**Department of Physics and Astronomy**
**Heidelberg University**

Bachelor Thesis in Physics
submitted by

**Finn Magnus Prem**

born in Stockholm (Sweden)

**2024**

# Connecting The Dots: Low Loss Paths Between Neighbor Embeddings

This Bachelor Thesis has been carried out by Finn Magnus Prem at the
Interdisciplinary Center for Scientific Computing in Heidelberg
under the supervision of
Prof. Fred Hamprecht

# Abstract

UMAP and t-SNE belong to the most common dimension reduction techniques for the visualization of high-dimensional data across various domains. Previous work focused largely on finding the best minima of these algorithms. We connect different minima on the loss surface, using the AutoNEB algorithm to optimize for a low loss path. For UMAP, we find paths with a loss continuously within the range of usual fluctuations between different minima. For t-SNE the loss rises with increasing distance from the initial minima and is not brought down fully even by further optimization. Additionally, we introduce a new way to visualize the loss contribution of individual points, which provides an intuitive overview over the local embedding quality.

UMAP und t-SNE gehören zu den gängigsten Verfahren zur Dimensionsreduktion und Visualisierung hochdimensionaler Datensätze in verschiedensten Gebieten. Bisherige Arbeiten konzentrierten sich hauptsächlich darauf, die besten Minima dieser Algorithmen zu finden. Wir verbinden die verschiedenen Minima mit einem Pfad durch die Loss-Landschaft, der mithilfe des AutoNEB Algorithmus optimiert wird. Für UMAP finden wir Pfade mit einem niedrigen Loss durchgehend innerhalb der üblichen Schwankungen zwischen verschiedenen Minima. Bei t-SNE stiegt der Loss mit zunehmendem Abstand von den ursprünglichen Minima und wird auch durch längere Optimierung kaum weiter reduziert. Darüber hinaus stellen wir eine neue Methode zur Visualisierung des Beitrags jedes einzelnen Punktes zum Loss vor, die einen intuitiven Überblick über die lokale Qualität der Visualisierung bietet.

# Acknowledgements

I would like to thank Sebastian Damrich and Dmitry Kobak for the interesting discussions, helpful ideas and pleasant company while supervising my thesis;

Felix Draxler for his kind support in getting to know his code;

my father for his patience and important feedback after reading my drafts for the umpteenth time;

Fred Hamprecht for his tireless –and luckily very successful– efforts to ensure a great work atmosphere for everyone within the lab;

and all members of the group for making this an unforgettable time in the best way possible!

# Contents

# 1 Introduction

Dimension reduction techniques have become an important tool to get an intuitive visual overview of structures in high-dimensional data of any kind. A multitude of different techniques have been developed, with UMAP (McInnes et al., 2018) and t-SNE (van der Maaten & Hinton, 2008) belonging to the most common. Their applications span a wide range of topics, including single-cell biology (Kobak & Berens, 2019), particle physics (Teixeira et al., 2022), bioinformatics (Trozzi et al., 2021), and quantum physics (Matty et al., 2021). Along with their wide-spread use, continuous research has been undertaken to better understand these techniques. This ranges from placing the different methods in a common general framework (Böhm et al., 2022) to surprising results, even revoking fundamental assumptions about their inner workings (Damrich & Hamprecht, 2021). Previous work also focused on different initialization possibilities in order to reach the best minima (Kobak & Linderman, 2021), while the relation of these minima to each other remained largely unexplored.

This work tries to fill that gap with an in-depth study of the connectivity of the minima of UMAP and t-SNE on the loss surface. Given two different embeddings, the aim is to find an interpolation between them, along which each embedding retains a high visualization quality. As the loss of an embedding reflects its quality, this can be formulated as finding a connection that maintains a low loss along the whole path. To find low-loss connections between minima of the respective dimension reduction technique, the Automated Nudged Elastic Band (AutoNEB) (Kolsbjerg et al., 2016) algorithm was used. It has already been successfully applied to connect minima of the loss surface of neural networks (Draxler et al., 2018).

For UMAP, low-loss paths connecting two minima are constructed, with a loss continuously within the usual range of fluctuations between the losses of different embeddings (Section 4.2). These findings are put into context by further discussing how the properties of UMAP's loss function and its embedding characteristics enable such connections.

For t-SNE, the embeddings on the path had a significantly higher loss compared to the usual variance between different minima (Section 4.3). The loss increased, the further the embeddings deviated from the minima. This was explicitly studied on a connection starting at a minimum and then moving gradually further away from it along a manually initialized path (Section 4.3.3).

Additionally, a new method is proposed for intuitive visualization of the embedding quality by coloring each point according to its individual contribution to the loss (Section 4.1.3).

## 1.1 Data And Code Availability

The source code used for this work is available on GitHub[1]. The AutoNEB code used here is based on Felix Draxler's work, see GitHub[2], originally written to connect Neural Network minima (Draxler et al., 2018). Animations of all connections as .gif can be viewed in the EU Open Research Repository[3].

[1] https://github.com/premfi/NE-AutoNEB
[2] https://github.com/fdraxler/PyTorch-AutoNEB
[3] https://doi.org/10.5281/zenodo.11200908

# 2 Background

## 2.1 Dimension Reduction

Real-world data are often high-dimensional, meaning that each individual data point comprises a multitude of features. Visualizing these data in a meaningful way is a non-trivial task, as no direct graphical representation of these high-dimensional structures exists. Instead, techniques to reduce these high-dimensional data into a low dimensional space (usually 2D or 3D) need to be used, all while retaining information about as many relevant properties of the original data as possible. In the case of reduction to 2 dimensions, the result can then be portrayed as a conventional image.

The two examples of such techniques that were studied in this work, are **Uniform Manifold Approximation and Projection (UMAP)** (McInnes et al., 2018) and **t-distributed stochastic neighbor embedding (t-SNE)** (van der Maaten & Hinton, 2008). Both are non-linear dimension reduction techniques that are used in current research to gain an understanding of hidden structures in datasets from various domains, including single-cell biology (Kobak & Berens, 2019), particle physics (Teixeira et al., 2022), bioinformatics (Trozzi et al., 2021), and quantum physics (Matty et al., 2021). As an example, the C. Elegans single cell dataset used in this work contains the RNA sequences of over 80000 cells of more than 30 different cell types (see Section 3.2).

## 2.2 UMAP

UMAP works by constructing a nearest neighbor graph of the high-dimensional data and using it to define similarities between each pair of points. The embedding is then optimized by pulling similar points together and repelling points from each other, the latter regardless of similarity. UMAP normally uses a sampling-based approach to optimize the loss more efficiently. The implementation used in this work instead directly optimizes the loss function Eq. (2.1) without sampling. This is only possible in an efficient way due to the use of symbolic matrices (see Section 3.3.2). This section outlining the UMAP algorithm and its actual loss function is based on (McInnes et al., 2018) and (Damrich & Hamprecht, 2021) respectively.

The UMAP algorithm consists of the following high-level steps, each of which is explained in more detail in the subsequent sections:

- **Construct** a nearest neighbor graph of the high-dimensional data.
- **Initialize** the low-dimensional embedding.
- **Optimize** the embedding using a loss function, comprising an attractive and a repulsive term.

### 2.2.1 Nearest Neighbor Graph

The original publication (McInnes et al., 2018) described representing the connectivity in terms of fuzzy sets, i.e., defining a similarity $\mu_{i \to j}$ between points $i$ and $j$, approximately reflecting to what degree point $j$ counts as a nearest neighbor to point $i$. Later work (Damrich & Hamprecht, 2021) showed that the actual UMAP implementation disregarded these nuances and effectively binarized the similarities, resulting in only the information of a conventional nearest neighbor graph being used, where points are either nearest neighbors or not. After its construction, the directed similarity $\mu_{i \to j}$ needs to be symmetrized to obtain an undirected nearest neighbor graph. This is achieved using $\mu_{ij} = \mu_{i \to j} + \mu_{j \to i} - \mu_{i \to j}\mu_{j \to i}$.

### 2.2.2 Initialization

#### Default

The default initialization uses Laplacian Eigenmaps to already place the points in a somewhat meaningful relation to each other before starting the optimization. For details about the default initialization, refer to the original publication, (McInnes et al., 2018, page 19), "Algorithm 4". Embeddings initialized this way often result in visualizations that are subjectively considered qualitatively good. This judgment can include the absence of unwanted features like clusters split apart and entangled or twisted clusters, though an objective measurement of embedding quality is difficult to introduce due to the vastly different properties of various datasets, for all of which an expressive visualization is to be achieved. One side effect of the default initialization is the similar outcomes after the optimization. Although the optimization includes some randomness, embeddings initialized this way usually result in very comparable global structures with only minor local differences.

#### Random

Alternatively, embeddings can be initialized randomly. This yields results that can differ a great deal in the overall structure, but also in the subjective visualization quality. In this work,

random initialization is used, except for UMAP on C. Elegans, in order to obtain different embeddings whose connections would be non-trivial.

### 2.2.3 Optimization

After initialization, a loss for the current embedding is computed, and its gradient is used to move the embedded points in a favorable direction. This is repeated for every optimization step, iteratively decreasing the loss. Given the positions of the embedded points $\{e_i\}$ as well as the high-dimensional similarities $\{\mu_{ij}\}$ for each pair of points, the loss can be computed. Its closed form formula

$$\mathcal{L}\left(\{e_i\}|\{\mu_{ij}\}\right) = 2 \sum_{1\leq i<j\leq n} \Big( \underbrace{\mu_{ij} \cdot \mathcal{L}^a_{ij}}_{\text{attractive}} + \underbrace{\frac{(d_i + d_j)m}{2n} \cdot \mathcal{L}^r_{ij}}_{\text{repulsive}} \Big) \tag{2.1}$$

was calculated by (Damrich & Hamprecht, 2021, Eq. 12). It consists of an attractive and a repulsive term, with an attractive $\mathcal{L}^a_{ij} = -\log\left(\phi(e_i, e_j)\right)$ and a repulsive loss function $\mathcal{L}^r_{ij} = -\log\left(1 - \phi(e_i, e_j)\right)$. When referring to the "contribution to the attractive/repulsive loss" later, the whole respective term in Eq. (2.1) is meant, including the prefactors. The attractive and repulsive loss functions depend on the positions of the embedded points, converted to a similarity between each pair by the low-dimensional similarity function $\phi(e_i, e_j) := \phi(||e_i - e_j||) = \left(1 + ||e_i - e_j||^2\right)^{-1}$. The gradients of $\mathcal{L}^a_{ij}$ and $\mathcal{L}^r_{ij}$ are calculated in each step in order to find the direction in which the loss decreases the most. The low-dimensional similarity function $\phi$ originally included two shape parameters $\phi(x; a, b) = \left(1 + ax^{2b}\right)^{-1}$. Here, both parameters were set to $a = b = 1$ throughout all experiments. $d_i = \sum_{j=1}^{n} \mu_{ij}$ denotes the degree of each node, which is roughly the number of neighbors of that node. $m = 5$ is the number of points used in negative sampling, a technique used to sample points on which to perform repulsion. Negative sampling is the main reason for the difference between UMAP's purported loss function as expected by its creators (McInnes et al., 2018) and its actual loss function Eq. (2.1), found by (Damrich & Hamprecht, 2021) and used throughout this work. The number of data points is represented by $n$.

The embedding $e_i$ is then optimized using gradient descent. Edges between two points are sampled according to their high-dimensional similarity $\mu_{ij}$. The points $e_i$ and $e_j$ are then pulled together according to the gradient of the attractive loss function $\mathcal{L}^a_{ij}$ and $e_i$ is repelled from $m = 5$ randomly sampled points according to the gradient of the repulsive loss function $\mathcal{L}^r_{ij}$. Usually, there are 500 optimization steps, with the learning rate decaying linearly until 0. The learning rate is a factor with which the gradient is multiplied before its application. Learning rate decay helps to reduce the size of the changes in each step when approaching a good embedding.

## 2.3 t-SNE

t-SNE approaches dimension reduction by interpreting the high-dimensional as well as the low-dimensional data in terms of probability distributions. For each pair of points, a similarity is defined, depending on their distance. The symmetrized and normalized similarities form a probability distribution. For the high-dimensional similarities, a Gaussian is used, while the low-dimensional similarity function is the Student's t-Distribution. The Kullback-Leibler divergence (KL divergence) between the high- and low-dimensional probability distributions serves as a loss function. The embedding is optimized using gradient descent to minimize the KL divergence. The description of the t-SNE algorithm in this section is based on the openTSNE documentation (Poličar, 2023).

The t-SNE algorithm consists of the following high-level steps, each of which is explained in more detail in the subsequent sections:

- **Construct** the high-dimensional similarities as Gaussian probability distribution $\mathbf{P}$.
- **Initialize** the low-dimensional embedding.
- **Construct** the low-dimensional similarities as Student's t-distribution $\mathbf{Q}$.
- **Optimize** the embedding by minimizing the Kullback-Leibler divergence between $\mathbf{P}$ and $\mathbf{Q}$.

### 2.3.1 High-Dimensional Gaussian Distribution

The original, high-dimensional data points $\mathbf{x}_i$ are transformed into the high-dimensional similarities $p_{i|j}$ by using a Gaussian distribution

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2\right)}, \tag{2.2}$$

where $\sigma$ is chosen according to the density of the region around $\mathbf{x}_i$ such that the number of nearest neighbors is approximately the same for each point. This is essentially what is controlled by the perplexity hyperparameter.

The conditional probabilities $p_{i|j}$ are symmetrized to obtain joint probabilities

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}. \tag{2.3}$$

### 2.3.2 Initialization

#### Default

By default, the low-dimensional embeddings are initialized using principal component analysis (PCA). This technique finds the (high-dimensional) direction in which the original data vary the most. Then, a second direction maximizing the variance is found, under the condition of being orthogonal to the first. These two directions of biggest variance are then used as the x- and y-axes of the embedding and the high-dimensional points are projected onto this plane. For datasets with a simple inherent structure, this alone may already yield a sufficiently clear visualization.

#### Random

Alternatively, the embedding can be initialized randomly. This results in embeddings with a much larger variance and major differences between them. However, these embeddings may feature unwanted attributes, such as clusters splitting apart more often, compared to initialization using PCA. All t-SNE embeddings used in this work were initialized randomly, as minima with non-trivial connections between them are desirable.

### 2.3.3 Low-Dimensional Student's t-Distribution

The low-dimensional embedding can never reproduce the high-dimensional similarities perfectly. In high dimensions, points can have more close neighbors, while these neighboring points do not necessarily lie close to one another. In the low-dimensional embedding, it is necessary to either crowd such points closer together to preserve their proximity to their shared nearest neighbors or to spread them out further to preserve the distance between them, sacrificing some neighbor relations. t-SNE uses the second approach, allowing some points to lie further apart than in the high-dimensional data. This is implemented by choosing the Student's t-distribution

$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}} \tag{2.4}$$

as the similarity function in the embedding space, which has thicker tails than a Gaussian, thereby reducing the penalty for points lying too far apart.

### 2.3.4 Optimization

In order to achieve a faithful embedding where the low-dimensional similarities $q_{ij}$ match the high-dimensional similarities well, the Kullback-Leibler divergence (KL divergence)

$$\mathcal{L}(\mathbf{y}|\mathbf{x}) = KL(\mathbf{P}\|\mathbf{Q}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.5}$$

between $\mathbf{P}$ and $\mathbf{Q}$ is minimized.

This corresponds to a loss function $\mathcal{L}$ for the embedded points $\mathbf{y}$, given the original high-dimensional data points $\mathbf{x}$. In each step, the gradient of this loss function $\mathcal{L}(\mathbf{y}|\mathbf{x})$ is calculated, and the embedding is optimized using gradient descent. This means that each time, a sample of embedded points is moved in the direction where the loss is decreasing the most. The gradient is multiplied by a learning rate before application. The learning rate is a hyperparameter determining the step size of each optimization step.

Default t-SNE uses an optimization strategy called early exaggeration, where the attractive force is increased in the first phase of the optimization. This helps points with a high similarity to agglomerate to form connected clusters. The early exaggeration phase is usually set to 250 iterations. After this initial phase, the factor of the attractive force is reduced to its normal value again, and another 500 iterations are executed, concluding the optimization.

## 2.4 AutoNEB

The explanations in this section are largely based on (Draxler et al., 2018), as their AutoNEB implementation formed the basis for most experiments in this work. The AutoNEB algorithm can be used to connect minima on arbitrary loss surfaces. It works by initializing a chain of pivots between the minima and subsequently optimizing the pivot along the chain until a path with reasonably low loss is created. Pivots are optimized using gradient descent, though their movement is restricted to directions perpendicular to the chain. To prevent pivots from avoiding peaks by sliding apart, target distances are defined, which fix the relative distances between pivots. The necessary sampling density in the different regions along the chain is not known beforehand. This is addressed by splitting the optimization into several smaller phases, between which new pivots can be inserted into the chain in regions where the loss between current pivots rises above a threshold.

The AutoNEB algorithm consists of the following high-level steps, each of which is explained in more detail in the subsequent sections:

- **Find** minima on the loss surface.

- **Initialize** the chain connecting the minima.
- **Redistribute** the pivots along the chain.
- **Calculate** the gradient of the loss for each pivot.
- **Move** pivots perpendicular to the direction of the chain.
- **Insert** pivots where closer sampling is needed (optional).
- **Analyze** the loss along the path.

### 2.4.1 Find Minima

In the first step, several minima on the loss surface in question need to be found. In the original use case of neural networks, this is done by training several instances of the network until a reasonably low loss is achieved. Here, the minima are found by running the default UMAP or t-SNE algorithms several times with random initialization.

### 2.4.2 Initialization

Typically, the chain is initialized by linearly interpolating between the two minima, which should be connected, and inserting pivots with equal spacing. The number of initial pivots along the chain can be chosen at will, especially since more points can be inserted later in relevant areas. In (Draxler et al., 2018), 3 initial pivots were used (5 points in total, when including the end points).

### 2.4.3 Redistribution

Before each update step, the pivots are redistributed along the chain to match specified target distances. This prevents pivots from sliding apart while trying to reach a lower-loss region. This would lead to a gap in the path, skipping over peaks in the loss landscape instead of finding good paths around them. Especially in these high-loss regions, further optimization would be needed the most. Being able to specify target distances allows for unequal spacing between the points, which is especially needed when additional points are inserted later to sample a certain region more densely.

### 2.4.4 Gradient Calculation

For each pivot, the (approximate) gradient of the loss function is calculated. It is then projected to act only perpendicular to the path. The default AutoNEB implementation uses stochastic

gradient descent (SGD), although other optimizers could be used as well. The gradient calculation for one step can be easily parallelized over the pivots, as they do not depend on each other.

### 2.4.5 Position Update

The position of each pivot along the chain is updated using the perpendicular loss force as calculated before. The perpendicular gradient is multiplied by a learning rate before being added to the current position of the pivot. The learning rate can be adjusted throughout the optimization procedure. A higher learning rate at the beginning allows points to escape local minima, while a lower learning rate may be sensible towards the end of the optimization to prevent overshooting good solutions.

### 2.4.6 Pivot Insertion

After several steps of optimization, neighboring pivots may lie on opposite sides of a peak in the loss landscape between them. Further optimization of these pivots will not improve the connection between them, as their gradient is largely unaffected by the peak. To find a path between them that passes around the peak, further pivots need to be inserted to sample the path more closely. During the further optimization, the new pivots will descend from the peak, hopefully finding a path with lower loss.

The implementation of (Draxler et al., 2018) offers two options for inserting additional pivots:

- Inserting a specified number of new pivots between each existing pair of pivots. The new pivots are equally spaced along the linear interpolation between existing pivots.
- Only inserting new pivots where the loss of the linear interpolation between existing pivots rises above a threshold. Between each pair of pivots, at most one new pivot is inserted. The maximum number of total new pivots can be specified.

### 2.4.7 Analysis

After each optimization phase, before the possible insertion of further pivots, the loss along the path is analyzed. For this, nine additional points on the linear interpolation between each pair of pivots were evaluated, as well as the pivots themselves. These additional points yield information about where further optimization is necessary, which couldn't be inferred from the loss on the pivots alone. The result of this analysis can also be used to plot the loss along the final path after the optimization is completed, as well as after each optimization phase.

# 3 Methods

This chapter explains relevant details about the implementation of the algorithms mentioned in Background (Chapter 2), the workflow for creating the results, as well as more information about the used datasets.

## 3.1 Objective

UMAP and t-SNE find local minima on the loss surface defined by their loss function for a certain dataset. When run again on the same dataset, the optimization might end up in a different local minimum, especially when using random initialization. The optimization takes place in a $2N$-dimensional space, as each data point has 2 degrees of freedom: motion along the x- and y-axes. Each full embedding can be represented as a single point in the $2N$-dimensional space. During optimization, the (2D) position of each point is adjusted, corresponding to moving the point representing the whole embedding in its $2N$-dimensional space until reaching the final configuration, again representable as one $2N$-dimensional point.

Similar to the movement during optimization, different final embeddings can be transformed into each other by moving the $2N$-dimensional point starting from the position corresponding to the first embedding to the position corresponding to the second embedding. Any path from the first to the second point can be imagined, though most paths will result in heavily distorted or seemingly randomly scattered images if visualized along the way. Using the AutoNEB algorithm, such an initial path shall be optimized in order to end up with a path where the corresponding visualization is of reasonable quality along the whole path.

## 3.2 Datasets

Fashion-MNIST (Xiao et al., 2017) consists of Zalando's article images, black-and-white, with a resolution of $28 \times 28$ pixels each. It is split into a train set of 60000 and a test set of 10000 images. Each image is assigned a label from one of 10 classes. The class labels are shown on the colorbar of e.g., Fig. 4.4. Fashion-MNIST was designed as a drop-in replacement for the

far simpler MNIST dataset which consists of images of handwritten digits in the same format. Fashion-MNIST download [1].

The C. Elegans dataset contains RNA data from 86024 individual cells of Caenorhabditis Elegans. For each of the cells, all RNA strands present at a specific time were extracted, amplified, and the number of occurrences of each strand counted by (Packer et al., 2019). As further preprocessing, Principal Component Analysis (PCA) was applied by (Narayan et al., 2021) and only the first 100 principal components included. Each data point, representing one cell, is thus 100-dimensional. The points were assigned a label from one of 37 classes, one of which ("nan") contains cells that couldn't be classified. The class labels are shown on the colorbar of e.g., Fig. 4.5. C. Elegans download[2].

Both UMAP and t-SNE produce clear visualizations on Fashion-MNIST, while the C. Elegans embeddings are more intricate and allow to spot more special cases. The C. Elegans dataset is furthermore an example of data which produce considerably different embeddings when visualized by UMAP using its default initialization, and it is thus of particular interest to study connections between them. Choosing two datasets with such different properties helps distinguish artifacts from actual findings and is meant to show the general applicability of the results.

## 3.3 UMAP

### 3.3.1 Embedding Generation

In the first step, two embeddings that show some globally different configurations are produced. This is done using the default UMAP implementation of the Python package "umap-learn" (McInnes, 2018) with shape parameters set to $a = 1, b = 1$.

For the Fashion-MNIST dataset, random initialization is used, as the default initialization results in embeddings with only minor differences. For the C. Elegans dataset, the default initialization already results in globally different embeddings. It is therefore preferred over random initialization to get some results closer to a real-world use case, where default initialization is expected to be more common.

### 3.3.2 Loss Function Implementation

For the optimization, AutoNEB expects a function that takes the position of the current pivot along the path as input and returns a loss value. The input is the 2D position of every point in

[1] https://github.com/zalandoresearch/fashion-mnist/tree/master/data/fashion
[2] http://cb.csail.mit.edu/cb/densvis/datasets/

the embedding, which is the same as the $2N$-dimensional coordinates of the whole embedding in parameter space. The output is the UMAP loss Eq. (2.1) for that embedding. The high-dimensional similarities $\mu_{ij}$ do not depend on the current embedding and do not need to be constructed again for every loss evaluation. Therefore, they are precomputed once before starting the optimization and are available throughout the whole process.

To implement the loss function Eq. (2.1) efficiently, PyKeOps (Charlier et al., 2021) LazyTensors are used. PyKeOps supports symbolic operations on matrices, i.e., the actual matrix entries in memory remain unchanged, and mathematical operations performed on them are saved symbolically in an accompanying string. They are only executed as soon as a "reduction operation" such as a row- or column-wise summation is performed. The result of such a reduction is a regular array. This has two major benefits for the use case at hand: Firstly, the symbolical computation results in a considerable speedup, and secondly, matrices saved as LazyTensors can be pushed to the GPU row by row. Without this, the GPU couldn't be used at all, as the whole matrix would be too large to fit on it at once.

The shape parameters $a$ and $b$ Section 2.2.3 are both set to their default value of $1$ throughout all experiments. The exponent $b$ cannot be used with this implementation because PyKeOps only supports integer exponents. $a$ could in principle be used even with PyKeOps, but had negligible influence on the results. It is fixed at $1$ for more comparability between runs.

#### Computation Speed

In the original UMAP implementation of (McInnes et al., 2018), points were only sampled when calculating the loss to increase computational efficiency. Later, (Damrich & Hamprecht, 2021) showed that this results in a different effective loss function than originally intended. The loss function used here, Eq. (2.1) is already the updated version, and no sampling is used in the calculation of the loss. This is possible due to the computational speedup of using PyKeOps LazyTensors and executing all major calculations on the GPU. Optimizing an embedding for 520 steps takes 36 seconds on Fashion-MNIST and 68 seconds on C. Elegans. The default implementation takes 31 seconds on Fashion-MNIST and 39 seconds on C. Elegans for 500 steps. While not leveraging a direct advantage in terms of computation time, the explicit use of the corrected loss function prevents the overcontraction that otherwise arises with increasing dataset size in the default implementation.

The total duration for optimizing a UMAP connection using AutoNEB lies between 8 and 17 minutes on Fashion-MNIST and between 15 and 28 minutes on C. Elegans. The large fluctuations depend on the optimization procedure, especially the number of pivots and when they are inserted. The total duration scales linearly with the total number of loss evaluations.

### 3.3.3 Path Initialization

Along the linear interpolation between the two end point embeddings, three path points are initialized with equal spacing. The end points remain fixed during the whole optimization process. This means that after the finished run, the points along the path will have undergone more optimization than the end points, as being part of the linear interpolation between optimized points can be seen as being closer to an optimal configuration. This is particularly the case for path points closer to the original embeddings, which start out with a comparably low loss and is less the case for points in the middle of the linear interpolation, where both embeddings mix in an unfavorable way.

To be able to meaningfully compare the loss along the path with the loss of the initial embeddings at its end points, this effect needs to be counteracted. Therefore, after the linear interpolation, the end points undergo the same optimization procedure as the points along the path subsequently undergo during AutoNEB.

### 3.3.4 Optimization Settings

Similar to the default UMAP, the optimization features a learning rate decay. AutoNEB expects several longer optimization phases with constant learning rates each, specified in a config file. To approximate a learning rate decay while avoiding reading the config file for every step, the learning rate is decreased every 10 optimization steps. As the path starts out with a linear interpolation between already optimized embeddings, the initial learning rate is set to 0.05 instead of 1 and decreases linearly until reaching 0 at the end of the optimization. Starting with a higher learning rate on an already optimized embedding would result in the repulsive loss term catapulting close points far apart and thus effectively restarting the whole optimization from a random configuration.

In total, 520 optimization steps are performed, similar to UMAP's default 500 steps. After 120 iterations, 4 additional pivots are inserted, as peaks in the interval adjacent to the end points did arise around that time and would continue to grow if no pivot directly optimizing them was added. Gradient descent with a momentum of 0.9 and gradient clipping similar to the default UMAP implementation are used. In each optimization step, the gradients are clipped at a value of $4.0$ before applying them.

### 3.3.5 Attractive / Repulsive Loss

The UMAP loss function Eq. (2.1) consists of an attractive and a repulsive term. In the implementation of the loss function, both terms are accessible individually and are added up only in the last step before the loss is returned. The attractive loss comprises an entry for each pair of

points. To get a total contribution for each single point, the entries of all pairs containing that point are summed up. This point-wise attractive loss is directly accessible in an intermediate step of the implementation. The repulsive loss comprises an entry for each point, containing the value of this point's total repulsion from all other points. This point-wise repulsive loss is directly accessible before the summation to get the total repulsive loss. This means that both the individual attractive and the individual repulsive contributions can be output for every loss evaluation with practically no additional computational cost.

## 3.4 t-SNE

### 3.4.1 Embedding Generation

Using OpenTSNE's implementation (Poličar et al., 2024), 150 t-SNE embeddings of Fashion-MNIST and 12 embeddings of C. Elegans are generated. On both datasets, random initialization is used to get more varied embeddings. The higher complexity of the C. Elegans dataset results in sufficient differences between most randomly generated embeddings, so fewer examples are needed. Apart from the random initialization, the standard t-SNE configuration is used, including early exaggeration with an exaggeration factor of 12 for 250 iterations, followed by 500 iterations without exaggeration.

For each of the datasets, two embeddings with equally low loss but major differences in the global layout are chosen to be connected. Both were among the embeddings with the lowest loss and feature a subjectively good visual quality, e.g., do not contain ripped-apart clusters.

### 3.4.2 Loss Function Implementation

The implementation of t-SNE's loss function is done similar to the implementation of UMAP's loss function; PyKeOps LazyTensors are used to allow for symbolical computations and the possibility of performing them on the GPU (please refer to Section 3.3.2 for more details). Losses computed with this implementation have a constant offset of $0.2795 \pm 0.0016$ compared to the Kullback-Leibler divergence (KL divergence) as output by openTSNE's implementation (Poličar et al., 2024). However, the differences between the losses of different embeddings are in accordance with the differences in KL divergences. The loss value is mainly necessary to compare embeddings, while its absolute value does not matter as much. In particular, a constant offset does not influence the optimization procedure, which only takes gradients into account. Due to t-SNE's different approach, not explicitly splitting the loss into an attractive and a repulsive part, retrieving each individual point's contribution to the attraction and the repulsion is not straightforward compared with UMAP and is not undertaken in this work.

#### Computation Speed

Optimizing an embedding for 750 steps takes 65s on Fashion-MNIST and 110s on C. Elegans. The default implementation takes 96s on Fashion-MNIST and 112s on C. Elegans for 750 steps. The total duration for optimizing a t-SNE connection using AutoNEB is around 12 minutes on Fashion-MNIST and around 37 minutes on C. Elegans. The total duration scales linearly with the number of loss evaluations during the optimization. The number of pivots and when they are inserted have a much greater influence than the total number of optimization iterations. The optimization phases (Section 3.4.4) are thus chosen in a way that delays adding most pivots as late as possible.

### 3.4.3 Path Initialization

The initial path is created by linearly interpolating between the end points, i.e., the embeddings to be connected. The number of initial pivots can be chosen at will, but using only one pivot proved sufficient, as more pivots will be added during the optimization. This has the advantage that one pivot can be optimized for more steps during the same computation time as compared to e.g., three initial pivots. The pivots added later will be added to the already partly optimized path, which means they can benefit from the one initial pivot's optimization as well.

Alternatively, the path can be initialized manually. This way, a particular movement of the clusters will be suggested, though it can slightly change during the optimization. In principle, any path can be chosen as initialization, but each cluster should follow a somewhat coherent path from one endpoint to the other. Otherwise, the optimization will not find a low loss connection, as the global movement of the clusters remains fairly close to the initial path.

Following either case of path initialization, the end points should be optimized prior to starting the AutoNEB optimization. The path points will be optimized further, while the end points remain fixed. Without this prior optimization, the end points retain their initial loss, considerably higher than the loss of the points along the path. This would prevent a meaningful assessment of the connection's quality. It would not be clear to what extent end points with higher loss than the connection could be attributed to the actual properties of the connection and how much of the difference would be an artifact.

### 3.4.4 Optimization Settings

A constant learning rate of $10^4$ (10.000) is used throughout the whole optimization process, including during the prior optimization of the end points. Learning rate decay is not applied. Gradient descent with a momentum of 0.9 and without gradient clipping is used. During

the optimization, more pivots are added after each phase. They are added only where the loss between existing pivots exceeds a threshold (the second insertion method described in Section 2.4.6). The subsequent number of optimized pivots is thus the maximum value, as it can happen that the loss exceeds the threshold in fewer intervals than the maximum number of points to be inserted.

The following AutoNEB optimization phases are used for t-SNE:

- 500 iterations with 1 pivot
- 800 iterations with 3 pivots
- 200 iterations with 7 pivots
- 100 iterations with 15 pivots
- 100 iterations with 31 pivots

This amounts to a total of 1700 optimization steps for the initial pivot, which at the same time determines the number of optimizations the end points need to undergo before starting the AutoNEB procedure. On the C. Elegans dataset, the path is initialized with two pivots instead of one, which results in one additional pivot for every optimization phase. In the last phase, however, the loss was already below the threshold in several intervals. The last optimization phase for C. Elegans thus only incorporates 27 pivots.

## 3.5 AutoNEB

The implementation of the AutoNEB algorithm (Draxler et al., 2018) that is used here was designed with the application on Neural Networks in mind. It requires modifications in several instances to add the functionality to connect minima on dimension reduction loss surfaces.

AutoNEB expects a model that can be passed data and return a result, e.g., a classification of the input. This result is passed to a loss function, which returns the corresponding loss, e.g., comparing the classification with the actual class labels. To fit into that scheme, a "model" is defined, which contains the current embedding as a model parameter (usually used to store model weights). During the forward pass, this embedding is simply returned. This "model output" can then directly be passed into the respective loss function for either UMAP or t-SNE (for details on the loss functions, see Section 3.3.2 and Section 3.4.2 respectively). The loss function then returns the loss for the embedding.

Using pytorch's Autograd engine, a backwards pass through the loss function is made to determine in which direction to move the coordinates of each embedded point in order to decrease the loss value. Autograd works by storing the gradient of every operation done for the calculation of the loss during the forward pass. In the backwards pass, Autograd traverses

through the operations backwards, starting at the loss value. In each step of the backwards pass, it is calculated in which direction the inputs to the current operation would need to change in order to shift the result of the operation in the desired direction (for the first step, of which the output is the loss, this would be to decrease that loss value). This is then repeated with the operation one step further from the output. The backwards pass is finished when the initial input, which is the embedding, is reached and the positions of all the embedded points can be updated.

In several instances, functions of the original AutoNEB code are modified in order to comply with the new requirements of the different use case:

- Optional gradient clipping is integrated after the Autograd backwards pass.
- Several functions involved in the forward pass are modified to not pass data to the "model" as it only returns the current embedding, which does not need any input.
- AutoNEB normally does not allow custom initialized paths, which are necessary in order to optimize the end points separately from the initial path. AutoNEB already has the functionality to pause the optimization of one connection, further optimize another connection and continue the optimization after that. This functionality is made use of by pretending the optimization did already start and was only paused. The initial path can then be introduced as being the latest path where the optimization was interrupted and AutoNEB will "continue" the optimization from there, while in reality just starting it.

# 4 Results

The main results of this work consist of dynamic visualizations of the embeddings during the connection. These results are accessible under **View animated figures**in the form of .gif files. Figs. 4.4, 4.5, 4.7, 4.8 and 4.10 comprise selected frames from these animations.

## 4.1 Loss Coloring

The UMAP loss Eq. (2.1) computes a sum over loss contributions from each individual point. In the following, embeddings are colored in a way to show each point's contribution to the loss. This is done separately for the attractive and the repulsive loss terms.

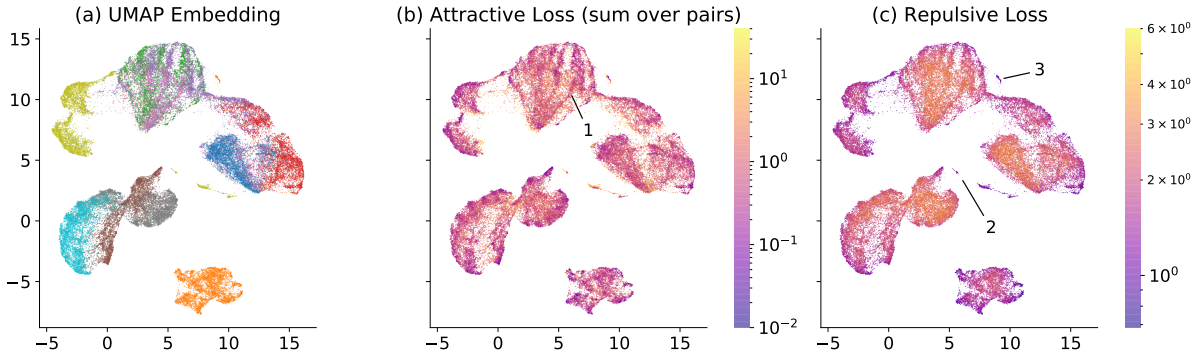



Figure 4.1: **(a)** UMAP embedding of the Fashion-MNIST dataset colored by class. Colors and labels as in Fig. 4.4. **(b)** Same embedding, each point colored logarithmically according to its contribution to the attractive loss. For each point, the sum over the loss of all pairs containing that point is shown. **(c)** Same embedding, each point colored logarithmically according to its contribution to the repulsive loss.

### 4.1.1 Attractive Loss

In Figs. 4.1(b) and 4.2 (b), the color of each point indicates how much this point contributes to UMAP's attractive loss term (see Eq. (2.1)). As the attractive loss is computed for pairs of points, it was summed over all pairs that include the point in question.In comparison to the repulsive loss, the general structure of the attractive loss is more varied on a small scale, with considerable structure even inside clusters. This can be attributed to its focus on the nearest

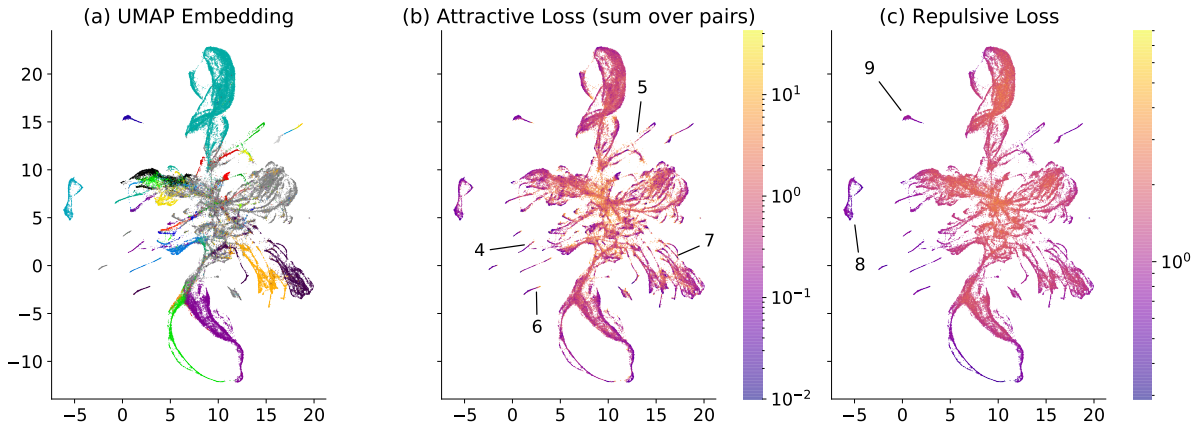



Figure 4.2: **(a)** UMAP embedding of the C. Elegans single cell dataset colored by class. Colors and labels as in Fig. 4.5. **(b)** Same embedding, each point colored logarithmically according to its contribution to the attractive loss. For each point, the sum over the loss of all pairs containing that point is shown. **(c)** Same embedding, each point colored logarithmically according to its contribution to the repulsive loss.

neighbor relationship between points, so the attractive loss of a point is mostly influenced by points in its near vicinity.

The attractive loss seems to prefer denser configurations. In the areas on the outer edges of clusters, the points are packed more tightly. These are also the regions with the lowest attractive loss. Even towards the middle of clusters, streaks of higher density can be seen, again with considerably lower attractive loss. An example of such a structure is annotated with "1" in Fig. 4.1 (b).Regions with sparser points but a high attractive loss mostly have a high repulsive loss as well. This points to a conflict between the attractive and repulsive loss forces in these regions, preventing further contraction.

Several clusters show a dramatically increasing attractive loss towards one edge. This happens especially frequently for very slim, almost linear clusters, of which nearly all show an increased loss on one end. Examples on C. Elegans, Fig. 4.2 (b), are annotated “4”, “5”, and “6”. The high loss indicates clusters that were ripped apart during the optimization. One cluster that was separated into two distant parts is annotated “6” and “7”, respectively (see Fig. 4.2 (b)). In such cases, the points near the break lack their nearest neighbors, which got disconnected, often because of another cluster lying in between. Clusters that were ripped apart do not correlate with the actual structure of the high-dimensional data, so it is advantageous to be able to spot them easily (see Section 4.1.3).

### 4.1.2 Repulsive Loss

The individual contribution of each point to UMAP’s repulsive loss term (see Eq. (2.1)) can be seen in Figs. 4.1 (c) and 4.2 (c). In comparison to the attractive loss, the repulsive loss does

not consider if points are adjacent in the nearest neighbor graph. The loss does only depend on the actual distance in the embedding, though not linearly.The repulsive loss coloring thus mainly indicates the point density throughout the embedding. The different clusters show great variation in their respective contributions. Remote and clearly separated clusters such as “8” and “9” in Fig. 4.2 (c) do only minimally contribute. Inside these clusters, their periphery has even lower loss than the more central parts. Especially the overcontracted, nearly linear structures seem favorable for the repulsive loss.

Towards the middle of the embedding in Fig. 4.2 (c), the repulsive loss rises considerably in a smooth transition, in contrast to the steep gradients of the attractive loss. The increase stems from the high density of points in this region, not from the central positioning itself. This can be seen in Fig. 4.1 (c), where the small distinct clusters near the center of the embedding “2” do not exhibit a high repulsive loss contribution at all.

The repulsive loss between two points decays quickly with increasing distance, so points beyond a certain radius can essentially be disregarded as they only have a minimal influence. This is in accordance with the observation of distinct, small clusters having a very low loss, no matter their overall position in the embedding. Linear configurations do also benefit from this, as they contain only a few close points in the direction of the linear formation, while the other directions remain empty.Similarly, small isolated specks, e.g., “2” and “3” in Fig. 4.1 show low repulsive loss, as only the few included points can contribute to the loss, despite being relatively densely packed.

Especially in the central regions of the C. Elegans embedding (Fig. 4.2), it seems difficult to balance the repulsive loss with the attractive loss. There, points may be connected in multiple intertwined ways, for which a faithful embedding in two dimensions is not necessarily possible. This results in a compromise between attractive forces between the various neighbors and the repulsive loss pushing them apart. In both the attractive and the repulsive coloring, these regions show a high loss compared to, e.g., linear configurations, cf. “4”, “5”, and “6” in Fig. 4.2 (b).

### 4.1.3 Benefits

The color of each point indicates how well the current embedding represents the high-dimensional distance to its neighboring points. Thus, a high loss contribution, corresponding to an unfaithful representation of the local connectivity, is highlighted. Sometimes, clusters get split up during the optimization, although they would ideally remain connected. On the other hand, clusters can actually consist of several barely connected subclusters. These cases would normally be difficult to distinguish just by looking at one final embedding. With the pointwise attractive loss coloring, clusters that are unfortunately ripped apart can be immediately

spotted. They will be highlighted, as the points near the split will not be close to their true neighbors, which results in a high loss.

Using this, it can be better evaluated which parts of a UMAP embedding portray the underlying structure of the data rather faithfully and which patterns are artifacts. If necessary, UMAP may then be re-run, e.g., with another initialization, to obtain a better visualization of the clusters in question.

Even UMAP with default initialization produces qualitatively different embeddings on some datasets (e.g., C. Elegans). Using the point-wise attractive loss coloring in addition to the usual colors according to class labels can help in comparing different embeddings regarding their truthfulness in depicting a cluster's structure.

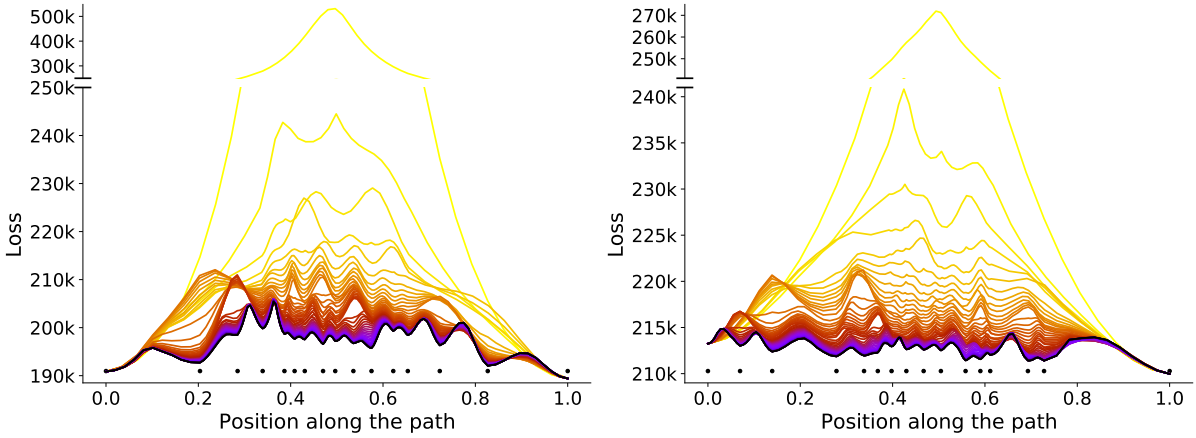
Calculating the individual loss contributions requires only the negligible computational cost of one optimization step, compared to 500 such steps for creating the embedding. It is thus a realistic option to integrate the loss coloring into the existing visualization workflow, i.e., plotting a coloring of the points' loss contribution along with every UMAP embedding that is created. This would reduce the common problem of estimating the quality of an embedding by giving a transparent and intuitive indicator for its faithfulness, even differentiating between individual regions.

Instead of the common practice of running UMAP several times and just choosing the embedding with the lowest loss, the loss coloring allows for choosing the final embedding based on its faithfulness in a specific region of interest, and it might even prevent the need to run UMAP again altogether.

## 4.2 UMAP

As can be seen in Fig. 4.3, the AutoNEB algorithm succeeded in finding a low-loss connection between minima for both Fashion-MNIST and C. Elegans, especially when compared to the initial linear interpolation.

### 4.2.1 Loss Curve




(a) Fashion-MNIST

(b) C. Elegans

Figure 4.3: **(a)** Loss curve along the path connecting two randomly initialized UMAP minima on Fashion-MNIST. **(b)** Loss curve along the path connecting two UMAP minima with default initialization on C. Elegans.
The final optimization result is shown in black; brighter colors show the course during the optimization. The x-axis shows the 2-norm between points along the path (normalized to 1). The y-axis is broken to scale up the final curve. The loss is plotted in the same orientation from left to right as the embeddings in Fig. 4.4 and Fig. 4.5. The black dots indicate pivots of the path that were directly optimized.

#### Fashion-MNIST

On Fashion-MNIST (Fig. 4.3a), the highest peak of the connection has a loss of 205.4k, compared to losses at the endpoints of 190.8k and 189.4k, respectively. The majority of the connection has a loss of around 200k, thus lying around 10k higher than the end points. When creating randomly initialized UMAP embeddings on Fashion-MNIST using the default umap-learn implementation (McInnes, 2018), the loss between the embeddings varied by 10k-15k, with a maximum difference between the best and worst of 20k. When running UMAP only once on the dataset, each of these embeddings would be considered completely optimized and a valid visualization result. It can thus be concluded that the path found by AutoNEB lies within the usual variance of different minima found by the default UMAP.

#### C. Elegans

The connection on C. Elegans Fig. 4.3b has a slight asymmetry with higher losses on the left side, which can be attributed to the already different loss of the connected embeddings. The left and right end points have a loss of 213k and 210k, respectively. Accordingly, the highest peak, with a loss of 214.8k, lies directly adjacent to the left end point. This amounts to a difference of only 1.6k to the higher and 4.9k to the lower of the two initial embeddings. This

is especially remarkable, as the usual variation between the different minima on C. Elegans, created using default initialization, lies between 5k and 7k. The loss along the path constantly stays within these bounds; thus, a low loss connection was found.

### Embedding Initialization

The UMAP embeddings on the C. Elegans dataset were created using default initialization, which is using Laplacian Eigenmaps, in contrast to all other embeddings used throughout this work, which were initialized randomly. On C. Elegans, the default initialization did already result in minima with different global structures, while random initialization produced embeddings often containing qualitative flaws such as heavily twisted clusters. Default initialization was also chosen to provide an example closer to real-world use cases, showing that even with default initialization, different minima are generated on some datasets.

### Optimization Schedule

Different learning rates and optimization schedules were tried, yet the outcome has remained fairly stable. Even a constant learning rate of 0.03 works well and results in a comparably low loss path. If the learning rate is too high, e.g., 1, the repulsive loss catapults close points far apart, resulting in a nearly random configuration of points. During the subsequent learning rate decay, that embedding is optimized again, essentially amounting to a UMAP run with random initialization for each point individually.

### Optimal Pivot Insertion

Major improvements of the final loss curve do primarily depend on the number of pivots inserted and the time after which they are inserted. Typically, the loss between one end point and its adjacent pivot starts to rise after about 120 optimization steps. To prevent a peak from forming, a new pivot should be inserted between the two points. As new points are only inserted if the loss rises above a threshold and the highest peaks are prioritized, it is necessary to delay the insertion until a considerable peak has begun to form. The new pivot will decrease that peak during the remaining optimization. A similar behavior can also occur at other points along the path. It is advisable to look at the loss plot, including its course during the optimization, to determine a suitable timing for the insertion of additional points.

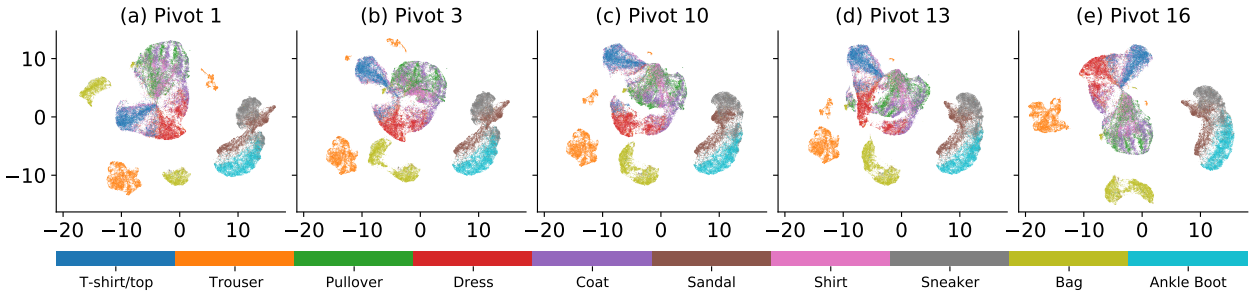



Figure 4.4: Points along the path connecting the UMAP embeddings (a) and (e) on Fashion-MNIST. Pivot numbers are in accordance with the black dots under the corresponding loss curve Fig. 4.3a. **View animated figures**.

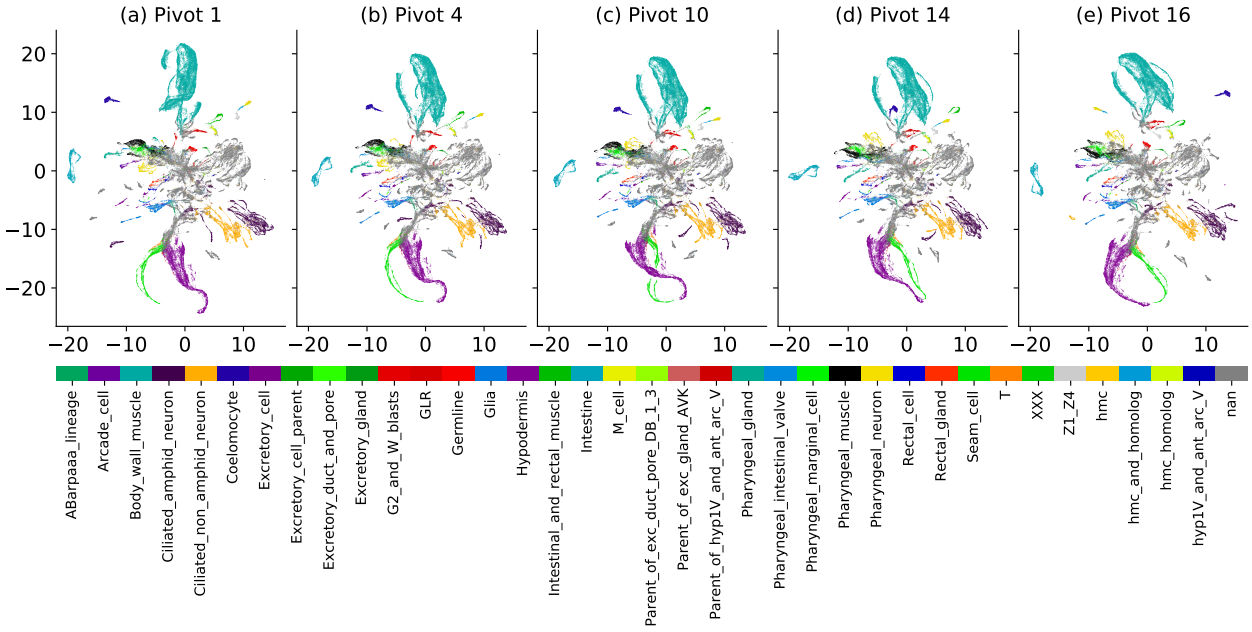



Figure 4.5: Points along the path connecting the UMAP embeddings (a) and (e) on the C. Elegans dataset. Pivot numbers are in accordance with the black dots under the corresponding loss curve Fig. 4.3b. **View animated figures**.

### 4.2.2 Visualization

#### Movement Segmentation In Time

Different clusters seem to move in distinct time intervals, during which other clusters remain unchanged. Only a small subset of all clusters moves at once, although most clusters undergo significant restructuring or repositioning at some point during the transformation from the one end point to the other. In contrast, when interpolating linearly between the end points, the movement of each cluster would be completely homogeneous in time, and all the clusters would be changing simultaneously.

During the AutoNEB optimization, the uniform behavior of the initial linearly interpolated path is gradually changed until there is a clear distinction between clusters currently in move-

ment and others remaining passive. These passive clusters only shift slowly or experience minor distortions such as limited stretching or warping.

#### Active Phases

Necessary changes in order to transform one end point embedding into the other include shifting, rotating, stretching, warping, and inversion. Each of these actions is carried out either by several clusters in unison, one cluster alone, or only a subgroup of one cluster. Subgroups can also include points of different classes if they lie adjacent, are interlinked, or overlap. For these major changes, the affected areas enter a designated active phase, where the specific change is carried out from start to finish. Notably, one change can be split up into several distinct actions; e.g., a cluster doesn't have to move from its initial position to its final position in one continuous active phase. Instead, the total shift can be partitioned into several individual actions, each shifting the cluster from one position to the next, together adding up to the total relocation necessary. After each active phase, the cluster will be in a somewhat favorable, or at least tolerable, position with respect to the loss. It can passively remain in such a position during several active phases of other clusters until entering its next active phase. Active phases of different kinds can alternate; e.g., a shift can be followed by an inversion, a pause, and finally a second shift.

#### Segmentation Prevents Loss Peaks

In the $2N$-dimensional parameter space, the distinct active phases correspond to movement in only a few directions at a time. If only some distinct clusters move simultaneously, the $2N$-dimensional point representing the whole embedding moves only along the coordinate axes associated with the points in these active clusters. When other clusters become active, the $2N$-dimensional point changes direction and starts moving along other axes associated with the new, now active clusters. This way, large parts of the embedding remain in the same favorable position of low loss, while only a few points at a time deviate from that minimal loss configuration. The active points move until they reach another favorable position, e.g., they start on one side of another cluster, move through it in one active phase, and end their active phase after reaching its other side. The overall loss does not suffer as much if only a subset of the total embedded points are located at unfavorable positions during the movement.

#### Sparse Layout Allows For Low Loss Paths

The characteristic UMAP layout with well-separated compact clusters allows for comparatively much freedom in choosing a low-loss path. Distinct clusters have attraction mainly between points inside the cluster and a sufficient distance from the rest of the embedding to

satisfy the repulsive loss conditions. They can move rather independently from their initial position to their destination, as long as their internal structure does not change too much. This way, many low-loss paths exist that all maintain sufficient distance from other clusters. In every optimization step, each point moves according to its own gradient only, unaware of the possible movement of the surrounding pivots. If only a single point would move while the other points of the surrounding cluster would remain in place, the loss would increase due to that point leaving its proper position. For a whole cluster to move, it is thus necessary that most of its points experience similar forces, pushing them in the same direction in unison. This behavior is facilitated by UMAP's compact clusters, as points on the nearer and farther sides of a single cluster can still both be close enough to points of adjacent clusters to be repelled from them.

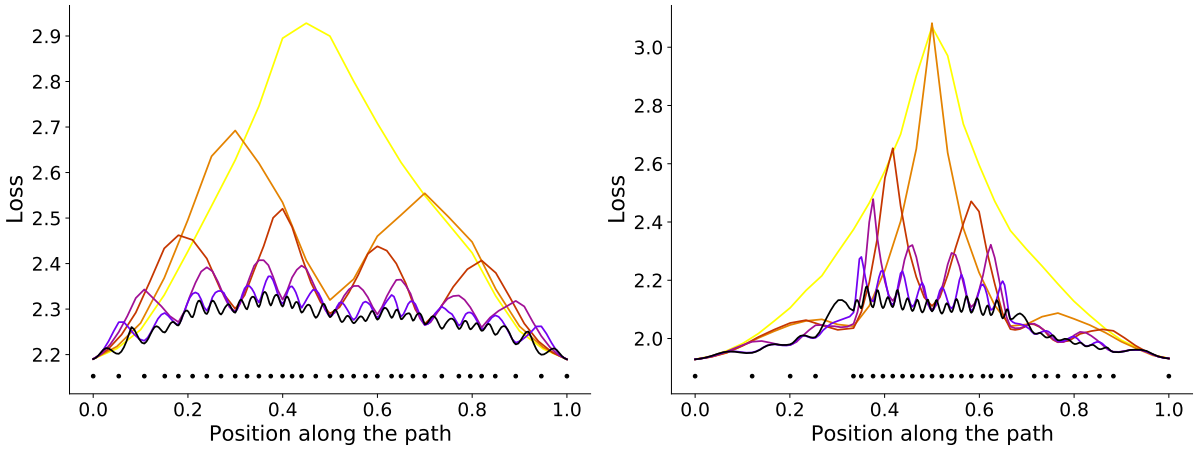
#### True Loss Function Characteristics

The loss-based version of UMAP used here seems to favor spreading out clusters more as well as increasing the amount of whitespace between them, compared to the default umap-learn implementation (McInnes, 2018). This is an effect of the corrected loss function, Eq. (2.1), which appropriately increases the repulsion, counteracting the over-contraction that usually occurs for UMAP, especially with increasing dataset size (Damrich & Hamprecht, 2021). This less compact layout helps in identifying structures inside individual clusters, which would otherwise be too dense. The clusters are still well separated, and the global configuration remains largely the same.

## 4.3 t-SNE

### 4.3.1 Loss Curve

The connection of t-SNE loss minima did not yield paths nearly as flat as was the case for UMAP. Instead, the loss rises continually towards the middle of the connection (see Figure Fig. 4.6). Generally, the loss increased, the further the embedding deviated from the original minimum (see Fig. 4.9).




(a) Fashion-MNIST

(b) C. Elegans

Figure 4.6: **(a)** Loss curve along the path connecting two t-SNE minima on Fashion-MNIST. **(b)** Loss curve along the path connecting two t-SNE minima on C. Elegans. On both datasets, the embeddings are initialized randomly. The final optimization result is shown in black; brighter colors show the course during the optimization. The x-axis shows the 2-norm between points along the path (normalized to 1). The loss is plotted in the same orientation from left to right as the embeddings in Fig. 4.7 and Fig. 4.8. The black dots indicate pivots of the path that are directly optimized.

### Fashion-MNIST

On Fashion-MNIST, the loss along the path (see Figure Fig. 4.6a) rises considerably with increasing distance from the end points. It has a maximum value of 2.338, rising around 0.15 higher than the end points at 2.19 . The usual differences between the losses of randomly initialized t-SNE embeddings are around 0.01-0.02 with a maximum difference of 0.04 . The standard deviation of the losses lies at 0.008 for the 150 embeddings generated. This means that the loss along the connection does not lie inside these fluctuations but lies instead significantly higher throughout most of the path. The dent exactly in the middle can be attributed to the optimization procedure, which started out by optimizing only one central pivot before adding further pivots in between. It is thus neither a property inherent to the dataset nor part of t-SNE's special dynamics. The initial large peak (bright yellow in Fig. 4.6a) corresponds to the naive linear interpolation used as initialization.

### C. Elegans

The loss on the C. Elegans connection does rise with increasing distance from the end points, as was already the case for Fashion-MNIST. The highest peak, on the left edge of the plateau, has a loss of 2.18; the loss of the end points is 1.93, amounting to a difference of 0.25. The usual loss variation of the 12 generated embeddings is much smaller, with a value of 0.015;

the connection does by far exceed this range. One major difference between the datasets seems to be the plateau in the central region, with a nearly constant loss of 2.12, apart from the peaks between pivots. This is, however, not a feature inherent to the dataset, but can be attributed to the optimization procedure.

Instead of one central initial pivot as for Fashion-MNIST, two initial pivots at 1/3 and 2/3 of the path, respectively, are used with C. Elegans, with otherwise the same optimization schedule and settings. This is necessary in order to optimize a peak lying at 2/3 of the initial path. It stemmed from the large turquoise cluster labeled "Body_wall_muscle", see Fig. 4.8, not being optimized well and performing a naive inversion completely in between two pivots. With one central pivot, no plateau emerged, and the general loss curve exhibited a shape similar to that of Fashion-MNIST (Fig. 4.6a). However, the peak at around 1/3 from the right end point still retained a loss of 2.279, much higher than the surrounding path. During most of the optimization, no pivot lay directly on the peak, so its high loss remained largely unchanged. In order to place a pivot on the peak right from the start, the path is initialized with two pivots. An alternative would be to insert a single new pivot during the second-to-last optimization phase, which would then explicitly optimize that point. With the two initial pivots, the "Body_wall_muscle" cluster does not invert naively but undergoes a more sophisticated restructuring, splitting into its subclusters.

The initial linear interpolation always has a large peak in the middle of the path. To optimize this loss, it is favorable to place a pivot on top of the peak already in the first step. It is thus advisable to choose an uneven number of initial pivots so that one initial pivot lies exactly at the center of the path.

#### Initial Peak

The same cluster can be located on opposite sides of different embeddings and is often inverted in that case. For Fashion-MNIST, this applies to the "Trouser" cluster (orange); see Fig. 4.7. During the linear interpolation before the optimization, such a cluster becomes slimmer towards the middle between both initial embeddings. There, the actual inversion happens, and it takes a nearly linear shape with most of its points superposing each other. Towards the other end point, the cluster broadens again, until it approximately takes its original shape again, but inverted. The heavily superposed conformation towards the middle results in the high initial loss peak before the optimization. During the optimization, the slim cluster gets spread out again to prevent the high loss associated with the overly dense layout.

#### Intermediate Points

Between the points that were optimized directly, small peaks appear in the loss plot (Fig. 4.6). In order to smoothly plot the loss curve, the loss of points along a linear interpolation between pivots is evaluated, in addition to the pivots of the chain themselves. These intermediate points regularly have a significantly higher loss than the adjacent optimized pivots.

In the visualization of the connection, these small peaks correspond to subclusters moving through each other instead of making way. This increases the loss, as they overlap and keep a too small distance between points that should optimally be further apart. This is not corrected by the AutoNEB algorithm, as the intermediate points are overlooked by the optimization.

#### Approximating The Optimal Loss Curve

To decrease the aforementioned peaks, new pivots are inserted on top of each of them along the chain. These pivots quickly get optimized to a similar loss value as the adjacent old pivots. Although all existing pivots continue to be part of the optimization, most pivots do not noticeably decrease any further in loss after the first optimization phase following their creation. One clear example exhibiting this behavior is the pivot at 3/4 of the path on Fashion-MNIST, Fig. 4.6a. It was introduced after the first optimization phase (see Section 3.4.4) to decrease the right of the two peaks and reached its final loss already after the first phase. Iteratively adding pivots on the peaks repeats this pattern, with pivots ending up at nearly the same height after their first optimization phase. This suggests that a lower bound for the loss exists along the path, below which it will not significantly drop but which can be approximated smoothly by adding further pivots.

The main mechanism for decreasing the loss is splitting up clusters into smaller sections until these parts are small enough to move through each other without increasing the loss too much. Optimizing peaks between pivots thus corresponds to splitting up even the remaining subclusters or moving them past instead of through each other. At some point, when all clusters are sufficiently split up, the loss will not decrease any further, no matter how many optimization steps ensue. The loss is then mainly influenced by the suboptimal global layout along the path and thus never reaches a value as low as the original embeddings at the end points. Section 4.3.3 covers in more detail why the global layout is not optimized by AutoNEB.

### 4.3.2 Visualization

#### Cluster Inversion

Inversions are accomplished without a large loss penalty by splitting the cluster up into its subclusters, as can be seen at the orange “Trouser” as well as the blue “T-shirt/top” cluster in

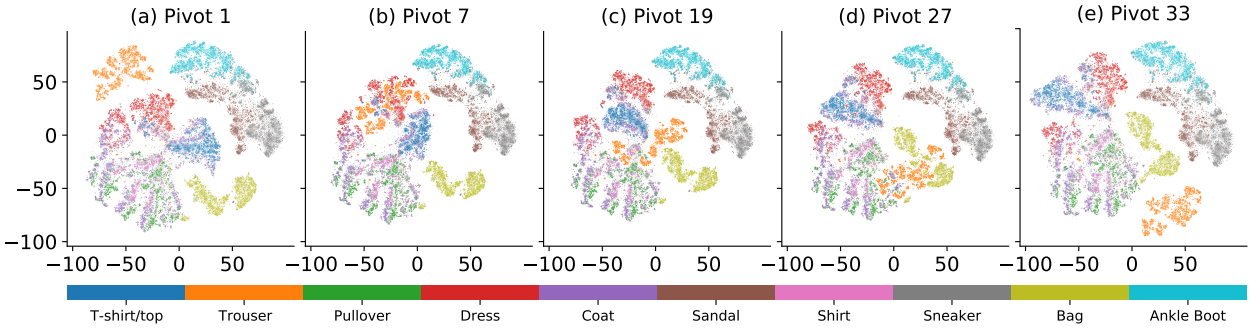



Figure 4.7: Points along the path connecting the t-SNE embeddings (a) and (e) on Fashion-MNIST. Pivot numbers are in accordance with the black dots under the corresponding loss curve Fig. 4.6a. **View animated figures**.

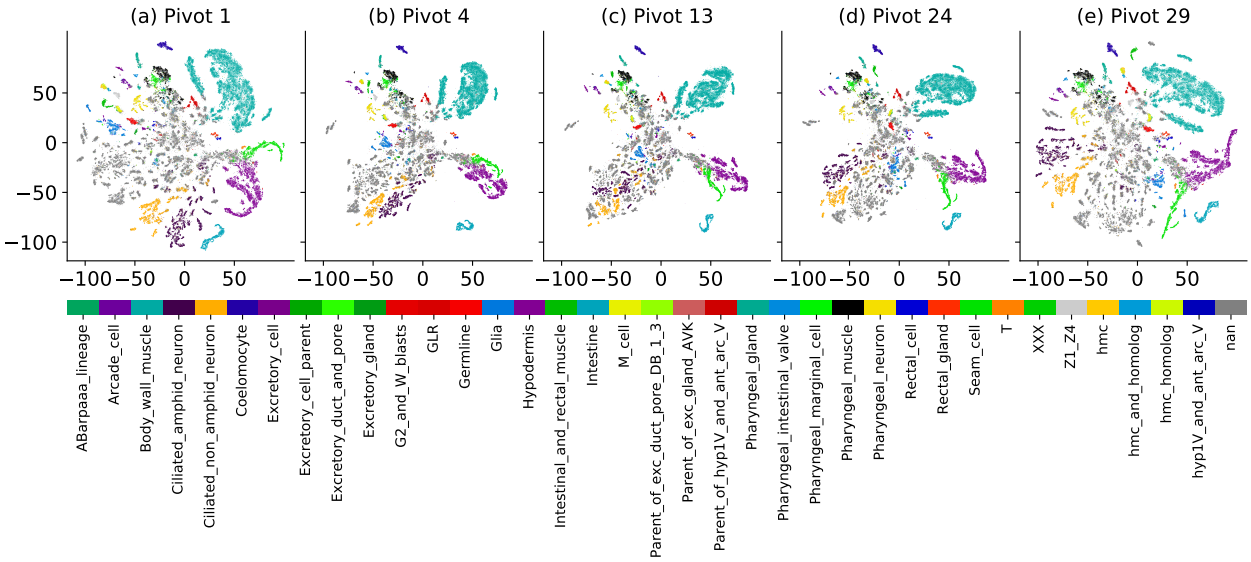



Figure 4.8: Points along the path connecting the t-SNE embeddings (a) and (e) on the C. Elegans dataset. Pivot numbers are in accordance with the black dots under the corresponding loss curve Fig. 4.6b. **View animated figures**.

Fig. 4.7. The subclusters can rearrange themselves by moving around the rest of the cluster or by moving through the cluster, in between the other subclusters. Groups of subclusters tend to remain close, which is achieved by more central clusters moving through the middle, while subclusters near the edge tend to move around the rest of the cluster. This way, the cluster is "turned inside out", resulting in an inversion.

### Superposition

During the initial linear interpolation, several clusters happen to end up superimposed on top of a cluster belonging to a different class. This is increasingly the case towards the middle of the path as the clusters move further from the well-separated positions they had at the start and end points. This effect is another reason for the high peak in the middle of the path before

optimization, i.e., when interpolating linearly, in addition to the aforementioned overly slim and compressed shape of single clusters.

If clusters of different classes overlap, the loss heavily increases, as the corresponding high-dimensional data points likely differ considerably. During the optimization, overlapping clusters split into their subclusters. Each subcluster then needs to shift only a small distance to evade an overlapping subcluster of the other class. This way, the two total clusters may be mixed in the same space, but their subclusters will not overlap anymore. An example where this behavior can be seen are the "Trouser" (orange) and "Dress" (red) clusters when they traverse through each other Fig. 4.7.

### 4.3.3 Preconfigured Path

Starting with the same embedding as in Fig. 4.7 (a), all points belonging to one specific cluster ("Trouser" label, colored orange in Fig. 4.10) are moved manually, gradually deviating further from their original position. This way, an arbitrary path can be constructed and used as an initialization, subsequently getting optimized. The chosen path is a half circle, rotating the orange "Trouser" cluster around the rest of the embedding for slightly more than 180°. This choice should allow for a relatively low-loss path while being clearly different from a linear interpolation. For the initialization, points belonging to other classes are not manipulated, though they may move during the optimization process. As with all previous connections, the start and end points are optimized after initialization for as many steps as the points along the path during the ensuing AutoNEB.

#### Loss Curve

The loss plot (Figure Fig. 4.9) shows an overall rise towards the right side, where the deviation from the original embedding is largest. Apart from a region near the left end point of similarly low loss, the curve exhibits a nearly perfectly linear slope rising towards the right. The right end point is the one furthest removed from the original embedding and does have the highest loss accordingly. The small peaks between the optimized points deviating from the linear slope have already been described in Section 4.3.1.

#### Initial Path Selection

Deliberately initializing the path can result in a relatively low loss, considering that the right end point already had a quite high loss. Such a path might even be a better choice for initializing a connection in order to achieve the lowest loss possible. However, when more than one cluster is different between the two embeddings to be connected, coming up with such a path might be challenging. In the video visualizing the whole path (**View animated figures**), it is

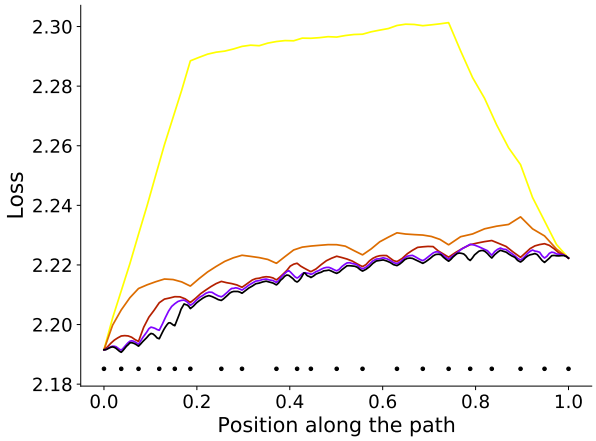



Figure 4.9: Loss curve along the preconfigured path connecting two t-SNE minima on Fashion-MNIST. The initial path was set manually, including the rightmost pivot. The final optimization result is shown in black; brighter colors show the course during the optimization. The x-axis shows the 2-norm between points along the path (normalized to 1). The y-axis is broken to scale up the final curve. The loss is plotted in the same orientation from left to right as the embeddings in Fig. 4.10. The black dots indicate pivots of the path that were directly optimized.

visible how the orange "Trouser" cluster moves closer to the rest of the embedding and only leaps further away where the distance was essentially predetermined by the initialization. For this example, the path has thus been initialized too far out, though without actually executing the optimization, the initial path will always remain an estimate. The linear interpolation used earlier (Section 4.3.1) is an easy-to-implement choice no matter the form of the embeddings to connect.

### Visualization

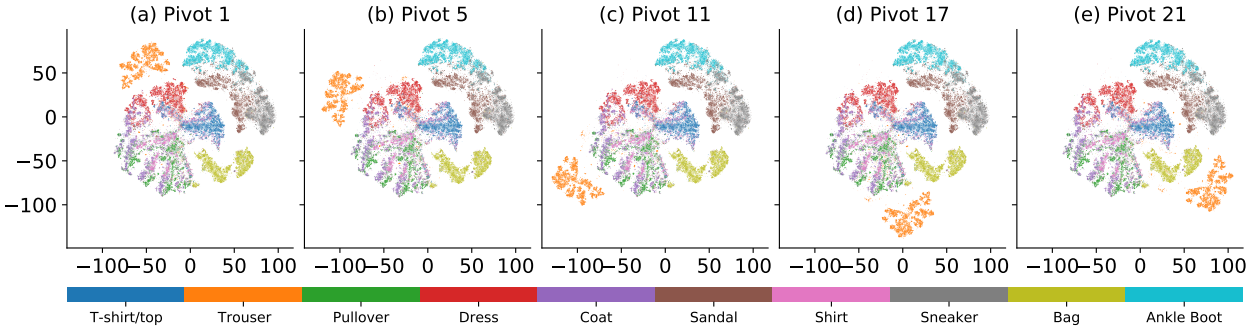



Figure 4.10: Points along the preconfigured path connecting the t-SNE embeddings (a) and (e) on Fashion-MNIST. Instead of interpolating linearly, the initial pivots along the path were set manually. Pivot numbers are in accordance with the black dots under the corresponding loss curve Fig. 4.6b. **View animated figures**.

### Optimization Only Acts Locally

As the path remains close to the initialized path on a large scale throughout the optimization, the initialization directly influences and determines the final path. It can thus be concluded that in this case, the AutoNEB algorithm does not find a global minimum by any means but rather stays close to the initial path and achieves loss reductions through small-scale optimizations. These steps do, however, fail to solve unfortunate arrangements rooted in the large-scale structure of the embeddings along the path. This is reflected in all loss curves (Fig. 4.6) retaining a higher loss along the path than at the end points throughout the optimization.

Large-scale restructurings aren't performed by the optimization, mainly due to two reasons: Firstly, changes in the global layout require a large number of points to be moved by a considerable amount. This corresponds to large distances in the parameter space to be overcome, which, given a fixed learning rate, would require more iterations than small-scale optimizations.

Secondly, shifting a whole cluster in unison is improbable, even for small shifts: When calculating the gradient, each point only "sees" its adjacent points as they are at that moment. Moving that one point while keeping all other points of its cluster fixed in place would mean leaving its optimal position inside the cluster, resulting in an increase in loss. The gradient force acting on each of the points will thus keep them fixed in place in order to not mix up the cluster's structure. Even though moving the whole cluster in a favorable direction would preserve its structure, the individual points only see their own deviation and do not take into account that the other points will move as well.

# 5 Conclusion

A new method for visualizing UMAP's embedding quality by highlighting points according to their individual contribution to the attractive and repulsive terms of the loss is proposed. It requires only negligible additional computational costs and can thus be used along with every conventional UMAP embedding to determine its faithfulness for each region individually.

Different minima of the loss surface of UMAP and t-SNE have been connected using the AutoNEB algorithm (Kolsbjerg et al., 2016). For UMAP, low-loss connections were achieved on both datasets studied, with a loss constantly within the range of usual fluctuations between different minima Section 4.2. Reasons for the existence of such connections were found to lie in UMAP's compact and well-separated cluster structure as well as a segmentation of the path into individual, consecutive actions distributed along the whole path.

On both analyzed datasets, the t-SNE loss increased, the further the respective embedding was deviating from the original loss minimum found (Fig. 4.6). This has been explicitly studied in Section 4.3.3, where a distinct slope was observed (4.9). The loss minima found by the default t-SNEproved rather stable, with no true low-loss connections found.

The AutoNEB algorithm used here to connect minima of UMAP and t-SNE is not limited to these techniques, but is rather applicable to arbitrary loss surfaces. Future research could investigate the loss landscapes of alternative visualization techniques or expand these studies into three dimensions.

# Bibliography

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, *3*(29), 861. https://doi.org/10.21105/joss.00861

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(86), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

Kobak, D., & Berens, P. (2019). The art of using t-sne for single-cell transcriptomics. *Nature communications*, *10*(1), 5416. https://doi.org/10.1038/s41467-019-13056-x

Teixeira, J., Rocha, V., Oliveira, J., Jorge, P. A. S., & Silva, N. A. (2022). Towards real-time identification of trapped particles with umap-based classifiers. *Journal of Physics: Conference Series*, *2407*(1), 012043. https://doi.org/10.1088/1742-6596/2407/1/012043

Trozzi, F., Wang, X., & Tao, P. (2021). Umap as a dimensionality reduction tool for molecular dynamics simulations of biomacromolecules: A comparison study. *The Journal of Physical Chemistry B*, *125*, 5022–5034. https://doi.org/10.1021/acs.jpcb.1c02081

Matty, M., Zhang, Y., Senthil, T., & Kim, E.-A. (2021). Entanglement clustering for ground-stateable quantum many-body states. *Physical Review Research*, *3*, 023212. https://doi.org/10.1103/PhysRevResearch.3.023212

Böhm, J. N., Berens, P., & Kobak, D. (2022). Attraction-Repulsion Spectrum in Neighbor Embeddings. *Journal of Machine Learning Research*, *23*(95), 1–32. http://jmlr.org/papers/v23/21-0055.html

Damrich, S., & Hamprecht, F. A. (2021). On UMAP's True Loss Function. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (pp. 5798–5809, Vol. 34). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2021/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf

Kobak, D., & Linderman, G. (2021). Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, *39*, 1–2. https://doi.org/10.1038/s41587-020-00809-z

Kolsbjerg, E. L., Groves, M. N., & Hammer, B. (2016). An automated nudged elastic band method. *The Journal of Chemical Physics*, *145*(9), 094107. https://doi.org/10.1063/1.4961868

Draxler, F., Veschgini, K., Salmhofer, M., & Hamprecht, F. (2018). Essentially No Barriers in Neural Network Energy Landscape. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (pp. 1309–1318, Vol. 80). PMLR. https://proceedings.mlr.press/v80/draxler18a.html

Poličar, P. G. (2023). "How t-SNE works" from the openTSNE documentation. Retrieved May 7, 2024, from https://opentsne.readthedocs.io/en/stable/tsne_algorithm.html

Xiao, H., Rasul, K., & Vollgraf, R. (2017, August 28). *Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms*. arXiv: 1708.07747 [cs.LG].

Packer, J. S., Zhu, Q., Huynh, C., Sivaramakrishnan, P., Preston, E., Dueck, H., Stefanik, D., Tan, K., Trapnell, C., Kim, J., Waterston, R. H., & Murray, J. I. (2019). A lineage-resolved molecular atlas of c. elegans embryogenesis at single-cell resolution. *Science*, *365*(6459), eaax1971. https://doi.org/10.1126/science.aax1971

Narayan, A., Berger, B., & Cho, H. (2021). Assessing single-cell transcriptomic variability through density-preserving data visualization. *Nature Biotechnology*. https://doi.org/10.1038/s41587-020-00801-7

McInnes, L. (2018). Umap-learn documentation. Retrieved May 7, 2024, from https://umap-learn.readthedocs.io/en/latest/index.html

Charlier, B., Feydy, J., Glaunès, J. A., Collin, F.-D., & Durif, G. (2021). Kernel Operations on the GPU, with Autodiff, without Memory Overflows. *Journal of Machine Learning Research*, *22*(74), 1–6. http://jmlr.org/papers/v22/20-275.html

Poličar, P. G., Stražar, M., & Zupan, B. (2024). Opentsne: A modular python library for t-sne dimensionality reduction and embedding. *Journal of Statistical Software*, *109*(3), 1–30. https://doi.org/10.18637/jss.v109.i03

# Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 07.05.2024,