Visualizing hierarchies in scRNA-seq data using a density tree-biased autoencoder

Quentin Garrido, Sebastian Damrich, Alexander Jäger, Dario Cerletti, Manfred Claassen, Laurent Najman, Fred Hamprecht

Visualizing hierarchies in scRNA-seq data using a density tree-biased autoencoder

Quentin Garrido [1,5*] Sebastian Damrich [1] Alexander Jäger [1] Dario Cerletti [2,3]
Manfred Claassen [4] Laurent Najman [5] Fred A. Hamprecht [1]

[1]HCI/IWR, Heidelberg University, Germany
[2] Institute of Molecular Systems Biology, ETH Zürich, Switzerland
[3] Institute of Microbiology, ETH Zürich, Switzerland
[4] Internal Medicine I, University Hospital Tübingen, University of Tübingen, Germany
[5] Université Gustave Eiffel, LIGM, Equipe A3SI, ESIEE, France

Abstract

Single cell RNA sequencing (scRNA-seq) data makes studying the development of cells possible at unparalleled resolution. Given that many cellular differentiation processes are hierarchical, their scRNA-seq data is expected to be approximately tree-shaped in gene expression space. Inference and representation of this tree-structure in two dimensions is highly desirable for biological interpretation and exploratory analysis. Our two contributions are an approach for identifying a meaningful tree structure from high-dimensional scRNA-seq data, and a visualization method respecting the tree-structure. We extract the tree structure by means of a density based minimum spanning tree on a vector quantization of the data and show that it captures biological information well. We then introduce DTAE, a tree-biased autoencoder that emphasizes the tree structure of the data in low dimensional space. We compare to other dimension reduction methods and demonstrate the success of our method experimentally. Our implementation relying on PyTorch [15] and Higra [16] is available at github.com/hci-unihd/DTAE.

1. Introduction

Single-cell RNA sequencing (scRNA-seq) data allows analyzing gene expression profiles at the single-cell level, thus granting insights into cell behavior at unparalleled resolution. In particular, this permits studying the cell development through time more precisely.

[21]'s popular metaphor likens the development of cells to marbles rolling down a landscape. While cells are all grouped at the top of the hill when they are not yet differentiated (e.g. stem cells), as they start rolling down, they can take multiple paths and end up in distinct differentiated states, or cell fates. In the illustrated case, the typical resulting topology of the trajectories is a tree.

However, for every cell, hundreds or thousands of expressed genes are recorded and this data is noisy. To summarize such high-dimensional data, it is useful to visualize it in two or three dimensions.

Our goal, then, is to identify the hierarchical (tree) structure of the scRNA-seq data and subsequently reduce its dimensionality while preserving the extracted hierarchical properties present. We address this in two steps, illustrated in figure 1.

First, we cluster the scRNA-seq data in high-dimensional space to obtain a more concise and robust representation. Then, we capture the hierarchical structure as a minimum spanning tree (MST) on our cluster centers with edge weights reflecting the data density in high-dimensional space. We dub the resulting tree "density tree".

Second, we embed the data to low dimension with an autoencoder, a type of artificial neural network. In addition to the usual aim of reconstructing its input, we bias the autoencoder to also reproduce the density tree in low-dimensional space. As a result, the hierarchical properties of the data are emphasized in our visualization.

2. Related Work

There are various methods for visualizing scRNA-seq data and trajectory inference and many of them have been reviewed for instance in [19]. We therefore mention only some exemplary approaches here. SCORPIUS [5] was one of the first such methods. It is limited to linear topolo-

*quentin.garrido@edu.esiee.fr

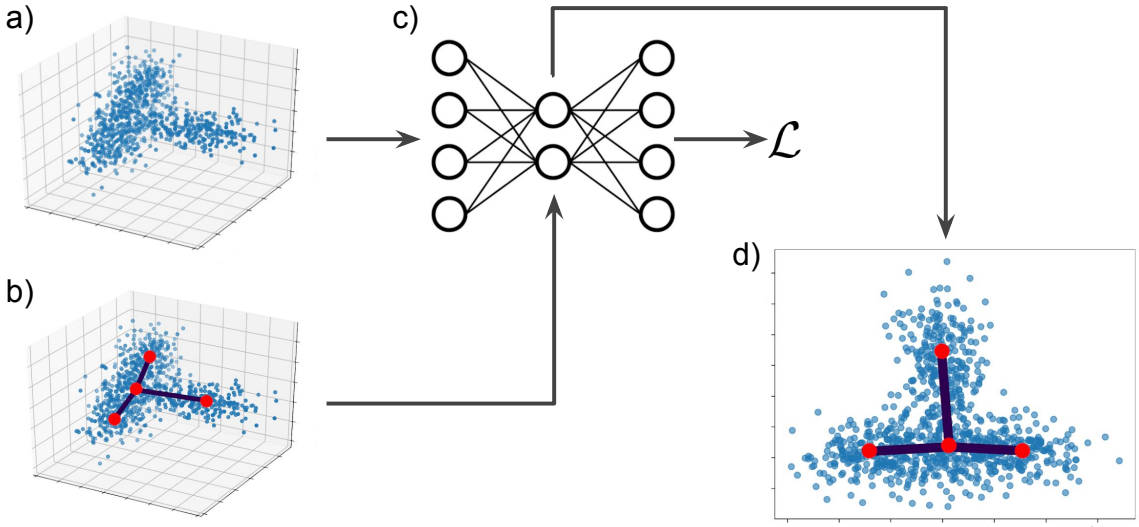

Figure 1. Schematic method overview. **a**) High-dimensional data. **b**) Proposed density tree. After computing the k-means centroids on the data, we build a tree based on the data density between pairs of centroids. **c**) DTAE. An autoencoder is used to learn a representation of our data. This embedding is regularized by the previously computed tree in order to preserve its hierarchical structure in low-dimensional space. **d**) The final DTAE embedding. After training of the autoencoder, the bottleneck layer visualizes the data in low dimension and respects the density structure.

gies rather than trees. Improvements include SLINGSHOT [20] which infers trajectories using any dimensionality reduction method; MONOCLE 2 [18] which embeds a tree on k-means centroids without relying on a neural network; SPADE [4, 17] which downsamples to equalize the data density and computes an MST on agglomerative clusters, but does not inform the MST by the actual data density and only visualizes the tree itself; PAGA [22] which learns a hierarchical graph representation of the data and PHATE [13] which computes diffusion probabilities on the data before applying multi-dimensional scaling.

Another noteworthy method, Poincaré maps [9], produces an embedding in hyperbolic space which is better suited for representing trees than Euclidean space.

The general purpose dimension reduction methods t-SNE [10] and UMAP [12, 3] are also popular for visualizing scRNA-seq data.

Other recent methods rely on neural networks, and are thus more similar to ours. Like our method, SAUCIE [1] relies on an autoencoder but focuses more on batch effect removal than hierarchical properties. Topological autoencoders [14] are closest to our idea of retaining topological properties during dimension reduction. Their method is couched in terms of Persistent Homology, but as they use only 0-dimensional topological features, this actually amounts to preserving the MST of the data. They use Euclidean distance and compute the MST on all points which produces less stable results than the proposed density-based approach on cluster centroids.

3. Approximating the High-dimensional scRNA-seq Data with a Tree

To summarize the high-dimensional data in terms of a tree the minimum spanning tree (MST) on the Euclidean distances is an obvious choice. This route is followed by [14] who reproduce the MST obtained on their high-dimensional data in their low-dimensional embedding.

However, scRNA-seq data can be noisy, and an MST built on all of our data is very sensitive to noise. Therefore, we first run k-means clustering on the original data yielding more robust centroids for the MST construction and also reducing downstream complexity.

A problem with the Euclidean MST, illustrated in figure 2, is that two centroids can be close in Euclidean space without having many data points between them. In such a case an Euclidean MST would not capture the skeleton of our original data well.

But it is crucial that the extracted tree follows the dense regions of the data if we want to visualize developmental trajectories of differentiating cells: A trajectory is plausible if we observe intermediate cell states and unlikely if there are jumps in the development. By preferring tree edges in high density regions of the data we ensure that the computed spanning tree is biologically plausible.

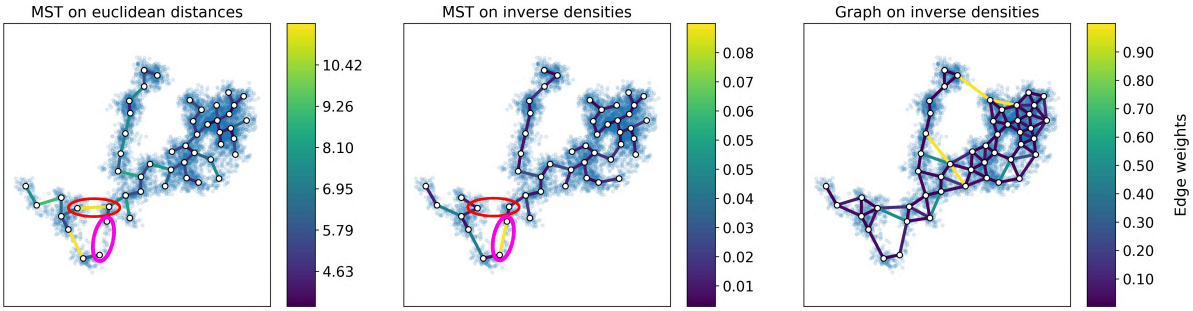

Figure 2. (a, b) Comparison of MST on k-means centroids using Euclidean distance or density weights. The data was generated using the PHATE library [13], with 3 branches in 2D. Original data points are transparently overlayed to better visualize their density. While the MST based on the Euclidean distance places connections between centroids that are close but have only few data points between them (see red ellipse), our MST based on the data density instead includes those edges that lie in high density regions (see pink ellipse). (c) Complete graph over centroids and its Hebbian edge weights. Infinite-weight edges, that is edges not supported by data, are omitted for clarity.

Following this rationale, we build the MST on the complete graph over centroids whose edge weights are given by the density of the data along each edge instead of its Euclidean distance. This results in a tree that we believe captures Waddington's hypothesis better than merely considering cumulative differences in expression levels.

To estimate the support that a data sample provides for an edge, we follow [11]. Consider the complete graph $G = (C, E)$ such that $C = \{c_1, \dots, c_k\}$ is the set of centroids. In the spirit of Hebbian learning, we count, for each edge, how often its incident vertices are the two closest centroids to any given datum. As pointed out by [11] this amounts to an empirical estimate of the integral of the density of observations across the second-order Voronoï region associated with this pair of cluster centers. Finally, we compute the maximum spanning tree over these Hebbian edge weights or, equivalently, the minimum spanning tree over their inverses. Our strategy for building the tree is summarized in algorithm 1.

Our data-density based tree follows the true shape of the data more closely than a MST based on the Euclidean distance weights as illustrated in figure 2. We claim this indicates it being a better choice for capturing developmental trajectories.

Having extracted the tree shape in high dimensions our goal is to reproduce this tree as closely as possible in our embedding.

4. Density-Tree biased Autoencoder (DTAE)

We use an autoencoder to faithfully embed the high-dimensional scRNA-seq data in a low-dimensional space and bias it such that the topology inferred in high-dimensional space is respected. An autoencoder is an artificial neural network consisting of two concatenated subnetworks, the encoder f, which maps the input to lower-dimensional space, also called embedding space, and the decoder g, which tries to reconstruct the input from the lower-dimensional embedding. It can be seen as a non-linear generalization of PCA. We visualize the low-dimensional embeddings $h_i = f(x_i)$ and hence choose their dimension to be 2.

The autoencoder is trained by minimizing the following loss terms, including new ones that bias the autoencoder to also adhere to the tree structure.

Algorithm 1 Density tree generation

Require: High-dimensional data $X \in \mathbb{R}^{n \times d}$
Require: Number of k-means centroids k
procedure GENERATETREE(X, k)
$C \leftarrow$ KMEANS(X, k) ▷ $\mathcal{O}(nkdt)$ with t the number of iterations
$G = (C, E)$ the complete graph on our centroids
for $\{i, j\}$ a two-element subset of $\{1, \dots, k\}$ **do** ▷ $\mathcal{O}(k^2)$
$d_{i,j} = 0$
end for
for $i = 1, \dots, |X|$ **do** ▷ $\mathcal{O}(nk)$
$a \leftarrow \arg\min_{j=1,\dots,k} ||x_i - c_j||_2$ ▷ Nearest centroid
$b \leftarrow \arg\min_{\substack{j=1,\dots,k \\ j \neq a}} ||x_i - c_j||_2$ ▷ Second nearest centroid
$d_{a,b} = d_{a,b} + 1$ ▷ Increase nearest centroids' edge strength
end for
for $\{i, j\}$ a two-element subset of $\{1, \dots, k\}$ **do** ▷ $\mathcal{O}(k^2)$
$W_{i,j} \leftarrow d_{i,j}^{-1}$ ▷ Edge weights are inverse edge strengths
end for
$T \leftarrow \text{MST}(G, W)$ ▷ $\mathcal{O}(k^2 \log k)$
return T, d ▷ Retains the density tree and the edge strengths
end procedure

4.1. Reconstruction Loss

The first term of the loss is the reconstruction loss, defined as

$$\mathcal{L}_{\text{rec}} = \text{MSE}(X, g(f(X))) = \frac{1}{N} \sum_{x_i \in X} ||x_i - g(f(x_i))||_2^2. \tag{1}$$

This term is the typical loss function for an autoencoder and ensures that the embedding is as faithful to the original data as possible, forcing it to extract the most salient data features.

4.2. Push-Pull Loss

The main loss term that biases the DTAE towards the density tree is the push-pull loss. It trains the encoder to embed the data points such that the high-dimensional data density and in particular the density tree are reproduced in low-dimensional-space.

We find a centroid in embedding space by averaging the embeddings of all points assigned to the corresponding k-means cluster in high-dimensional space. In this way, we can easily relate the centroids in high and low dimension and will simply speak of centroids when the ambient space is clear from the context.

For a given state of the encoder and a resulting embedding $h_i = f(x_i)$, $x_i \in X$, we define $c_{h_i,1}$ and $c_{h_i,2}$ as the two centroids closest to h_i in embedding space. We similarly define $c'_{h_i,1}$ and $c'_{h_i,2}$ as the low-dimensional centroids corresponding to the two closest centroids of x_i in high-dimensional space. As long as $c'_{h_i,1}$, $c'_{h_i,2}$ differ from $c_{h_i,1}$ and $c_{h_i,2}$, the encoder places h_i next to different centroids than in high-dimensional space. To ameliorate this, we want to move $c'_{h_i,1}$, $c'_{h_i,2}$ and h_i towards each other while separating $c_{h_i,1}$ and $c_{h_i,2}$ form h_i. The following preliminary version of our push-pull loss implements this:

$$\tilde{\mathcal{L}}_{\text{push}}(h_i) = -\left(||h_i - c_{h_i,1}||_2 + ||h_i - c_{h_i,2}||_2\right)^2 \tag{2}$$

$$\tilde{\mathcal{L}}_{\text{pull}}(h_i) = \left(||h_i - c'_{h_i,1}||_2 + ||h_i - c'_{h_i,2}||_2\right)^2 \tag{3}$$

$$\tilde{\mathcal{L}}_{\text{push-pull}} = \frac{1}{N} \sum_{x_i \in X} \tilde{\mathcal{L}}_{\text{push}}(f(x_i)) + \tilde{\mathcal{L}}_{\text{pull}}(f(x_i)). \tag{4}$$

We have inserted $h_i = f(x_i)$ in equation (4).

The push loss decreases as h_i and the currently closest centroids, $c_{h_i,1}$ and $c_{h_i,2}$ are placed further apart from each other, while the pull loss decreases when h_i gets closer to the correct centroids $c'_{h_i,1}$ and $c'_{h_i,2}$. Indeed, the push-pull loss term is minimized if and only if each embedding h_i lies in the second-order Voronoï region of those low-dimensional centroids whose high-dimensional counterparts contain the data point x_i in their second-order Voronoï region. In other words, the loss is zero precisely when we are reproducing the edge densities from high dimension in low dimension.

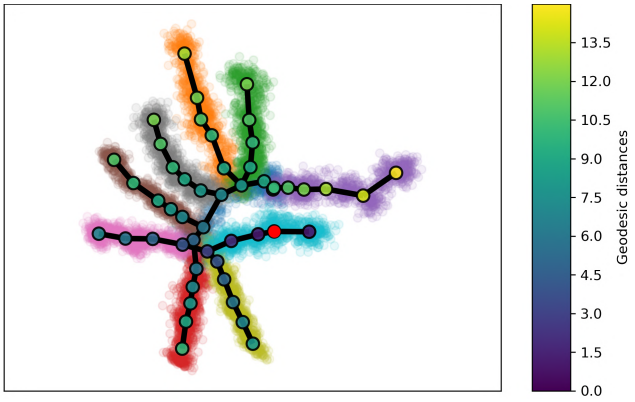

Figure 3. Density-tree on the low-dimensional centroids and superimposed on the DTAE embedded data, which is colored by ground truth branches. The vertex colors correspond to their geodesic distance to the red vertex. The data was generated using the PHATE library.

Note that we let the gradient flow through both the individual embeddings and through the centroids which are means of embeddings themselves.

This naïve formulation of the push-pull loss has the drawback that it can become very small if all embeddings are nearly collapsed into a single point, which is undesirable for visualization. Therefore, we normalize the contribution of every embedding h_i by the distance between the two correct centroids in embedding space. This prevents the collapsing of embeddings and also ensures that each datapoint x_i contributes equally regardless of how far apart their two closest centroids are in embedding space. The push-pull loss thus becomes

$$\mathcal{L}_{\text{push}}(h_i) = -\left(\frac{||h_i - c_{h_i,1}||_2 + ||h_i - c_{h_i,2}||_2}{||c'_{h_i,1} - c'_{h_i,2}||_2}\right)^2 \tag{5}$$

$$\mathcal{L}_{\text{pull}}(h_i) = \left(\frac{||h_i - c'_{h_i,1}||_2 + ||h_i - c'_{h_i,2}||_2}{||c'_{h_i,1} - c'_{h_i,2}||_2}\right)^2 \tag{6}$$

$$\mathcal{L}_{\text{push-pull}} = \frac{1}{N} \sum_{x_i \in X} \mathcal{L}_{\text{push}}(f(x_i)) + \mathcal{L}_{\text{pull}}(f(x_i)). \tag{7}$$

So far, we only used the density information from high-dimensional space for the embedding, but not the extracted density tree itself. The push-pull loss in equation (7) is agnostic to the positions of the involved centroids within the density tree, only their Euclidean distance to the embedding h_i matters. In contrast, the hierarchical structure is important for the biological interpretation of the data: It is much less important if an embedding is placed close to two centroids that are on the same branch of the density tree than it is if the embedding is placed between two different

branches. In the first case, cells are just not ordered correctly within a trajectory, while in the second case we get false evidence for an altogether different pathway. The situation is illustrated on toy data in figure 3. There are many points between the red centroid on the cyan branch and the purple branch, which can falsely indicate a circluar trajectory.

We tackle this problem by reweighing the push-pull loss with the geodesic distance along the density tree. The geodesic distance $d_{\text{geo}}(c_i, c_j)$ with $c_i, c_j \in C$ is defined as the number of edges in the shortest path between c_i and c_j in the density tree. By correspondence between centroids in high- and low-dimensional space, we can extend the definition to centroids in embedding space.

Centroids at the end of different branches in the density have a higher geodesic distance than centroids nearby on the same branch, see figure 3. By weighing the push-pull loss contribution of an embedded point by the geodesic distance between its two currently closest centroids, we focus the push-pull loss on embeddings which erroneously lie between different branches.

The geodesic distances can be computed quickly in $\mathcal{O}(k^2)$ via breadth first search and this only has to be done once before training the autoencoder.

The final version of our push-pull loss becomes

$$\mathcal{L}_{\text{push-pull}} = \frac{1}{N} \sum_{x_i \in X} \Big(d_{\text{geo}}(c_{f(x_i),1}, c_{f(x_i),2}) \cdot (\mathcal{L}_{\text{push}}(f(x_i)) + \mathcal{L}_{\text{pull}}(f(x_i))) \Big). \tag{8}$$

Note, that the normalized push-pull loss in equation (7) and the geodesically reweighted push-pull loss in (8) both also get minimized if and only if the closest centroids in embedding space correspond to the closest centroids in high-dimensional space.

4.3. Compactness loss

The push-pull loss replicates the empirical high-dimensional data density in embedding space by moving the embeddings into the correct second-order Voronoï region, which can be large or unbounded. This does not suffice for an evidently visible tree structure in which the skeleton of the embedded data should be locally one-dimensional. More precisely, an embedding should not only be in the correct second-order Voronoï region, but lie compactly around the line between its two centroids. To achieve this, we add the compactness loss, which is just another instance of the pull loss

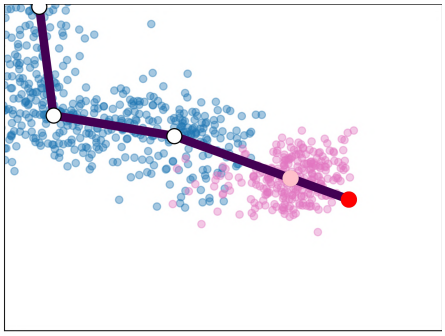

Figure 4. Illustration of a ghost point and its addition to the minimum spanning tree. The pink vertex is obtained as the mean of the red points. It represent the end of a branch and the red vertex the corresponding ghost point. Without the ghost point, all pink points want to lie on the edge on the left of the pink vertex. This is incompatible with the definition of the pink vertex as the mean of the pink points, motivating the addition of ghost points.

$$\mathcal{L}_{\text{comp}} = \frac{1}{N} \sum_{x_i \in X} \left(\frac{||h_i - c'_{h_i,1}||_2 + ||h_i - c'_{h_i,2}||_2}{||c'_{h_i,1} - c'_{h_i,2}||_2} \right)^2 \tag{9}$$

$$= \frac{1}{N} \sum_{x_i \in X} \mathcal{L}_{\text{pull}}(h_i), \tag{10}$$

where we wrote h_i instead of $f(x_i)$ for succinctness. The compactness loss is minimized if the embedding h_i is exactly between the correct centroids $c'_{h_i,1}$ and $c'_{h_i,2}$ and has elliptic contour lines with foci at the centroids.

4.4. Cosine loss

Since the encoder is a powerful non-linear map it can introduce artifactual curves in the low-dimensional tree branches. However, especially tight turns can impede the visual clarity of the embedding. As a remedy, we propose an optional additional loss term that tends to straighten branches.

Centroids at which the embedding should be straight are the ones within a branch, but not at a branching event of the density tree. The former can easily be identified as the centroids of degree 2.

Let c be a centroid in embedding space of degree 2 with its two neighboring centroids $n_{c,1}$ and $n_{c,2}$. The branch is straight at c if the two vectors $c - n_{c,1}$ and $n_{c,2} - c$ are parallel or, equivalently, if their cosine is maximal. Denoting by $C_2 = \{c \in C \mid \deg(c) = 2\}$ the set of all centroids of degree 2, considered in embedding space, we define the cosine loss as

$$\mathcal{L}_{\text{cosine}} = 1 - \frac{1}{|C_2|} \sum_{c \in C_2} \frac{(c - n_{c,1}) \cdot (n_{c,2} - c)}{||c - n_{c,1}||_2 \, ||n_{c,2} - c||_2}. \tag{11}$$

Essentially, it measures the cosine of the angles along the tree branch and becomes minimal if all these angles are zero and the branches straight.

4.5. Complete loss function

Combining the four loss terms of the preceding sections, we arrive at our final loss

$$\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{push-pull}}\mathcal{L}_{\text{push-pull}} + \lambda_{\text{comp}}\mathcal{L}_{\text{comp}} + \lambda_{\text{cos}}\mathcal{L}_{\text{cos}}. \tag{12}$$

The relative importance of the loss terms, especially of $\mathcal{L}_{\text{comp}}$ and $\mathcal{L}_{\text{cos}}$, which control finer aspects of the visualization, might depend on the use-case. In practice, we found $\lambda_{\text{rec}} = \lambda_{\text{push-pull}} = \lambda_{\text{comp}} = 1$ and $\lambda_{\text{cos}} = 15$ to work well.

4.6. Ghost points

As defined, our loss function can never be zero, as not all points at the end of a branch can lie between the desired centroids. To illustrate the problem consider a leaf centroid c and the set S of embedding points whose mean it is. Typically, these embeddings should lie on the line from the leaf centroid to the penultimate centroid $\tilde{c}$ of the branch according to the compactness loss. In particular, the compactness loss pulls them all to one side of the leaf centroid, which is futile as the leaf centroid is by construction their mean, see figure 4. To alleviate this small technical issue we add pairs of "ghost points" g in high- and low-dimensional space at

$$g = \tilde{c} + \frac{1}{2} \times (c - \tilde{c}). \tag{13}$$

After computing the density tree, we change it by adding the ghost points as new leaves and allowing the data points corresponding to S to choose ghost points as one of their nearest centroids. The push-pull and compactness losses consider ghost points in embedding space like normal centroids. As the ghost points are not a mean of embedding points, all loss terms can now theoretically reach zero. As an additional benefit the ghost centroids help to separate the ends of branches from other branches in embedding space.

4.7. Training procedure

Firstly, we compute the k-means centroids, the edge densities, the density tree, ghost points and geodesic distances. This has to be done only once as an initialization step, see algorithm 2. Secondly, we pretrain the autoencoder with only the reconstruction loss via stochastic gradient descent on minibatches. This provides a warm start for finetuning the autoencoder with all losses in the third step.

During finetuning, all embedding points are needed to compute the centroids in embedding space. Therefore, we perform full-batch gradient descent during finetuning. The full training procedure is described in algorithm 3.

We always used $k = 50$ centroids for k-means clustering in our experiments. Our autoencoder always has a bottleneck dimension of 2 for visualization. In the experiments we used intermediate layer dimensions $d(\text{input dimension}), 2048, 256, 32, 2, 32, 256, 2048, d$. We omitted hidden layers of dimension larger than the input. We use fully connected layers and ReLU activations after every layer but the last encoder and decoder layer and employ the Adam [8] optimizer with learning rate 2×10^{-4} for pretraining and 1×10^{-3} for finetuning unless stated otherwise. We used a batch size of 256 for pretraining in all experiments.

Algorithm 2 Initialization

Require: Input data X
$T \leftarrow \text{TREEGENERATION}(X)$
$C_{\text{ghost}} \leftarrow \text{GHOSTPOINTS}(C)$
$C \leftarrow C \cup C_{\text{ghost}}$
$\text{UPDATECLOSESTCENTROIDS}(X, C)$ ▷ Allow choosing a ghost point
$d_{\text{geo}} \leftarrow \text{GEODESICDISTANCE}(T)$
$C_2 \leftarrow \{c \in C \mid \deg(c) = 2\}$

Algorithm 3 Training loop

Require: Autoencoder $(g \circ f)_\theta$
Require: Pretraining epochs n_p, batch size b and learning rate α_p
Require: Finetuning epochs n_f and learning rate α_f
Require: Weight parameters for the loss λ_{rec},$\lambda_{\text{push-pull}}$,$\lambda_{\text{comp}}$,$\lambda_{\text{cos}}$
$T, C, C_2, d_{geo} \leftarrow \text{INITIALIZATION}(X)$
$\#Pretraining$
for $t = 0, 1, \ldots, n_p$ **do**
  for $i = 0, 1, \ldots, n_p/b$ **do**
    Sample a minibatch m from X
    $\hat{m} \leftarrow g(f(m))$
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{rec}}$
    $\theta^{t+1} \leftarrow \theta^t - \alpha_p \nabla \mathcal{L}$
  end for
end for
$\#Finetuning$
for $t = n_p, \ldots, n_p + n_f$ **do**
  $h \leftarrow f(X)$
  $\hat{X} \leftarrow g(h)$
  $\mathcal{L} \leftarrow \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{push-pull}}\mathcal{L}_{\text{push-pull}} + \lambda_{\text{comp}}\mathcal{L}_{\text{comp}} + \lambda_{\text{cos}}\mathcal{L}_{\text{cos}}$
  $\theta^{t+1} \leftarrow \theta^t - \alpha_f \nabla \mathcal{L}$
end for

5. Results

In this section we show the performance of our method on toy and real scRNA-seq datasets and compare it to a vanilla autoencoder, PHATE, UMAP and PCA.

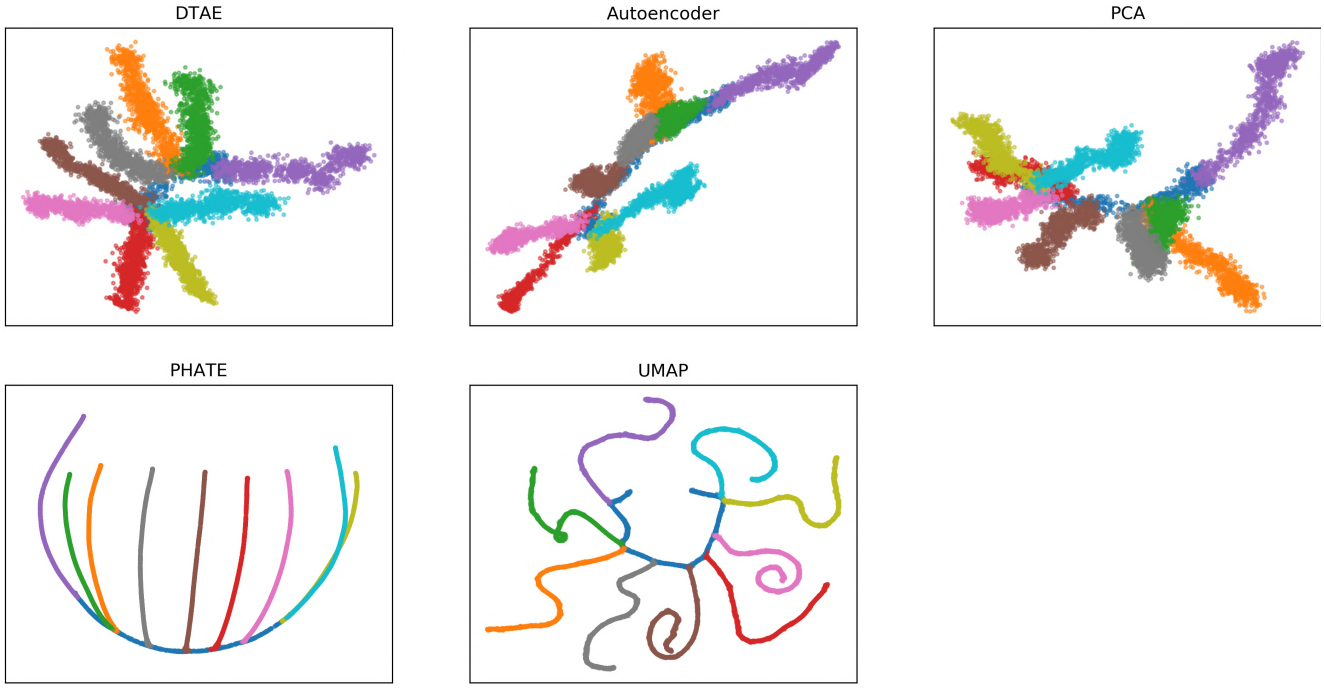

Figure 5. Results obtained using data generated by the PHATE library. Branches are coloured by groundtruth labels.

5.1. PHATE generated data

We applied our method to an artificial dataset created with the library published alongside [13], to demonstrate its functionality in a controlled setting. We generated a toy dataset whose skeleton is a tree with one backbone branch and 9 branches emanating from the backbone consisting in total of 10,000 points in 100 dimensions.

We pretrained for 150 epochs with a learning rate of 10^{-3} and finetuned for another 150 epochs with a learning rate of 10^{-2}.

Figure 5 shows the visualization results. The finetuning significantly improves the results of the pretrained autoencoder, whose visualisation collapses the grey and green branch onto the blue branch. Overall, DTAE, PHATE and UMAP achieve satisfactory results that make the true tree structure of the data evident. While PHATE and UMAP produce overly crisp branches compared to the PCA result, the reconstruction loss of our autoencoder guards us from collapsing the branches into lines. PHATE appears to overlap the cyan and yellow branches near the backbone and UMAP introduces artificially curved branches. The results on this toy dataset demonstrate that our method can embed high-dimensional hierarchical data into 2D and emphasize its tree-structure while avoiding to collapse too much information compared to state-of-the-art methods. In our method all branches are easily visible.

5.2. Endocrine pancreatic cell data

We evaluated our method on the data from [2]. It represents endocrine pancreatic cells at different stages of their development and consists of gene expression information for 36351 cells and 3999 gens. Preprocessing information can be found in [2]. We pretrained for 300 epochs and used 250 epochs for finetuning.

Figures 6 and 7 depicts visualizations of the embryonic pancreas development with different methods. Our method can faithfully reproduce the tree structure of the data, especially for the endocrine subtypes. The visualized hierarchy is biologically plausible, with a particularly clear depiction of the α-, β- and ε-cell branches and a visible, albeit too strong, separation of the δ-cells. This is in agreement with the results from [2]. UMAP also performs very well and attaches the δ-cells to the main trajectory. But it does not exhibit the ε-cells are a distinct branch and the α- and β-cell branches are not as prominent as in DTAE. PHATE does not manage to separate the δ- and ε-cells discernibly from the other endocrine subtypes. As on toy data in figure 5, it produces overly crisp branches for the α- and β-cells. PCA mostly overlays all endocrine subtypes. All methods but the vanilla autoencoder show a clear branch with tip and ancinar cells and one via EP and Fev+ cells to the endocrine subtypes, but only our method manages to also hint at the more generic trunk and multipotent cells from which these two major branches emanate. The ductal and Ngn3 low EP cells overlap in all methods.

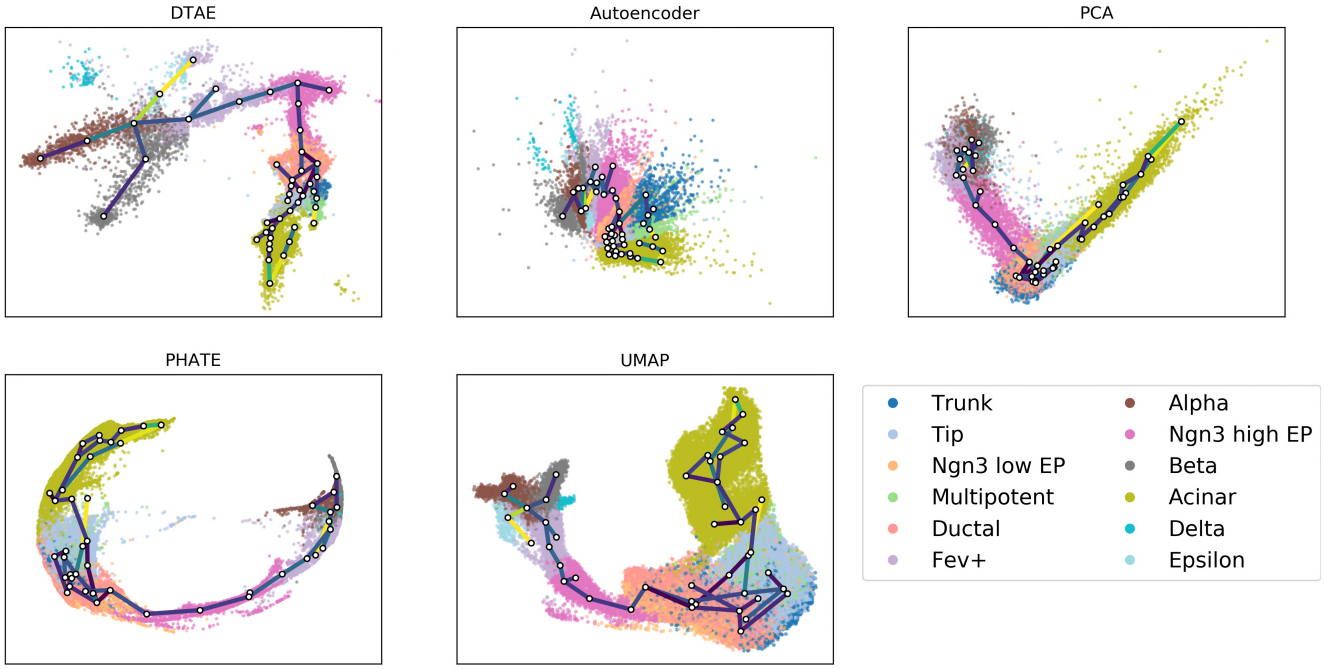

Figure 6. Pruned density MST superimposed over our results on the endocrine pancreatic cell dataset, coloured by cell subtypes. We use finer labels for the endocrine cells. Darker edges represent denser edges. Only edges with more than 100 points contributing to them are plotted here

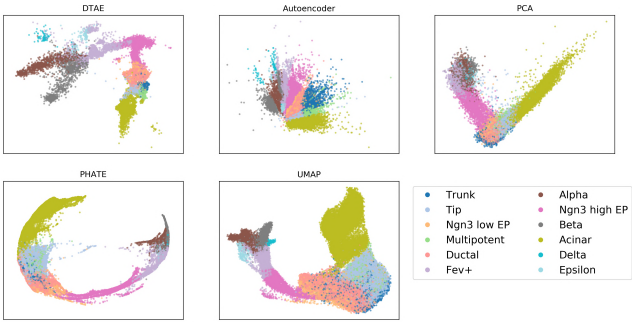

Figure 7. Results obtained on the endocrine pancreatic cells dataset. Colours correspond to cell types, with finer labels for the endocrine cells.

It is worth noting that the autoencoder alone was not able to visualize meaningful hierarchical properties of the data. However, the density tree-biased finetuning in DTAE made this structure evident, highlighting the benefits of our approach.

In figure 6, we overlay DTAE's embedding with a pruned version of the density tree and see that the visualization closely follows the tree structure around the differentiated endocrine cells. This combined representation of low-dimensional embedding and overlayed density tree further facilitates the identification of branching events and shows the full power of our method. It also provides an explanation for the apparent separation of the δ-cells. Since there are relatively few δ-cells, they are not represented by a distinct k-means centroid.

Our method places more k-means centroids in the dense region in the lower right part of DTAE's panel in figure 6 than is appropriate to capture the trajectories, resulting in many small branches. Fortunately, this does not result in an exaggerated tree-shaped visualization that follows every spurious branch, which we hypothesize is thanks to the successful interplay between the tree bias and the reconstruction aim of the autoencoder: If the biological signal encoded in the gene expressions can be reconstructed by the decoder from an embedding with enhanced hierarchical structure, the tree-bias shapes the visualization accordingly. Conversely, an inappropriate tree-shape is prevented if it would impair the reconstruction. Overall, the density tree recovers the pathways identified in [2] to a large extent. Only the trajectory from multipotent via tip to ancinar cells includes an unexpected detour via the trunk and ductal cells, which the autoencoder mends by placing the tip next to the multipotent cells.

The density tree also provides useful information in conjunction with other dimension reduction methods. In figure 6, we also overlay their visualizations with the pruned density tree by computing the centroids in the respective embedding spaces according to the k-means cluster assignments. The density tree can help finding branching events and gain insights into the hierarchical structure of the data that is visualized with an existing dimension reduction

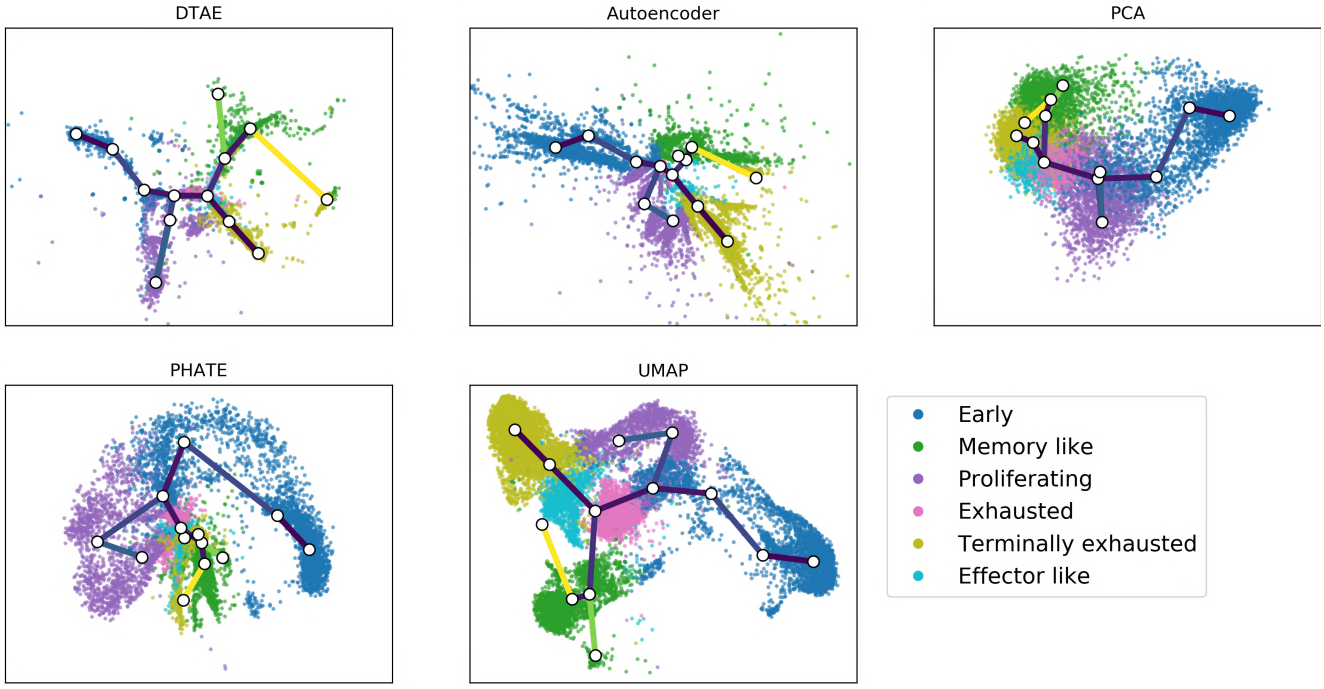

Figure 8. Pruned density MST superimposed over our results on the chronic part of the T-cell data, coloured by phenotypes. Darker edges represent denser edges. Only edges with more than 100 points contributing to them are plotted here

method. For instance, together with the density tree, we can identify the ε-cells as a separate branch and find the location of the branching event into different endocrine subtypes in the UMAP embedding.

5.3. T-cell infection data

We further applied our method to T-cell data of a chronic and an acute infection, which was shared with us by the authors of [6]. The data was preprocessed using the method described in [23], for more details confer [6]. It contains gene expression information for 19029 cells and 4999 genes. While we used the combined dataset to fit all dimension reduction methods, we only visualize the 13707 cells of the chronic infection for which we have phenotype annotations from [6] allowing us to judge visualization quality from a biological viewpoint. We pretrained for 600 epochs and used 250 epochs for finetuning.

Figures 8 and 9 demonstrate that our method makes the tree structure of the data clearly visible. The visualized hierarchy is also biologically significant: The two branches on the right correspond to the memory-like and terminally exhausted phenotypic states, which are identified as the main terminal fates of the differentiation process in [6]. Furthermore, the purple branch at the bottom contains the proliferating cells. Since the cell cycle affects cell transcription significantly, those cells are expected to be distinct from the rest.

It is encouraging that DTAE makes the expected biological structure apparent even without relying on known marker genes or differential cell expression, which were used to obtain the phenotypic annotations in [6].

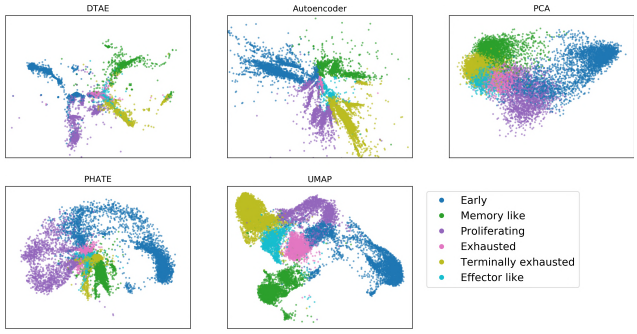

Figure 9. Results obtained on the chronic infection from the T-cell dataset, coloured by phenotypes.

Interestingly, our method places the branching event towards the memory-like cells in the vicinity of the exhausted cells, as does UMAP, while [6] recognized a trajectory directly from the early stage cells to the memory-like fate. The exact location of a branching event in a cell differentiation process is difficult to determine precisely. We conjecture that fitting the dimensionality reduction methods on the gene expression measurements of cells from an acute infection in addition to those from the chronic infection analyzed in [6] provided additional evidence for the trajectory via exhausted cells to the memory-like fate. Unfortunately,

an in-depth investigation of this phenomenon is beyond the scope of this methodological paper.

The competing methods expose the tree-structure of the data less obviously than DTAE. The finetuning significantly improves the results from the autoencoder, which shows no discernible hierarchical structure. PHATE separates the early cells, proliferating cells and the rest. But its layout is very tight around the biologically interesting branching event towards memory-like and terminally exhausted cells. PCA exhibits only the coarsest structure and fails to separate the later states visibly. The biological structure is decently preserved in the UMAP visualization but the hierarchy is less apparent than in DTAE. Overall, our method arguably outperforms the other visualization methods on this dataset.

In figure 8, we again overlay our embedding with a pruned version of the density tree and see that DTAE's visualization indeed closely follows the tree structure. It is noteworthy that even the circular behavior of proliferation cells is accurately captured by a self-overlayed branch although our tree-based method is not directly designed to extract circular structure.

Figure 8 also shows the other dimension reduction methods in conjunction with the pruned density tree. Reassuringly, we find that all methods embed the tree in a plausible way, i.e. without many self-intersections or oscillating branches. This is evidence that our density tree indeed captures a meaningful tree structure of the data. As for the endocrine pancreas dataset, the density tree can enhance hierarchical structure in visualizations of existing dimension reduction methods. It, for example, clarifies in the UMAP plot that the pathway towards the terminally exhausted cells is via the exhausted and effector like cells and not directy via the proliferating cells.

6. Limitations

6.1. Hierarchy assumption

Our method is tailored to Waddington's hierarchical structure assumption of developmental cell populations in which highest data density is along the developmental trajectory. It produces convincing results in this setting as shown above. However, if the assumption is violated, for instance because the dataset contains multiple separate developmental hierarchies or a mixture of hierarchies and distinct clusters of fully differentiated cell fates, the density tree cannot possibly be a faithful representation of the dataset. Indeed, in such a case our method yields a poor result. As an example confer figure 10 with visualizations of the dentate gyrus dataset from [7], preprocessed according to [23]. This dataset consists of a mostly linear cell trajectory and several distinct clusters of differentiated cells and consequently does not meet our model's assumption. Indeed, DTAE manages to only extract some linear structures, but overall fails on this dataset similar to PHATE. UMAP seems to produce the most useful visualization here.

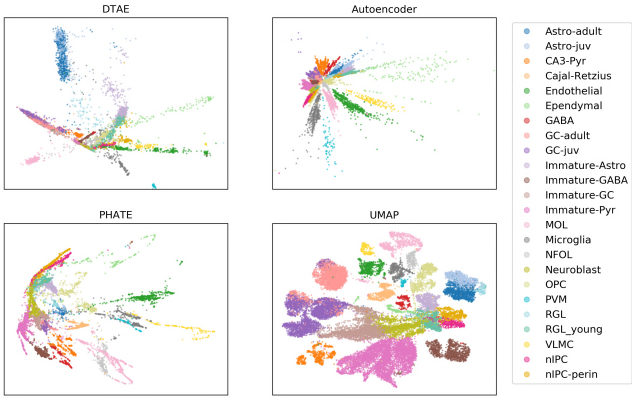

Figure 10. Failure case: Highly clustered data violates our underling assumption of a tree structure. Dentate gyrus data from [7] with clusters coloured by groundtruth cluster assignments.

One could adapt our method by extracting a forest of disconnected density trees by cutting edges below a density threshold. However, if little is known a priori about the structure of the dataset a more general dimension reduction method might be preferable for initial data exploration.

6.2. Neural network limitations

Artificial neural networks are powerful non-linear functions that can produce impressive results. Unfortunately, they require the choice of a number of hyperparameters, such as the dimension of the hidden layers and the learning rate, making them less end-user friendly than their classical counterparts.

7. Conclusion

We have introduced a new way of capturing the hierarchical properties of scRNA-seq data of a developing cell population with a density based minimum spanning tree. This tree is a hierarchical representation of the data that places edges in high density regions and thus captures biologically plausible trajectories. The density tree can be used to inform any dimension reduction method about the hierarchical nature of the data.

Moreover, we used the density tree to bias an autoencoder and were thus able to produce promising visualizations exhibiting clearly visible tree-structure both on synthetic and real world scRNA-seq data of developing cell populations.

References

[1] Matthew Amodio, David van Dijk, Krishnan Srinivasan, William S. Chen, Hussein Mohsen, Kevin R. Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha

Venkataswamy, Anita Desai, V. Ravi, Priti Kumar, Ruth Montgomery, Guy Wolf, and Smita Krishnaswamy. Exploring single-cell data with deep multitasking neural networks. *Nature Methods*, 16(11):1139–1145, Nov. 2019.

[2] Aimée Bastidas-Ponce, Sophie Tritschler, Leander Dony, Katharina Scheibner, Marta Tarquis-Medina, Ciro Salinno, Silvia Schirge, Ingo Burtscher, Anika Böttcher, Fabian J. Theis, Heiko Lickert, and Mostafa Bakhti. Comprehensive single cell mRNA profiling reveals a detailed roadmap for pancreatic endocrinogenesis. *Development*, 146(12):dev173849, June 2019.

[3] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel W. H. Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W. Newell. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology*, 37(1):38–44, Jan. 2019. Number: 1 Publisher: Nature Publishing Group.

[4] Sean C Bendall, Erin F Simonds, Peng Qiu, D Amir El-ad, Peter O Krutzik, Rachel Finck, Robert V Bruggner, Rachel Melamed, Angelica Trejo, Olga I Ornatsky, et al. Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. *Science*, 332(6030):687–696, 2011.

[5] Robrecht Cannoodt, Wouter Saelens, Dorine Sichien, Simon Tavernier, Sophie Janssens, Martin Guilliams, Bart Lambrecht, Katleen De Preter, and Yvan Saeys. SCORPIUS improves trajectory inference and identifies novel modules in dendritic cell development. preprint, Bioinformatics, Oct. 2016.

[6] Dario Cerletti, Ioana Sandu, Revant Gupta, Annette Oxenius, and Manfred Claassen. Fate trajectories of CD8 $^+$ T cells in chronic LCMV infection. preprint, Immunology, Dec. 2020.

[7] Hannah Hochgerner, Amit Zeisel, Peter Lönnerberg, and Sten Linnarsson. Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell RNA sequencing. *Nature Neuroscience*, 21(2):290–299, Feb. 2018.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[9] Anna Klimovskaia, David Lopez-Paz, Léon Bottou, and Maximilian Nickel. Poincaré maps for analyzing complex hierarchies in single-cell data. *Nature Communications*, 11(1):2966, Dec. 2020.

[10] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[11] Thomas Martinetz and Klaus Schulten. Topology representing networks. *Neural Networks*, 7(3):507–522, Jan. 1994.

[12] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, Sept. 2020. arXiv: 1802.03426.

[13] Kevin R. Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B. Burkhardt, William S. Chen, Kristina Yim, Antonia van den Elzen, Matthew J. Hirn, Ronald R. Coifman, Natalia B. Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37(12):1482–1492, Dec. 2019.

[14] Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological Autoencoders. *arXiv:1906.00722 [cs, math, stat]*, Feb. 2020. arXiv: 1906.00722.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv:1912.01703 [cs, stat]*, Dec. 2019. arXiv: 1912.01703.

[16] B. Perret, G. Chierchia, J. Cousty, S.J. F. Guimarães, Y. Kenmochi, and L. Najman. Higra: Hierarchical Graph Analysis. *SoftwareX*, 10:100335, July 2019.

[17] Peng Qiu, Erin F Simonds, Sean C Bendall, Kenneth D Gibbs, Robert V Bruggner, Michael D Linderman, Karen Sachs, Garry P Nolan, and Sylvia K Plevritis. Extracting a cellular hierarchy from high-dimensional cytometry data with spade. *Nature biotechnology*, 29(10):886–891, 2011.

[18] Xiaojie Qiu, Qi Mao, Ying Tang, Li Wang, Raghav Chawla, Hannah A Pliner, and Cole Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nature Methods*, 14(10):979–982, Oct. 2017.

[19] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37(5):547–554, May 2019.

[20] Kelly Street, Davide Risso, Russell B. Fletcher, Diya Das, John Ngai, Nir Yosef, Elizabeth Purdom, and Sandrine Dudoit. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC Genomics*, 19(1):477, June 2018.

[21] Conrad Hal Waddington. *The strategy of the genes : a discussion of some aspects of theoretical biology*. Routledge Library Editions: 20th Century Science. Routledge, 1957.

[22] F. Alexander Wolf, Fiona K. Hamey, Mireya Plass, Jordi Solana, Joakim S. Dahlin, Berthold Göttgens, Nikolaus Rajewsky, Lukas Simon, and Fabian J. Theis. PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome Biology*, 20(1):59, Mar. 2019.

[23] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8(1), Apr. 2017.