed: 2022-05-11.
Cohen, Taco & Max Welling (June 2016). "Group Equivariant Convolutional Networks". In: *International Conference on Machine Learning*. Ed. by Maria Florina Balcan & Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA, pp. 2990–2999.

Cohen, Taco S., Mario Geiger, Jonas Köhler, & Max Welling (2018). "Spherical CNNs". In: *International Conference on Learning Representations*.

Damrich, Sebastian, Jan Niklas Böhm, Fred A. Hamprecht, & Dmitry Kobak (July 2023). "From t-SNE to UMAP with contrastive learning". In: *International Conference on Learning Representations*.

Damrich, Sebastian & Fred Hamprecht (2021). "On UMAP's true loss function". In: *Advances in Neural Information Processing Systems*.

Defazio, Aaron, Francis Bach, & Simon Lacoste-Julien (2014). "SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives". In: *Advances in Neural Information Processing Systems*. Vol. 27.

Diaz-Papkovich, Alex, Luke Anderson-Trocmé, Chief Ben-Eghan, & Simon Gravel (2019). "UMAP reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts". In: *PLoS Genetics* 15.11.

Ding, Jiarui, Anne Condon, & Sohrab P Shah (2018). "Interpretable dimensionality reduction of single cell transcriptome data with deep generative models". In: *Nature Communications* 9.1, pp. 1–13.

Domingos, Pedro (Oct. 2012). "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10, pp. 78–87.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, & Neil Houlsby (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*.

Draganov, Andrew, Tyrus Berry, Jakob Rødsgaard Jørgensen, Katrine Scheel Nellemann, Ira Assent, & Davide Mottin (2022). "GiDR-DUN; Gradient Dimensionality Reduction–Differences and Unification". In: *arXiv*.

Draganov, Andrew, Jakob Jørgensen, Katrine Scheel, Davide Mottin, Ira Assent, Tyrus Berry, & Cigdem Aslay (2023). "ActUp: analyzing and consolidating t-SNE & UMAP". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 3651–3658.

Draganov, Andrew, Sharvaree Vadgama, & Erik J Bekkers (2024). "The Hidden Pitfalls of the Cosine Similarity Loss". In: *High-dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*.

Draganov, Andrew, Sharvaree Vadgama, Sebastian Damrich, Jan Niklas Böhm, Lucas Maes, Dmitry Kobak, & Erik Bekkers (Jan. 2025). "On the Importance of Embedding Norms in Self-Supervised Learning". In: *Under review*.

Dyer, Chris (2014). "Notes on Noise Contrastive Estimation and Negative Sampling". In: *arXiv preprint arxiv: 1410.8251*.

Eades, Peter (May 1984). "A Heuristic for Graph Drawing". In: *Congressus Numerantium* 42.11, pp. 149–160.

Eisenmann, Peter (2019). "Fast Visualization of High-Dimensional Data via Parallelized UMAP on GPUs". MA thesis. Karlsruhe Institue of Technology.

Fey, Matthias & Jan E. Lenssen (2019). "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Fisher, Ronald Aylmer (1936). "The use of multiple measurements in taxonomic problems". In: *Annals of Eugenics* 7.2, pp. 179–188.

Fruchterman, Thomas M. J. & Edward M. Reingold (1991). "Graph drawing by force-directed placement". In: *Software: Practice and Experience* 21.11, pp. 1129–1164.

Gao, Leo, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, & Connor Leahy (2020). *The Pile: An 800GB Dataset of Diverse Text for Language Modeling*.

Garrido, Quentin, Yubei Chen, Adrien Bardes, Laurent Najman, & Yann LeCun (2023). "On the duality between contrastive and non-contrastive self-supervised learning". In: *International Conference on Learning Representations*.

Goldberg, Yoav & Omer Levy (2014). "word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method". In: *arXiv preprint arxiv: 1402.3722*.

González Márquez, Rita, Luca Schmidt, Benjamin M. Schmidt, Philipp Berens, & Dmitry Kobak (June 2024). "The landscape of biomedical research". In: *Cell* 5.6.

Grobecker, Pascal, Thomas Sakoparnig, & Erik van Nimwegen (July 2024). "Identifying cell states in single-cell RNA-seq data at statistically maximal resolution". In: *PLOS Computational Biology* 20.7, pp. 1–21.

Grohe, Martin (2020). "word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data". In: *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 1–16.

Grover, Aditya & Jure Leskovec (2016). "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864.

Guo, Xiaojun, Yifei Wang, Zeming Wei, & Yisen Wang (2023). "Architecture Matters: Uncovering Implicit Mechanisms in Graph Contrastive Learning". In: *Thirty-seventh Conference on Neural Information Processing Systems*.

Gutmann, Michael & Aapo Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, pp. 297–304.

— (2012). "Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics." In: *Journal of Machine Learning Research* 13.2.

Hadsell, Raia, Sumit Chopra, & Yann LeCun (2006). "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* 2, pp. 1735–1742.

Hagberg, Aric, Pieter Swart, & Daniel S Chult (2008). *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.

Harrell, James A. & V. Max Brown (1992). "The World's Oldest Surviving Geological Map: The 1150 BC Turin Papyrus from Egypt". In: *The Journal of Geology* 100.1, pp. 3–18.

Harris, Charles R. et al. (Sept. 2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362.

Hassani, Kaveh & Amir Hosein Khasahmadi (2020). "Contrastive multi-view representation learning on graphs". In: *International Conference on Machine Learning*. PMLR, pp. 4116–4126.

He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, & Ross Girshick (2020). "Momentum Contrast for Unsupervised Visual Representation Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, & Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Vision and Pattern Recognition*, pp. 770–778.
Hinton, Geoffrey Everest & Sam Roweis (2002). "Stochastic neighbor embedding". In: *Advances in Neural Information Processing Systems* 15.
Hinton, Geoffrey Everest & Sam T Roweis (2003). "Stochastic neighbor embedding". In: *Advances in Neural Information Processing Systems*, pp. 857–864.
Hintze, Jerry L & Ray D Nelson (1998). "Violin plots: A box plot-density trace synergism". In: *The American Statistician* 52.2, pp. 181–184.
Hotelling, Harold (Sept. 1933). "Analysis of a complex of statistical variables into principal components". In: *Journal of Educational Psychology* 24, pp. 417–441.
Hu, Tianyang, Zhili Liu, Fengwei Zhou, Wenjia Wang, & Weiran Huang (2023). "Your Contrastive Learning is Secretly Doing Stochastic Neighbor Embedding". In: *International Conference on Learning Representations*.
Hu, Weihua, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, & Jure Leskovec (2020). "Open graph benchmark: Datasets for machine learning on graphs". In: *Advances in Neural Information Processing Systems* 33, pp. 22118–22133.
Hu, Yifan (2005). "Efficient, high-quality force-directed graph drawing". In: *Mathematica Journal* 10.1, pp. 37–71.
Hubert, Lawrence & Phipps Arabie (1985). "Comparing partitions". In: *Journal of Classification* 2 (1), pp. 193–218.
Hunter, John D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95.
Iverson, Kenneth Eugene (Oct. 1979). "Notation as a Tool of Thought". In: *Communications of the ACM* 23.8, pp. 444–465.
Jacomy, Mathieu, Tommaso Venturini, Sebastien Heymann, & Mathieu Bastian (2014). "ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software". In: *PloS One* 9.6.
Jing, Li, Pascal Vincent, Yann LeCun, & Yuandong Tian (2022). "Understanding Dimensional Collapse in Contrastive Self-supervised Learning". In: *International Conference on Learning Representations*.
Johnson, William B & Joram Lindenstrauss (1984). "Extensions of Lipschitz mappings into a Hilbert space". In: *Contemporary Mathematics* 26, pp. 189–206.
Jolicoeur, P. & J. E. Mosimann (Dec. 1960). "Size and shape variation in the painted turtle. A principal component analysis". In: *Growth*.
Jozefowicz, Rafal, Oriol Vinyals, Mike Schuster, Noam Shazeer, & Yonghui Wu (2016). "Exploring the limits of language modeling". In: *arXiv*.
Kalantidis, Yannis, Carlos Eduardo Rosar Kos Lassance, Jon Almazán, & Diane Larlus (2022). "TLDR: Twin Learning for Dimensionality Reduction". In: *Transactions on Machine Learning Research*.
Kamada, Tomihisa & Satoru Kawai (Apr. 1989). "An algorithm for drawing general undirected graphs". In: *Information Processing Letters* 31.1, pp. 7–15.
Kanton, Sabina, Michael James Boyle, Zhisong He, Malgorzata Santel, Anne Weigert, Fátima Sanchís-Calleja, Patricia Guijarro, Leila Sidow, Jonas Simon Fleck, Dingding Han, et al. (2019). "Organoid single-cell genomic atlas uncovers human-specific features of brain development". In: *Nature* 574.7778, pp. 418–422.
Karczewski, Konrad J, Laurent C Francioli, Grace Tiao, Beryl B Cummings, Jessica Alföldi, Qingbo Wang, Ryan L Collins, Kristen M Laricchia, Andrea Ganna,

Daniel P Birnbaum, et al. (2020). "The mutational constraint spectrum quantified from variation in 141,456 humans". In: *Nature* 581, pp. 434–443.
Le-Khac, Phuc H, Graham Healy, & Alan F Smeaton (2020). "Contrastive Representation Learning: A Framework and Review". In: *IEEE Access* 8, pp. 193907–193934.
Khosla, Megha, Vinay Setty, & Avishek Anand (2019). "A comparative study for unsupervised network representation learning". In: *IEEE Transactions on Knowledge and Data Engineering* 33.5, pp. 1807–1818.
Kingma, Diederik P & Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations*.
Kipf, Thomas N & Max Welling (2017). "Semi-supervised classification with graph convolutional networks". In: *International Conference for Learning Representations*.
Kirchhof, Michael, Enkelejda Kasneci, & Seong Joon Oh (2023). "Probabilistic Contrastive Learning Recovers the Correct Aleatoric Uncertainty of Ambiguous Inputs". In: *International Conference on Machine Learning*.
Kirchhof, Michael, Bálint Mucsányi, Seong Joon Oh, & Enkelejda Kasneci (2023). "URL: A Representation Learning Benchmark for Transferable Uncertainty Estimates". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
Kirchhof, Michael, Karsten Roth, Zeynep Akata, & Enkelejda Kasneci (2022). "A Non-isotropic Probabilistic Take on Proxy-based Deep Metric Learning". In: *European Conference on Computer Vision*.
Knyazev, A. V., M. E. Argentati, I. Lashuk, & E. E. Ovtchinnikov (2007). "Block Locally Optimal Preconditioned Eigenvalue Xolvers (BLOPEX) in Hypre and PETSc". In: *SIAM Journal on Scientific Computing* 29.5, pp. 2224–2239.
Kobak, Dmitry & Philipp Berens (2019). "The art of using t-SNE for single-cell transcriptomics". In: *Nature Communications* 10.1, pp. 1–14.
Kobak, Dmitry & George Linderman (2021). "Initialization is critical for preserving global data structure in both t-SNE and UMAP". In: *Nature Biotechnology* 39, pp. 156–157.
Kobak, Dmitry, George Linderman, Stefan Steinerberger, Yuval Kluger, & Philipp Berens (2019). "Heavy-tailed kernels reveal a finer cluster structure in t-SNE visualisations". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 124–139.
Kohonen, Teuvo (Jan. 1982). "Self-Organized Formation of Topologically Correct Feature Maps". In: *Biological Cybernetics* 43.1, pp. 59–69.
Koren, Yehuda, Robert Bell, & Chris Volinsky (Aug. 2009). "Matrix factorization techniques for recommender systems". In: *Computer* 42.8, pp. 30–37.
Korzybski, Alfred (Dec. 28, 1931). "A Non-Aristotelian System and Its Necessity for Rigour in Mathematics and Physics". In: *American Mathematical Society*.
Krizhevsky, Alex (Apr. 2009). *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto.
Krizhevsky, Alex, Ilya Sutskever, & Geoffrey E. Hinton (May 2017). "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6, pp. 84–90.
Kruiger, Johannes F, Paulo E Rauber, Rafael Messias Martins, Andreas Kerren, Stephen Kobourov, & Alexandru C Telea (2017). "Graph Layouts by t-SNE". In: *Computer Graphics Forum*. Vol. 36. Wiley Online Library, pp. 283–294.

Lause, Jan, Philipp Berens, & Dmitry Kobak (Oct. 2024). "The art of seeing the elephant in the room: 2D embeddings of single-cell data do make sense". In: *PLOS Computational Biology* 20.10, pp. 1–5.
Lawton, William H. & Edward A. Sylvestre (1971). "Self Modeling Curve Resolution". In: *Technometrics* 13.3, pp. 617–633.
LeCun, Yann, Léon Bottou, Yoshua Bengio, & Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.
Lee, Daniel D. & H. Sebastian Seung (Oct. 1999). "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755, pp. 788–791.
Lee, Namkyeong, Junseok Lee, & Chanyoung Park (2022). "Augmentation-free self-supervised learning on graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36, pp. 7372–7380.
Leow, Yao Yang, Thomas Laurent, & Xavier Bresson (2019). "GraphTSNE: a visualization technique for graph-structured data". In: *Representation Learning on Graphs and Manifold Workshop at the International Conference for Learning Representations*.
Levy, Omer & Yoav Goldberg (2014). "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems*. Vol. 27, pp. 2177–2185.
Li, Haifeng, Jun Cao, Jiawei Zhu, Qinyao Luo, Silu He, & Xuying Wang (2023). "Augmentation-Free Graph Contrastive Learning of Invariant-Discriminative Representations". In: *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11.
Linderman, George, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, & Yuval Kluger (2019). "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data". In: *Nature Methods* 16.3, pp. 243–245.
Linderman, George & Stefan Steinerberger (2019). "Clustering with t-SNE, provably". In: *SIAM Journal on Mathematics of Data Science* 1.2, pp. 313–332.
Loshchilov, Ilya & Frank Hutter (2017). "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *International Conference on Learning Representations*.
von Luxburg, Ulrike, Mikhail Belkin, & Olivier Bousquet (2008). "Consistency of spectral clustering". In: *The Annals of Statistics*, pp. 555–586.
Ma, Zhuang & Michael Collins (2018). "Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3698–3707.
van der Maaten, Laurens (2009). "Learning a parametric embedding by preserving local structure". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 384–391.
— (2014). "Accelerating t-SNE using tree-based algorithms". In: *The Journal of Machine Learning Research* 15.1, pp. 3221–3245.
van der Maaten, Laurens & Geoffrey Everest Hinton (Nov. 2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605.
Mahalanobis, Prasanta Chandra, Dhirendra Nath Majumdar, & Calyampudi Radhakrishna Rao (Mar. 1949). "Anthropometric survey of the United Provinces, 1941: a statistical study." In: *Sankhyā: The Indian Journal of Statistics* 9. Ed. by Prasanta Chandra Mahalanobis, pp. 89–324.
Mallock, R. R. M. (May 3, 1933). "An electrical calculating machine". In: *Proceedings of the Royal Society of London, Series A* 140 (841).

McInnes, Leland, John Healy, & Steve Astels (2017). "hdbscan: Hierarchical density based clustering". In: *The Journal of Open Source Software* 2.11, p. 205.
McInnes, Leland, John Healy, & James Melville (2018). "UMAP: Uniform manifold approximation and projection for dimension reduction". In: *arXiv:1802.03426*.
Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, & Jeff Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Neural Information Processing Systems*. Vol. 26, pp. 3111–3119.
Mirkes, Evgeny M., Jeza Allohibi, & Alexander Gorban (Sept. 2020). "Fractional Norms and Quasinorms Do Not Help to Overcome the Curse of Dimensionality". In: *Entropy* 22.10.
Mitrovic, Jovana, Brian McWilliams, & Melanie Rey (2020). "Less can be more in contrastive learning". In: *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, pp. 70–75.
Mnih, Volodymyr et al. (Feb. 2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533.
Moore, Gordon Earle (Apr. 19, 1998). "Cramming More Components Onto Integrated Circuits". In: *Proceedings of the IEEE* 86.1, pp. 82–85.
Narayan, Ashwin, Bonnie Berger, & Hyunghoon Cho (2021). "Assessing single-cell transcriptomic variability through density-preserving data visualization". In: *Nature Biotechnology*, pp. 1–10.
Nicholas, Clément, Yvan Pailler, Pierre Stéphan, Julie Pierson, Laurent Aubry, Bernard Le Gall, Vincent Lacombe, & Joël Rolet (Apr. 2021). "La carte et le territoire : la dalle gravée du Bronze ancien de Saint-Bélec (Leuhan, Finistère)". In: *Bulletin de la Société préhistorique française* 118, pp. 99–146.
Noack, Andreas (2007). "Energy Models for Graph Clustering". In: *Journal of Graph Algorithms and Applications* 11.2, pp. 453–480.
— (2009). "Modularity clustering is force-directed layout". In: *Physical Review E* 79.2.
Nolet, Corey J, Victor Lafargue, Edward Raff, Thejaswi Nanditale, Tim Oates, John Zedlewski, & Joshua Patterson (2021). "Bringing UMAP Closer to the Speed of Light with GPU Acceleration". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 1, pp. 418–426.
Northcutt, Curtis G, Anish Athalye, & Jonas Mueller (2021). "Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.
Nozawa, Kento & Issei Sato (2021). "Understanding Negative Samples in Instance Discriminative Self-supervised Representation Learning". In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 5784–5797.
Nwabufo, Ifeoma Veronica, Jan Niklas Böhm, Philipp Berens, & Dmitry Kobak (2024). "Self-supervised Visualisation of Medical Image Datasets". In: *arXiv*.
van den Oord, Aaron, Yazhe Li, & Oriol Vinyals (2018). "Representation Learning with Contrastive Predictive Coding". In: *arXiv*.
Oskolkov, Nikolay (2019). *How Exactly UMAP Works*. Blog post on Towards Data Science.
Paatero, Pentti, Unto Tapper, Pasi Aalto, & Markku Kulmala (1991). "Matrix factorization methods for analysing diffusion battery data". In: *Journal of Aerosol Science* 22. European Aerosol Conference, pp. 273–276. ISSN: 0021-8502.
Packer, Jonathan S, Qin Zhu, Chau Huynh, Priya Sivaramakrishnan, Elicia Preston, Hannah Dueck, Derek Stefanik, Kai Tan, Cole Trapnell, Junhyong Kim, Robert

H. Waterston, & John I. Murray (2019). "A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution". In: *Science* 365.6459, pp. 1265–1274.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32.

Pearson, Karl (1901). "On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11, pp. 559–572.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. (2011). "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Perozzi, Bryan, Rami Al-Rfou, & Steven Skiena (2014). "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710.

Pitsianis, Nikos, Alexandros-Stavros Iliopoulos, Dimitris Floros, & Xiaobai Sun (2019). "Spaceland Embedding of Sparse Stochastic Graphs". In: *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8.

Poličar, Pavlin Gregor, Martin Strazar, & Blaz Zupan (2019). "openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding". In: *bioRxiv:731877*.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, & Ilya Sutskever (July 2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *International Conference on Machine Learning*, pp. 8748–8763.

Rao, Calyampudi Radhakrishna (1948). "The Utilization of Multiple Measurements in Problems of Biological Classification". In: *Journal of the Royal Statistical Society* 10.2, pp. 159–193.

Rokhlin, Vladimir (Sept. 1985). "Rapid solution of integral equations of classical potential theory". In: *Journal of Computational Physics* 60.2.

Roth, Karsten, Timo Milbich, & Bjorn Ommer (2020). "PADS: Policy-Adapted Sampling for Visual Similarity Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6568–6577.

Roweis, Samuel T. & Lawrence K. Saul (Dec. 2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* 290.5500, pp. 2323–2326.

Ruder, Sebastian (2016). *On word embeddings - Part 2: Approximating the Softmax*. http://ruder.io/word-embeddings-softmax. Accessed: 2022-05-17.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, & Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3, pp. 211–252.

Sainburg, Tim, Leland McInnes, & Timothy Q Gentner (2021). "Parametric UMAP Embeddings for Representation and Semisupervised Learning". In: *Neural Computation* 33.11, pp. 2881–2907.

Sammon, John W. (1969). "A Nonlinear Mapping for Data Structure Analysis". In: *IEEE Transactions on Computers* C-18.5, pp. 401–409.

Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, & Liang-Chieh Chen (2018). "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520.

Schmidt, Benjamin (2018). "Stable random projection: lightweight, general-purpose dimensionality reduction for digitized libraries". In: *Journal of Cultural Analytics*.

Schmors, Lisa, Dominic Gonschorek, Jan Niklas Böhm, Yongrong Qiu, Na Zhou, Dmitry Kobak, Andreas Tolias, Fabian Sinz, Jacob Reimer, Katrin Franke, Sebastian Damrich, & Philipp Berens (Jan. 2025). "TRACE: Contrastive learning for multi-trial time series data in neuroscience". In: *Under review*.

Schönemann, Peter H. (1966). "A generalized solution of the orthogonal procrustes problem". In: *Psychometrika*.

Shi, Yujun, Jian Liang, Wenqing Zhang, Vincent Tan, & Song Bai (2023). "Towards Understanding and Mitigating Dimensional Collapse in Heterogeneous Federated Learning". In: *International Conference on Learning Representations*.

Snow, John (1855). *On the Mode of Communication of Cholera*. 2nd ed. Princess Street, Soho, London: John Churchill.

Steiner, Andreas Peter, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, & Lucas Beyer (2022). "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers". In: *Transactions on Machine Learning Research*. ISSN: 2835-8856.

Szubert, Benjamin, Jennifer E Cole, Claudia Monaco, & Ignat Drozdov (2019). "Structure-preserving visualisation of high dimensional single-cell datasets". In: *Scientific Reports* 9.1, pp. 1–10.

Tan, Mingxing & Quoc Le (June 2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri & Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research, pp. 6105–6114.

Tang, Jian, Jingzhou Liu, Ming Zhang, & Qiaozhu Mei (2016). "Visualizing large-scale and high-dimensional data". In: *Proceedings of the 25th International Conference on World Wide Web*, pp. 287–297.

Tarin, Clanuwat, B Mikel, K Asanobu, L Alex, Y Kazuaki, & H David (2018). "Deep Learning for Classical Japanese Literature". In: *Proceedings of 2018 Workshop on Machine Learning for Creativity and Design (Thirty-second Conference on Neural Information Processing Systems)*. Vol. 3.

Tenenbaum, Joshua B, Vin De Silva, & John C Langford (Dec. 2000). "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500, pp. 2319–2323.

Tenenbaum, Joshua B. (1997). "Mapping a manifold of perceptual observations". In: *Advances in Neural Information Processing Systems*.

Thakoor, Shantanu, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, & Michal Valko (2021). "Large-scale representation learning on graphs via bootstrapping". In: *arXiv preprint arXiv:2102.06514*.

Tian, Yonglong, Dilip Krishnan, & Phillip Isola (2020). "Contrastive Multiview Coding". In: *Proceedings of the European Conference on Computer Vision*. Springer, pp. 776–794.

Tian, Yuandong (2022). "Understanding Deep Contrastive Learning via Coordinate-wise Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, & Kyunghyun Cho.

Torgerson, Warren S. (1952). "Multidimensional Scaling: I. Theory and Method". In: *Psychometrika* 17.4, pp. 401–419.

Treisman, Anne M. & Garry Gelade (1980). "A feature-integration theory of attention". In: *Cognitive Psychology* 12.1, pp. 97–136.

Trivedi, Puja, Ekdeep Singh Lubana, Yujun Yan, Yaoqing Yang, & Danai Koutra (2022). "Augmentations in graph contrastive learning: Current methodological flaws & towards better practices". In: *Proceedings of the ACM Web Conference 2022*, pp. 1538–1549.

Tufte, Edward Rolf (May 2001). *The Visual Display of Quantitative Information*. 2nd ed. ISBN: 0-961-39214-2.

Tutte, William Thomas (1963). "How to Draw a Graph". In: *Proceedings of the London Mathematical Society* s3-13.1, pp. 743–767.

Velickovic, Petar, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, & R Devon (2019). "Deep graph infomax". In: *International Conference for Learning Representations* 2.3, p. 4.

Vishnubhotla, Ankit, Charlotte Loh, Akash Srivastava, Liam Paninski, & Cole Hurwitz (2023). "Towards robust and generalizable representations of extracellular data using contrastive learning". In: *Advances in Neural Information Processing Systems* 37.

Wagner, Daniel E., Caleb Weinreb, Zach M. Collins, James A. Briggs, Sean G. Megason, & Allon M. Klein (2018). "Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo". In: *Science* 360.6392, pp. 981–987.

Wang, Minjie, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. (2019). "Deep graph library: A graph-centric, highly-performant package for graph neural networks". In: *arXiv preprint arXiv:1909.01315*.

Wang, Tongzhou & Phillip Isola (July 2020). "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research, pp. 9929–9939.

Wang, Yifei, Qi Zhang, Tianqi Du, Jiansheng Yang, Zhouchen Lin, & Yisen Wang (2023). "A message passing perspective on learning dynamics of contrastive learning". In: *International Conference on Learning Representations*.

Wang, Yingfan, Haiyang Huang, Cynthia Rudin, & Yaron Shaposhnik (2021). "Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization." In: *Journal of Machine Learning Research* 22.201, pp. 1–73.

Wattenberg, Martin, Fernanda Viégas, & Ian Johnson (2016). "How to Use t-SNE Effectively". In: *Distill*.

Willeke, Konstantin F., Kelli Restivo, Katrin Franke, Arne F. Nix, Santiago A. Cadena, Tori Shinn, Cate Nealley, Gabrielle Rodriguez, Saumil S. Patel, Alexander S. Ecker, Fabian H Sinz, & Andreas Savas Tolias (2023). "Deep learning-driven characterization of single cell tuning in primate visual area V4 unveils topological organization". In: *bioRxiv*.

Wu, Chao-Yuan, R Manmatha, Alexander J Smola, & Philipp Krahenbuhl (2017). "Sampling Matters in Deep Embedding Learning". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2840–2848.

Wu, Zhirong, Yuanjun Xiong, Stella X Yu, & Dahua Lin (2018). "Unsupervised Feature Learning via Non-Parametric Instance Discrimination". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742.

Yang, Zhirong, Yuwei Chen, Denis Sedov, Samuel Kaski, & Jukka Corander (2023). "Stochastic cluster embedding". In: *Statistics and Computing* 33.1, p. 12.

Yang, Zhirong, Irwin King, Zenglin Xu, & Erkki Oja (2009). "Heavy-tailed symmetric stochastic neighbor embedding". In: *Advances in Neural Information Processing Systems*, pp. 2169–2177.

Yang, Zhirong, Jaakko Peltonen, & Samuel Kaski (2013). "Scalable optimization of neighbor embedding for visualization". In: *International Conference on Machine Learning*, pp. 127–135.

Zang, Zelin, Siyuan Li, Di Wu, Ge Wang, Kai Wang, Lei Shang, Baigui Sun, Hao Li, & Stan Z. Li (2022). "DLME: Deep Local-Flatness Manifold Embedding". In: *Computer Vision – ECCV 2022*, pp. 576–592.

Zbontar, Jure, Li Jing, Ishan Misra, Yann LeCun, & Stephane Deny (July 2021). "Barlow Twins: Self-Supervised Learning via Redundancy Reduction". In: *International Conference on Machine Learning*. Ed. by Marina Meila & Tong Zhang. Vol. 139. PMLR, pp. 12310–12320.

Zhang, Hengrui, Qitian Wu, Yu Wang, Shaofeng Zhang, Junchi Yan, & Philip S Yu (2022). "Localized Contrastive Learning on Graphs". In: *arXiv preprint arXiv:2212.04604*.

Zhang, Hengrui, Qitian Wu, Junchi Yan, David Wipf, & Philip S Yu (2021). "From canonical correlation analysis to self-supervised graph neural networks". In: *Advances in Neural Information Processing Systems* 34, pp. 76–89.

Zhang, Jesse et al. (2025). "Tahoe-100M: A Giga-Scale Single-Cell Perturbation Atlas for Context-Dependent Gene Function and Cellular Modeling". In: *bioRxiv*.

Zhong, Fahai, Mingliang Xue, Jian Zhang, Fan Zhang, Rui Ban, Oliver Deussen, & Yunhai Wang (2023). "Force-directed graph layouts revisited: a new force based on the t-Distribution". In: *IEEE Transactions on Visualization and Computer Graphics*.

Zhu, Minfeng, Wei Chen, Yuanzhe Hu, Yuxuan Hou, Liangjun Liu, & Kaiyuan Zhang (2020a). "DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction". In: *IEEE Transactions on Visualization and Computer Graphics* 27.2, pp. 1666–1676.

Zhu, Yanqiao, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, & Liang Wang (2020b). "Deep graph contrastive representation learning". In: *arXiv preprint arXiv:2006.04131*.

— (2021). "Graph contrastive learning with adaptive augmentation". In: *Proceedings of the Web Conference 2021*, pp. 2069–2080.

Zimmermann, Roland S., Yash Sharma, Steffen Schneider, Matthias Bethge, & Wieland Brendel (2021). "Contrastive Learning Inverts the Data Generating Process". In: Proceedings of Machine Learning Research 139, pp. 12979–12990.