Accelerating hyperbolic t-SNE using the Lorentz Hyperboloid
Exploring a different way to speed up hyperbolic t-SNE

Daniel Peter[1]

Supervisor(s): Elmar Eisemann[1]**, Martin Skrodzki**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty, Delft University of Technology,
In Partial Fulfillment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Daniel Peter
Final project course: CSE3000 Research Project
Thesis committee: Elmar Eisemann, Martin Skrodzki, David Tax

Abstract

This paper investigates a method for accelerating hyperbolic t-SNE — a popular high-dimensional data visualization technique. In particular, it focuses on building a hyperbolic t-SNE variant that uses a different model of hyperbolic space (called the Lorentz Hyperboloid model) for representing the low-dimensional embeddings. An acceleration algorithm based on a tree data-structure is then used to achieve a better asymptotic runtime complexity compared to the original version. The paper then compares this implementation with other alternatives — including accelerated variants — and shows that it computes embeddings of better quality at a similar rate.

Keywords: visualization, dimensionality reduction, acceleration structure, hyperbolic embedding, hyperboloid model, Lorentz model

1 Introduction

In order for humans to be able to interpret and analyse data that has more than two or three dimensions, it has to be embedded on a lower-dimensional plane. This would allow developments of hypotheses about the underlying structure of the process that generated this data. Because of this, visualization and analysis techniques for high-dimensional data play a crucial role in a wide range of applications.

Dimensionality reduction is the process through which high-dimensional data is embedded into a low-dimensional plane. In order to illustrate the relationships between the embedded points, the process usually tries to preserve different metrics that hold significance both in the high and low-dimensional spaces. Because of the nature of high-dimensional data, the proximity relationships between points within one embedding are most reliable. This means that close data points (in the high-dimensional plane) will be embedded close to each other. Examples of proximity preserving metrics include the pairwise scalar product [14], pairwise distances [9] or rank information [26]; a review of these methods can be found in [5].

One widespread technique to embed high-dimensional data into a low-dimensional space is t-distributed Stochastic Neighbour Embedding (t-SNE) [35]. This method works by minimizing the Kullback-Leibler divergence between two distributions, thus preserving proximity of the embeddings. However, there is data that exhibits intrinsic hyperbolic-like features (e.g., exponential growth), such as graphs, trees, or other hierarchical data. Hyperbolic space is the complete, simply connected Riemannian manifold with constant negative sectional curvature (described in Subsection 2.4). Applications where data can be modelled in this form, e.g., social networks [36], the Internet [4], or gene expression [40], would benefit from extending visualizing techniques to the hyperbolic spaces. Given the usefulness of analysing the structures of these datasets, various methods have been proposed for extending t-SNE for hyperbolic spaces [12; 15].

When it comes to representing these hyperbolic spaces in 2 dimensions, there are four popular models: Poincaré disk and half-plane models, Klein model and, lastly, the Lorentz Hyperboloid model. In this paper, we focus on the latter. The main advantage of this model is that its distance function avoids numerical instabilities that arise from the fraction in the Poincaré distance, which improves the efficiency in learning embeddings. For this reason, it is frequently used when searching for hierarchical relationships in data [25]. Similar studies suggested this model for use in conjunction with hyperbolic t-SNE (Ht-SNE) [15; 31].

The computation of t-SNE gradients involves considering positive and negative forces between all pairs of points. Thus, the computational complexity of this process scales quadratically in the number of data points. In order to visualize larger data sets, a data-structure has been proposed for t-SNE that makes use of an acceleration algorithm, called Barnes-Hut (that we will discuss in Subsection 2.3) [34], in order to reduce its run-time complexity. For hyperbolic spaces, due to their negative curvature, accelerating using this data-structure does not work without modifications [17]. This is due to the absence of linear interpolation or averages in non-linear and hyperbolic spaces. To solve this, a method that uses a Poincaré disk and modifies the Barnes-Hut scheme to work in hyperbolic spaces has been proposed and bench-marked [31]. It makes use of a polar quad-tree [37] as the acceleration data-structure.

This paper aims to provide an answer to the question regarding whether the Lorentz Hyperboloid model can be used as the embedding space, in conjunction with an octree acceleration data-structure, to improve the quality of the embeddings of Ht-SNE, while maintaining a similar run-time efficiency. By leveraging the advantages of the Lorentz model in this way, we can obtain a better representation of high-dimensional data, that would improve the preservation of the relationships between the points and make the development of hypotheses easier. Moreover, improving computational efficiency would make Ht-SNE more viable for use in large data-sets and for various other data visualization tasks.

To this end, this paper introduces this aforementioned Ht-SNE variant that uses the Lorentz Hyperboloid model as the embedding space. Then, it describes an acceleration data-structure based on the Barnes-Hut algorithm that can be used to speed up the computation. Because the Lorentz model has one extra dimension, an octree is used (described in Subsection 2.2). This data-structure then needs to be adjusted in order to reflect hyperbolic properties (Subsection 4.1). Lastly, in Section 5 we compare this approach with other existing methods, such as non-accelerated (exact) Ht-SNE, or the quadtree-accelerated Poincaré Ht-SNE [31] and show that it computes embeddings of better quality, at a similar rate.

2 Background

This section explains some of the concepts that the acceleration algorithm is built upon. It introduces t-SNE and the octree acceleration data-structure. Then, it explains the Euclidean way to accelerate it, using a quad-tree as part of the Barnes-Hut algorithm. Finally, it presents some concepts

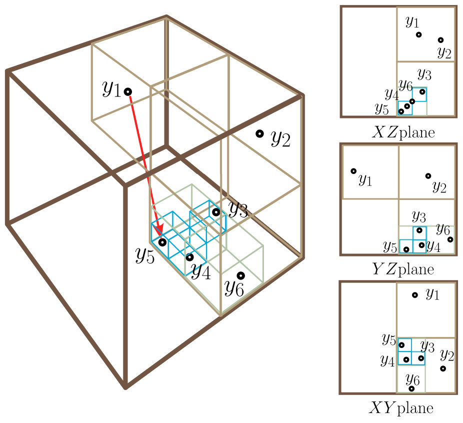

Figure 1: Representation of an octree in 3D and projected onto each plane. Only the cells which contain at least one point are displayed.

about hyperbolic spaces, and, in particular, about the hyperboloid model, that are being employed in the algorithm.

2.1 t-distributed Stochastic Neighbour Embedding

t-SNE is a visualization technique for high-dimensional data. It aims to minimize the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding space [35]. For the d-dimensional input object $\{x_1, \ldots, x_N\} \subseteq \mathbb{R}^d$, t-SNE defines joint probabilities

$$p_{j|i} = \frac{exp(-\|\mathrm{x}_i - \mathrm{x}_j\|^2/2\sigma_i)}{\sum_{k\neq i} exp(-\|\mathrm{x}_i - \mathrm{x}_k\|^2/2\sigma_i^2)}, \quad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}. \tag{1}$$

Here, $p_{i|i} = 0$ and the bandwidth of the Gaussian kernels, σ_i, is set such that the perplexity value of the conditional probability P_i equals a predefined perplexity u. The perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)}, \tag{2}$$

where $H(P_i)$ is the Shannon entropy of P_i, measured in bits. The corresponding probability distribution for the low-dimensional embedding $\{y_1, \ldots, y_N\} \subseteq \mathbb{R}^s$ is

$$q_{ij} = \frac{(1 + \|\mathrm{y}_i - \mathrm{y}_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|\mathrm{y}_k - \mathrm{y}_l\|^2)^{-1}}. \tag{3}$$

To compute the positions y_i of the low-dimensional embedding, t-SNE starts with an initial embedding obtained by principal component analysis (PCA), as suggested in previous papers [16]. Then, the locations of the embedding points y_i are determined by minimizing the Kullback-Leibler divergence between the joint distributions P and Q:

$$C = \mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4}$$

which has the gradient

$$\frac{\delta C}{\delta \mathrm{y}_i} = 4\left(\sum_{j\neq i} p_{ij} q_{ij} Z(\mathrm{y}_i - \mathrm{y}_j) - \sum_{j\neq i} q_{ij}^2 Z(\mathrm{y}_i - \mathrm{y}_j)\right), \tag{5}$$

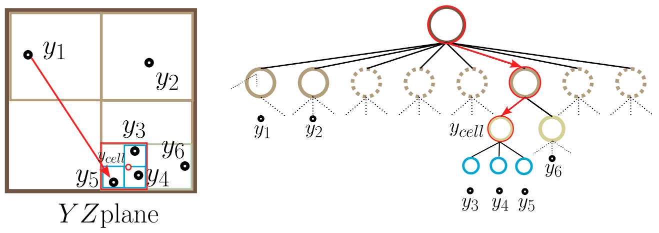

Figure 2: Barnes-Hut scheme applied on octree in Figure 1. When calculating the distance from y_1 to any of y_3, y_4 or y_5, the midpoint y_{cell} is used, assuming that Equation 6 holds.

where $Z = \sum_{k\neq l}\left(1 + \|y_k - y_l\|^2\right)^{-1}$. If the probability distribution P is sparse, we can efficiently compute the first term of the equation. However, for the second term, because we need to compute all pairwise positive and negative forces between the points, the runtime complexity of a naive implementation of this algorithm would be $\mathcal{O}(n^2)$.

2.2 Octree

An octree [24] is a tree in which each node represents a cube shaped cell with a particular centre, width, height, and depth. Non-leaf nodes have eight children that split up the cell into eight smaller, equally sized cells (octants), as shown in Figure 1. Leaf nodes represent cells that contain at most one point of the embedding; the root node represents the cell that contains the complete embedding. Each node stores the centre-of-mass of the embedding points that are located inside the corresponding cell, $\mathrm{y}_{\mathrm{cell}}$, and the total number of points that lie inside the cell, N_{cell}.

An octree has $\mathcal{O}(n \log n)$ nodes and can be constructed generally in $\mathcal{O}(n \log n)$ time by inserting the points one-by-one, splitting a leaf node whenever a second point is inserted in its cell, and updating $\mathrm{y}_{\mathrm{cell}}$ and N_{cell} of all visited nodes. However, in the unlikely worst case scenario, the octree can be built in $\mathcal{O}(n^2)$ time.

2.3 Barnes-Hut Acceleration Structure for t-SNE

Previous studies have shown that Euclidean t-SNE can be accelerated to run in $\mathcal{O}(n \log n)$ time (when choosing proper hyperparameters) by using a data-structure used by the Barnes-Hut algorithm [34]. This algorithm aims to approximate the n-body computation by making use of a *quadtree* for two dimensions, or an *octree* for three dimensions [2].

When applied to t-SNE, the embedding space is divided up into square cells via a quadtree, so that only points from nearby cells need to be treated individually, and points in distant cells can be treated as a single large point centred at the cell's centre of mass. This allows for approximating the second term in Equation 5: when evaluating the gradient for an embedding point y_i, the quadtree structure is traversed. For each cell, starting at the root, we check if

$$\frac{r_{\mathrm{cell}}}{\|\mathrm{y}_i - \mathrm{y}_{\mathrm{cell}}\|} < \theta \tag{6}$$

holds, where r_{cell} is the length of the diagonal of the cell, $\mathrm{y}_{\mathrm{cell}}$ is the arithmetic midpoint of all points stored in the cell,

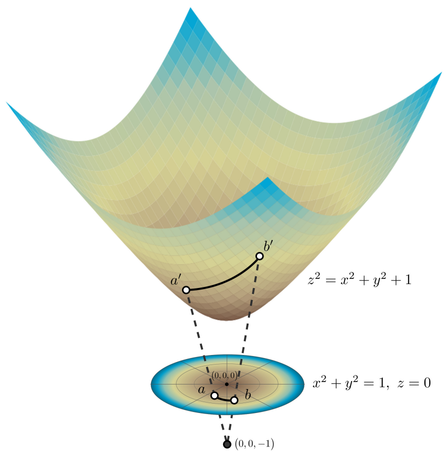

Figure 3: Relationship between the Poincaré and Lorentz models; changing between the models involves projecting the points through a point situated at $(0, 0, -1)$

and θ is a hyperparameter that influences the trade-off between efficiency and quality of the embedding [34]. The cell and its children are summarized to this arithmetic mean y_{cell}, weighted by the number of embedding points inside the cell, if Equation 6 holds.

Figure 2 contains an illustration of this procedure on a 3-dimensional octree. This procedure is analogous for a 2-dimensional quadtree.

2.4 Hyperbolic Spaces and the Hyperboloid Model

Hyperbolic space of dimension 2 is the unique, complete, simply connected, 2-dimensional Riemannian manifold with constant negative sectional curvature, equal to -1. There exist multiple equivalent models for hyperbolic space, such as the Poincaré ball and half-plane model, the Klein model and the Lorentz Hyperboloid model [1]. Equivalence in this case means that all of these models are compatible with each other and conversion between them is not computationally expensive. Figure 3 contains an illustration that shows the transformation from a set of Poincaré points to Lorentz coordinates.

Other studies based their approach for learning and visualizing embeddings on the Poincaré disk model, due to its conformality and convenient parameterization [25; 31]. In this paper, we use the Lorentz model for representing the 2-dimensional embedding space, before applying the t-SNE algorithm. The main advantage of this model is that its distance function (see Equation 9) avoids numerical instabilities that arise from the fraction in the Poincaré distance.

In the following, let any vector $u \in \mathbb{R}^{n+1}$ be written as

$$u = (u_t, u'), \text{ where } u' \in \mathbb{R}^n,\ u_t \in \mathbb{R}.$$

The Lorentz model of 2-dimensional hyperbolic space is defined as the Riemannian manifold $\mathcal{L}^2 = (\mathcal{H}^2, g_{\mathrm{y}}^{\mathcal{L}})$, where $g_{\mathrm{y}}^{\mathcal{L}}$ is the Riemannian metric tensor $\mathrm{diag}(-1, 1, 1)$ and

$$\mathcal{H}^2 = \{\mathrm{y} = (\mathrm{y}_t, \mathrm{y}') \in \mathbb{R}^3 : \langle \mathrm{y}, \mathrm{y} \rangle_{\mathcal{L}} = -1, \mathrm{y}_t > 0\}, \quad (7)$$

is the upper sheet of a two-sheeted 2-dimensional hyperboloid (see top surface in Figure 3). Above,

$$\langle u, v \rangle_{\mathcal{L}} = -u_t v_t + \langle u', v' \rangle \quad (8)$$

denotes the *Lorentzian scalar product*, where $\langle \cdot, \cdot \rangle$ is the standard inner product between two vectors in $\mathbb{R}^n$. The distance for the Lorentz model between two points y_i and y_j is

$$d_{ij}^{\mathcal{L}} = \mathrm{arcosh}(-\langle \mathrm{y}_i, \mathrm{y}_j \rangle_{\mathcal{L}}). \quad (9)$$

3 Related Work

There is an already existing body of work that focused on decreasing the asymptotic runtime complexity of algorithms that ran in quadratic time, such as t-SNE. This paper builds on top and expands it by studying the effects using the Lorentz Hyperboloid model as the embedding space for t-SNE. This model has seen usage in various fields of science, and we will give a brief overview of them in Subsection 3.1. We will also explore another technique that can be employed to speed up t-SNE in Subsection 3.2.

3.1 Other Applications for the Lorentz model

Recently, hyperbolic geometry has been the subject of many studies in the field of data visualization, due to its ability to represent data with a non-Euclidean nature. Data that exhibits hyperbolic properties especially benefit from using hyperbolic space for embedding, having a better representation capacity and generalization ability [4; 17; 15; 8; 36].

Particularly, the Lorentz Hyperboloid model (discussed in Subsection 2.4) was used to define the basic operations of hyperbolic neural networks [25]. Additionally, there have been developments of various neural network architectures using the Lorentz model. Take, for example, hyperbolic graph neural networks [39], that use the same model for a similar reason as us, namely better numerical stability. Another example are the fully hyperbolic neural networks that ”are based on the Lorentz model by adapting the Lorentz transformations to formalize essential operations of neural networks” [6].

The hyperboloid model has seen great use in the field of physics, in particular for special and general relativity. Minkowski showed that this model can be used to perform geometry on his proposed theory about spacetime in the absence of gravitation, which has inherent hyperbolic properties [7; 38]. More recent advancements in this field include the study of the geometry of the asymptotic infinities of Minkowski space-time [10]. A comprehensive study of Minkowski's proposal, its applications can be found in [23; 22].

3.2 t-SNE Acceleration Techniques

In [34], two tree-based acceleration algorithms were compared for the Euclidean case of t-SNE; these are Barnes-Hut and the Dual-Tree algorithm. The Barnes-Hut scheme is already explained in Subsection 2.3. The dual-tree approximation [11] works by trying to decide, for every pair of nodes, whether the interaction between the cells of two identical quad-trees (A and B, built on the data points) can be used as "summary" for the interactions between all points inside these two cells. If the condition

$$\frac{max(r_{\text{cell-}A}, r_{\text{cell-}B})}{\|\mathrm{y}_{\text{cell-}A} - \mathrm{y}_{\text{cell-}B}\|^2} < \theta$$

holds, the corresponding force between the two cell midpoints is computed. Then, the computed force gets added to all points from each cell, together with the number of points in the other cell. Similarly with Barnes-Hut, this approximation is also used to compute the repulsive forces of Equation 19. However, as found by [34], using this acceleration scheme is slightly slower than using the Barnes-Hut algorithm. This is because, even though the Dual-Tree approximation minimizes the number of distances having to be computed between points, we would still need to know on what points this approximation applies on. Therefore, the time complexity stays the same as Barnes-Hut. This motivates our focus on the latter.

4 Octree Accelerated Hyperbolic t-SNE

Accelerating Ht-SNE on the Lorentz Hyperboloid implies the need for a suitable acceleration structure, that takes into account the specific properties of the space it operates on. We describe our proposed solution in Subsection 4.1. Then, we describe the steps involved in computing the hyperbolic gradient on the hyperboloid, and show how the data structure can be used to approximate it in Subsection 4.2.

4.1 Octree Over the Lorentz Model

We aim to use the Lorentz Hyperboloid model for embedding high-dimensional data using t-SNE and gain the advantages that this model provides. In order to speed up this algorithm, this paper sets to find an acceleration data-structure that matches the properties of this space. Recall that the ambient space of a 2-dimensional hyperboloid is $\mathbb{R}^3$. To this end, the algorithm presented in this paper makes use of the 3-dimensional extension of the quadtree: the octree.

Splitting Criterion

For the regular Barnes-Hut approach using a quadtree in Euclidean space, the splitting point is chosen as the Euclidean centre of the parent. For 3-dimensional space, the procedure is the same. This rule splits each cell into eight cubes of equal Euclidean volume. Although this criterion works best with Euclidean data, we have decided to use it for speeding up Ht-SNE because of its simplicity and ease of computation. However, because the hyperboloid is a thin surface, this has the consequence that most of the volume of each cube is not occupied by any data points. This, in turn, leads to redundant splitting and increases the maximum depth of the tree, which poses overhead when summarizing the tree.

Midpoint Computation

In order to be able to perform the Barnes-Hut approximation from Equation 6, each inner cell of the octree has to include a midpoint of all the points included in its children. If every leaf contains one point y_i, call the midpoint of each cell y_{cell}. In Euclidean space, this is simply the weighted arithmetic mean

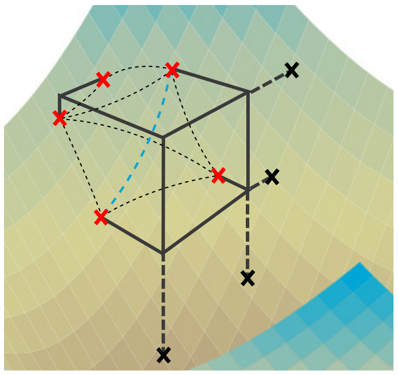

Figure 4: Cell intersection with hyperboloid. Points marked with an **X** are intersection points of the lines defined by the edges of the cube. The ones in black reside outside the edge boundaries, while the ones with red are inside. The dotted lines represent pairwise distances between intersection points. The blue dotted line is the longest of them.

of the stored embedding points, which is not available in hyperbolic space. For the Lorentz Hyperboloid model, there exists an analogous method, called the *Lorentz centroid* [19], with formula

$$c(\mathrm{y}_j) = \frac{\frac{1}{N}\sum_j^N \mathrm{y}_j}{\|\frac{1}{N}\sum_j^N \mathrm{y}_j\|_{\mathcal{L}}} \tag{10}$$

where $\|\mathrm{y}\|_{\mathcal{L}} = \sqrt{|\langle \mathrm{y}, \mathrm{y}\rangle_{\mathcal{L}}|}$. An important property of this centroid is that it can be written as a rolling average, taking $\mathcal{O}(1)$ time to compute.

In a similar study, the midpoint is given by the pseudo-Fréchet mean, that uses Klein coordinates [31]. The study claims an error rate of less than 7% with regard to the Fréchet variance problem, according to [41]. In our research, we found that using the Lorentz centroid was slightly faster and more precise, due to not having to compute the Klein coordinates for each point, thus avoiding the floating point division (see Figure 14 in the appendix).

Maximum Cell Width Computation

All cells in the tree should include r_{cell}, the maximum distance in hyperbolic space that is covered by the cell. In Euclidean space, this is the long diagonal of the cube that contains all the points of the leaves. However, when using the octree on the hyperboloid, this metric is no longer applicable. Because of the nature of hyperbolic spaces, given a set of points on the manifold, inside a bounding cube, we can only find out the maximal distance by computing all pairwise hyperbolic distances and choosing the maximum. Unfortunately, this is quadratic time. Alternatively, one could project the vertices of the bounding cube on the hyperboloid and compute these pairwise distances. However, projecting from Euclidean space onto the hyperboloid is not trivial and computationally expensive. Therefore, we can opt for an approximation [30].

Given a bounding box $\mathcal{B}$ for a set of points on the hyperboloid, we compute the maximum distance as follows:

1. Given the endpoints A_i, B_i of each edge e_i of $\mathcal{B}$, with coordinates $(A_{ix}, A_{iy}, A_{iz}), (B_{ix}, B_{iy}, B_{iz})$, compute

the parametric line equation that passes through those points, namely

$$l(\omega \in \mathbb{R}) = A_i + \omega(B_i - A_i). \tag{11}$$

2. Compute the intersection point between l and the hyperboloid $\mathcal{H}$, by plugging each component from Equation 11 into the hyperboloid formula from Equation 7:

$$l_z(\omega')^2 = l_x(\omega')^2 + l_y(\omega')^2 + 1, \tag{12}$$

where ω' is the parameter of the intersection point $l(\omega')$. This is a simple quadratic equation that can be solved in constant time. However, we are only interested in the points that reside on the boxes' edges, so we restrict $0 \leq \omega' \leq 1$.

3. Compute the pairwise hyperbolic distances $d^{\mathcal{H}}$ between these points and select the maximum one.

A representation of this process can be seen in Figure 4. It is worth pointing out that this procedure can be implemented to have a constant runtime complexity, due to the fact that there can be at most 2 intersection points per edge, so 24 intersection points in total. However, in practice, there are always less than 24 intersection points.

Because this is just an approximation, there might be two points in the cube that are further away than the maximum distance computed as described above. This is because the contour formed by connecting all the intersection points between the edges and the hyperboloid might not be the same as the contour formed by intersecting it with the faces of the cube. Therefore, a point on one face of the cube (and not on the edge) might be further away from another point than the closest intersection points. This inaccuracy might lead to points being summarized when they shouldn't, according to the condition in Equation 6, which would propagate as a slight precision drop in the outcome of the embedding. However, calculating the exact performance difference falls outside the scope of this paper.

4.2 Gradient Descent on the Hyperboloid

When it comes to computing the gradient descent needed for minimizing the Kullback-Leibler divergence on the hyperboloid, the standard t-SNE algorithm has to be modified to reflect the properties of hyperbolic space. This paper proposes an algorithm that best resembles the Euclidean case. We keep the probabilities in the high-dimensional plane the same as in Equation 1, but we modify the embeddings probabilities to include the distance metric $d_{ij}^{\mathcal{L}}$ from Equation 9, specific to the Lorentz Hyperboloid model. The new probabilities would then be described by the following equation:

$$q_{ij}^{\mathcal{L}} = \frac{\left(1 + (d_{ij}^{\mathcal{L}})^2\right)^{-1}}{\sum_{k \neq l} \left(1 + (d_{ij}^{\mathcal{L}})^2\right)^{-1}}. \tag{13}$$

This method is analogous to the one used in [31], where, instead of the Lorentz Hyperboloid model, the Poincaré model was used. The process of gradient descent on the manifold is also discussed in [25; 6].

In the following, we will abuse the notation and represent the variance of a function f with respect to some vector v in a

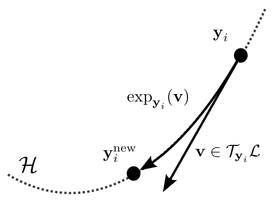
y_i
$\exp_{\mathbf{y}_i}(\mathbf{v})$
$\mathrm{y}_i^{\mathrm{new}}$
$\mathbf{v} \in \mathcal{T}_{\mathbf{y}_i}\mathcal{L}$
$\mathcal{H}$

Figure 5: Vector **v** that resides in the tangent space $\mathcal{T}_{\mathbf{y}_i}\mathcal{L}$ being projected back on the hyperboloid. The gray, dotted line is a cross-section of the hyperboloid $\mathcal{H}$.

space $\mathcal{S}$ as $\nabla_{\mathrm{v}}^{\mathcal{S}} f(\mathrm{v})$. The gradient of the cost function C from Equation 4, accounting for hyperbolic distance and taken in the ambient space $\mathbb{R}^3$, is then

$$\nabla_{\mathrm{y}_i}^{\mathbb{R}^3} C^{\mathcal{L}} = 4 \sum_{j \neq i} \left(p_{ij} - q_{ij}^{\mathcal{L}}\right) \left(1 + (d_{ij}^{\mathcal{L}})^2\right)^{-1} \nabla_{\mathrm{y}_i}^{\mathbb{R}^3} d_{ij}^{\mathcal{L}}. \tag{14}$$

Here, the variation of the hyperbolic distance $\nabla_{\mathrm{y}_i}^{\mathbb{R}^3} d_{ij}^{\mathcal{L}}$ (taken in the ambient space) is

$$\nabla_{\mathrm{y}_i}^{\mathbb{R}^3} d_{ij}^{\mathcal{L}} = \frac{1}{\sqrt{\langle \mathrm{y}_i, \mathrm{y}_j \rangle_{\mathcal{L}}^2 - 1}} \mathrm{y}_j. \tag{15}$$

Because the gradient from Equation 14 resides in the ambient space $\mathbb{R}^3$, it has to be projected first to the tangent space $\mathcal{T}_{\mathrm{y}_i}\mathcal{L}$ of the hyperboloid $\mathcal{L}$ at point y_i. We obtain this by using the following formula:

$$\nabla_{\mathrm{y}_i}^{\mathcal{L}} C^{\mathcal{L}} = \nabla_{\mathrm{y}_i}^{\mathbb{R}^3} C^{\mathcal{L}} + \left\langle \mathrm{y}_i, \nabla_{\mathrm{y}_i}^{\mathbb{R}^3} C^{\mathcal{L}} \right\rangle_{\mathcal{L}} \cdot \mathrm{y}_i. \tag{16}$$

In order for each gradient descent step taken from y_i in the hyperbolic tangent space $\mathcal{T}_{\mathrm{y}_i}\mathcal{L}$ to be projected to the correct point on the hyperboloid, a standard procedure is to utilize an exponential map (see Figure 5). That is, for a vector $v \in \mathcal{T}_{\mathrm{y}_i}\mathcal{L}$, we project the corresponding point taken from a point y_i back on the manifold to

$$\exp_{\mathrm{y}_i}(\mathrm{v}) = \cosh(\|\mathrm{v}\|_{\mathcal{L}})\mathrm{y}_i + \sinh(\|\mathrm{v}\|_{\mathcal{L}}) \frac{\mathrm{v}}{\|\mathrm{v}\|_{\mathcal{L}}}, \tag{17}$$

[25]. It is important to remark that this exponentiation step takes into account the starting point y_i, so it moves the gradient to that point. Therefore, we can express the update equation of each point after a gradient descent step as

$$\mathrm{y}_i^{\mathrm{new}} = \exp_{\mathrm{y}_i}(-\alpha \cdot \nabla_{\mathrm{y}_i}^{\mathcal{L}} C^{\mathcal{L}}), \tag{18}$$

where α is the learning rate hyperparameter.

An important remark is that the gradient in Equation 14 can be rewritten in a split form of two sums, similar to Equation 5:

$$\nabla_{\mathrm{y}_i}^{\mathbb{R}^3} C^{\mathcal{L}} = 4 \left(\sum_{j \neq i} p_{ij} q_{ij}^{\mathcal{L}} Z^{\mathcal{L}} \nabla_{\mathrm{y}_i}^{\mathbb{R}^3} d_{ij}^{\mathcal{L}} - \sum_{j \neq i} \left(q_{ij}^{\mathcal{L}}\right)^2 Z^{\mathcal{L}} \nabla_{\mathrm{y}_i}^{\mathbb{R}^3} d_{ij}^{\mathcal{L}} \right), \tag{19}$$

where $Z^{\mathcal{L}} = \sum_{k \neq l} \left(1 + (d_{ij}^{\mathcal{L}})^2\right)^{-1}$. Given this, we can use the modified octree as the main acceleration tool to speed up the evaluation of the hyperbolic gradient. The first term of Equation 19, just as in the Euclidean space, given a sparse high-dimensional probability distribution P with truncated Gaussians, can be computed without negatively affecting the algorithm's performance. The second term of the equation can be sped up by using an analogous method to Barnes-Hut [31]. That is, assuming that a cell of the octree is sufficiently small and sufficiently far away from a query point y_i, the contributions $-(q^{\mathcal{L}}_{ij})^2\, Z^{\mathcal{L}}\, \nabla_{y_i} d^{\mathcal{L}}_{ij}$ will be similar for all points y_j in this cell. Therefore, we can replace these summands by

$$-N_{\text{cell}}\, (q^{\mathcal{L}}_{ij})^2\, Z^{\mathcal{L}}\, \nabla_{\mathrm{y}_i} d^{\mathcal{L}}(\mathrm{y}_i, \mathrm{y}_{\text{cell}}), \tag{20}$$

where N_{cell} is the number of points in the cell, y_{cell} is the midpoint of the cell, and

$$q^{\mathcal{L}}_{ij}\, Z^{\mathcal{L}} = \left(1 + \left(d^{\mathcal{L}}(\mathrm{y}_i, \mathrm{y}_{\text{cell}})\right)^2\right)^{-1}.$$

The process of computing the hyperbolic gradient in an accelerated manner is showcased in Algorithm 1.

5 Experimental Setup and Results

Several experiments on various data-sets of different sizes were performed, in order to evaluate the performance of the proposed accelerated Lorentz Ht-SNE algorithm. This section firstly describes the used data-sets and the experimental setup, then it presents the two categories of experiments performed: the quality of the embeddings is assessed in Subsection 5.1 and the runtime efficiency of the algorithm is analysed in Figure 5.2.

Data Sets

The experiments were performed on 4 data-sets: the Planaria data-set [29], C. Elegans [13], MyeloidProgenitors data-set [18] and the MNIST data-set [20]. The first three contain data from single-cell RNA sequencing. The MNIST data-set is a popular choice for testing dimensionality reduction algorithms. These data-sets were chosen for their varying characteristics, such as number of points, number of dimensions and data origin, as well as for being used by related hyperbolic t-SNE implementations [12; 15; 40; 31]. A summary of these data-sets and their properties can be found in Table 1.

Algorithm 1 Summary of a Gradient Descent Step at y_i

```
Require: $Y \subseteq \mathcal{H}$, $\mathrm{y}_i \in Y$
  $f_{pos} \leftarrow \sum_{j \neq i} p_{ij}\, q^{\mathcal{L}}_{ij}\, Z^{\mathcal{L}}\, \nabla_{y_i} d^{\mathcal{L}}_{ij}$
  $f_{neg} \leftarrow 0$
  $octree \leftarrow$ build octree on $Y$
  for all $\mathrm{y}_j \in Y$ do
      ▷ Perform depth-first traversal of the octree                    ◁
      cell ← root of $octree$                         ▷ Start with root cell
      while $r_{\text{cell}}/d^{\mathcal{L}}(\mathrm{y}_i, mid_{\text{cell}}) \geq \theta$ do          ▷ Equation 6
          cell ← child of $tree$ that contains $\mathrm{y}_j$
      ▷ From Equation 20,                                              ◁
      $f_{neg} \leftarrow f_{neg} - N_{\text{cell}}\, (q^{\mathcal{L}}_{ij})^2\, Z^{\mathcal{L}}\, \nabla_{\mathrm{y}_i} d^{\mathcal{L}}(\mathrm{y}_i, mid_{\text{cell}})$
  $\nabla^{\mathbb{R}^3}_{\mathrm{y}_i} C^{\mathcal{L}} \leftarrow f_{pos} + f_{neg}$                       ▷ Equation 19
  $\nabla^{\mathcal{L}}_{\mathrm{y}_i} C^{\mathcal{L}} \leftarrow \nabla^{\mathbb{R}^3}_{\mathrm{y}_i} C^{\mathcal{L}} + \langle \mathrm{y}_i, \nabla^{\mathbb{R}^3}_{\mathrm{y}_i} C^{\mathcal{L}} \rangle_{\mathcal{L}} \cdot \mathrm{y}_i$      ▷ Equation 16
  $\mathrm{y}_i \leftarrow \exp_{\mathrm{y}_i}(-\alpha \cdot \nabla^{\mathcal{L}}_{\mathrm{y}_i} C^{\mathcal{L}})$                   ▷ Equation 18
```

Name	Data Type	# Points	# Dim.	# Cl.
PLANARIA	single-cell	21,612	50	51
C_ELEGANS	single-cell	89,701	20,222	37
MYELOID	single-cell	8,000	11	5
MNIST	images	70,000	784	10

Table 1: Data sets used for experiments in Section 5. For each data set, the size, dimension of data points and the number of classes are given.

Experimental Setup

For the following experiments, this paper follows the experimental setup from [31] as closely as possible. We firstly use principal component analysis (PCA) to reduce the data to 50 dimensions to speed up computations. Afterwards, a stage of early exaggeration (EE) is applied, as part of the standard t-SNE strategy. EE is a series of gradient descent steps for which the attractive forces p_{ij} are amplified by multiplying them with a factor, in this case 12 [3]. After this stage, several non-exaggerated gradient descent steps are employed.

We use different learning rates for all the different instances of the algorithm that we compare against. For the Euclidean version, we use a learning rate $\alpha = n/12$ [3], where n is the number of points in the dataset. For the Poincaré Ht-SNE, we use the proposed $\alpha = n/12000$ as discussed in [31]. For our proposed method, we opted for an intermediate value for α (see discussion in last experiment of Subsection 5.1). However, by using the same learning rate for both the EE phase and for the normal phase, there is no suitable value for α that keeps the EE from spreading too far, while making the normal phase distribute the points evenly. Thus, we have different values α_{EE}, α_N for the EE and normal phases, respectively. Thus, for the Lorentz Ht-SNE, we chose

$$\alpha_N = \frac{n}{12 \cdot 5}, \; \alpha_{EE} = \frac{\alpha_N}{12} = \frac{n}{12 \cdot 60}.$$

Together with this learning rate, we follow the study in [31] by using momentum and gains for performing the gradient descent on the hyperboloid [35]. By using this method, we ensure a small learning rate in the beginning, that builds up over time. If we visualize data on the Poincaré disk, this varying learning rate helps the data uniformly distribute over the area of the unit circle. We keep the recommended values for these parameters for the Poincaré implementation, namely 0.5 during EE and 0.8 for the normal phase [31]. For our implementation, these values are unsuitable, as the points would be quickly pushed towards infinity. That's why, for the Lorentz Ht-SNE, we use a value of 0.35 for the momentum during EE, and 0.6 in the normal phase.

Moreover, we keep the θ value, as recommended in previous studies, with a value of 0.5 [35] and we run all experiments with a uniform perplexity value of 30 [16].

Lastly, unless explicitly mentioned, when we mention a 'run' of the algorithm, we refer to running the algorithm with 250 iterations of EE, followed by 750 iterations of normal gradient descent. For the last 750 iterations, we use an early stopping criterion.

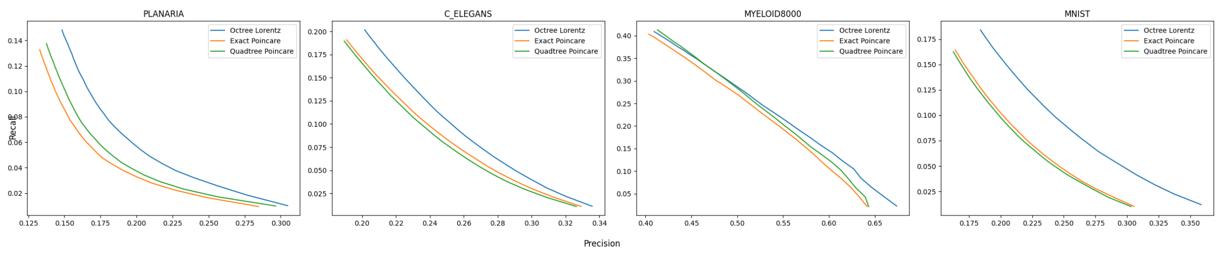

Figure 6: Precision vs. recall graphs for C_ELEGANS (far left), PLANARIA (centre left), MyeloidProgenitors (centre right) and MNIST (far-right) data-sets

5.1 Embedding Quality Assessment

In this first set of experiments, we aim to evaluate the quality of the resulting embeddings, by comparing our implementation with the Poincaré [31] and the exact implementations. We first investigate the effect of changing the model, by comparing the exact versions and the accelerated versions against each other. Then, we check what effect does changing the hyperparameter θ have on the quality of the embeddings. The last experiment investigates the result of varying the learning rate for our proposed implementation.

Effect of Acceleration on Embedding Quality

In order to assess the quality of the accelerated version of Ht-SNE, we use the precision / recall metric [28]. According to previous studies, we set a maximum neighbourhood size $k_{max} = 30$. Then, ”*for each $k \in \{1, \dots, k_{max}\}$, we compute the number of true positives as $TP_k = N_{k_{max}}(X) \cap N_K(Y)$, that is, the points that are in the high-dimensional neighbourhood and also in the low-dimensional embedded neighbourhood, given the respective metrics. From this value, we obtain the precision as $PR_k = |TP_k|/k$ and the recall as $RC_k = |TP_k|/k_{max}$. Therefore, ideally, the precision is always 1, while the recall grows as k/k_{max}, yet, a data set might not exhibit such a solution, nor does t-SNE necessarily find this solution*” [31]. Instead, we aim to show that our proposed implementation increases this quality, while maintaining the speed-up offered by the acceleration.

As can be seen in the precision/recall curves in Figure 6, for all the data sets that were tested, our proposed solution outperforms not only the accelerated Poincaré version, but the exact one as well. This might happen due to the numerical stability of the Lorentz model, especially since the regular, exact computation, still struggles with precision issues, when representing points very close to the disk boundary.

Effect of Theta on Embedding Quality

The parameter θ, used in Equation 6, dictates whether a subtree of the hierarchy is explored or approximated. This is the main hyperparameter of our algorithm. By setting $\theta = 0$, the algorithm never approximates, and it is equivalent to the exact version. This test was performed by choosing a range of $\theta \in \{0.0, 0.1, \dots, 1.0\}$, and running the algorithm with this value for all data-sets in Table 1. For each of these runs, we compute the precision / recall curves, similar to the previous experiment.

These curves can be seen in Figure 7 for the MNIST data-set. From this figure, we can see that most curves are clustered together, meaning that varying the value of θ does not significantly influence the embedding quality. This suggests that the performance benefit from increasing θ (discussed in Figure 5.2) greatly outweighs the drawbacks from decreased quality.

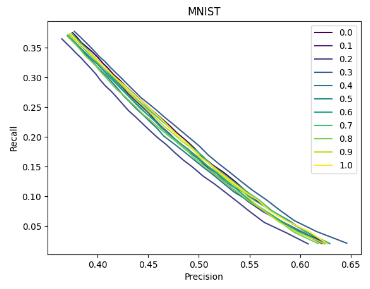

Figure 7: Precision / recall curves for different values of θ

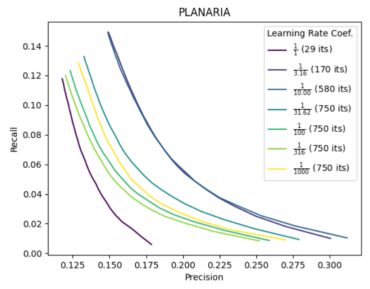

Figure 8: Precision vs. recall graph for the PLANARIA data-set with various values of ν.

Effect of Learning Rate on Embedding Quality

In order to find the optimal learning rate, we perform a grid search. For all $\nu \in \{10^0, 10^{0.5}, 10^1, \cdots, 10^3\}$, we investigate $\alpha = \nu \cdot \frac{n}{12}$. We choose the base learning rate $\frac{n}{12}$, according to the original t-SNE study [34], and the maximum bound the one proposed in the Poincaré implementation paper [31]. The reason behind this choice is simple: in the Lorentz model of hyperbolic space, we work with numbers outside the range $(-1, 1)$ (as opposed to the Poincaré model), but the points still grow quickly to infinity the further from the origin we go (unlike Euclidean space). Therefore, the search for our value has to lie somewhere between these two extremes.

We analysed the precision vs. recall curves, as described earlier, after running the algorithm for all learning rates. The curves can be seen in Figure 8 for the PLANARIA data-set. The graphs for the rest of the datasets can be found in the appendix. We can see that choosing a higher learning rate decreases the number of iterations needed for convergence. For lower learning rates, this criterion is not met, so the gradient descent reaches the maximum of 750 iterations, and it has a lower value for precision / recall. However, choosing too big of a learning rate makes the algorithm spread the points over the maximum floating point value before the size threshold is checked. This behaviour can be observed in Figure 9. Thus, we opted for a learning rate in between $\frac{n}{12 \cdot 3.16}$ and $\frac{n}{12 \cdot 10}$.

5.2 Runtime Efficiency Comparison

Next, we analyse the runtime efficiency improvement, again compared to existing Ht-SNE implementations, namely the Poincaré and the exact versions. In the following, we compare not only the absolute run time, but also the effect of hyperparameter θ on the run time.

Absolute Run Time Comparison

In this experiment, 3 versions of Ht-SNE were timed against each other: the exact and accelerated Poincaré versions [31] and our version. Each data-set from Table 1 was sampled 5 times for 10 different sizes $\frac{1}{10}n, \frac{2}{10}n, \ldots, n$, where n is the data-set size. All Ht-SNE variants were run on each of these samples, timing the negative forces' computation, as this is the limiting factor for t-SNE when it comes to absolute runtime [34]. These tests were performed on all cores of an Intel Xeon 6448Y "Sapphire Rapids", equipped with 64 cores and a memory bandwidth of 400 GB/s.

Table 2 shows the results after performing this experiment. The table clearly indicates a significant speed-up of our proposed method compared to the exact version. Moreover, the statistics of the Poincaré algorithm are very similar to our own implementation, further supporting the claim that the Lorentz model can be used to increase quality of embeddings, at a similar rate. A plot of these results can be found in the appendix, in Figure 11.

Effect of Theta on the Run Time

As discussed in Section 5.1, θ is the main hyperparameter for our algorithm. Because it dictates the degree of approximation in the octree, it directly influences the runtime of the algorithm. Hence, in this test, we ran the algorithm with a range of $\theta \in \{0.0, 0.1, \ldots, 1.0\}$ and timed the negative force computation time. These tests were performed on all 4 cores of an Intel i5-4690.

The results can be seen in Figure 10. As expected, by increasing the value of θ, we get better time performance.

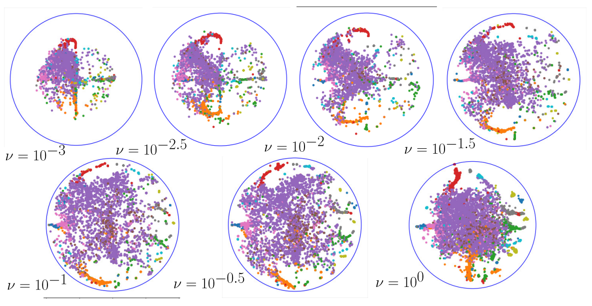

Figure 9: Final embeddings for different values of ν. For the $\nu = 1$ case, the algorithm executed 29 iterations before it exploded past the float maximum value.

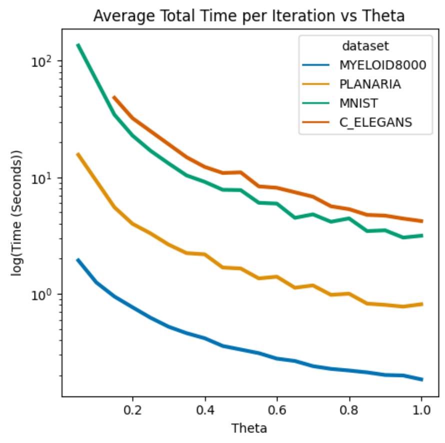

Figure 10: Run-time change when varying θ on the data-sets in Table 1

6 Responsible Research

With the advancements of big-data processing and artificial intelligence in recent times, the need for research performed respecting ethical and reproducibility guidelines has increased. In this section, we present steps that have been taken to ensure reproducibility, as well as some ethical considerations regarding the use of visualization algorithms.

6.1 Reproducibility

Reproducibility is an important aspect in the world of research. Even more so in the domain of Computational Sciences, where codebase can be easily shared, and the results are most of the time deterministic. This aspect is relevant because it enables other people to follow and verify the work one has done, to be able to build on top of it with their own research, or to just better grasp the topic being discussed. This has been the goal of the scientific method from its creation: generating verifiable knowledge.

To this end, this paper implements several steps that ensure its reproducibility, as proposed in [33]. Firstly, the codebase for this paper, alongside a digital copy of this paper, is publicly available at respository.tudelft.nl, and https://github.com/XDead27/hyperbolic-tsne-lorentz. The current commit hash that this paper is based on is `d7d3414`. Next, the source code has an MIT licence, supporting further improvements and developments, as recommended in [32]. Moreover, all formats used are non-proprietary and belong to long-established technologies (e.g., python, portable network graphics) and ensure their usability in the future. Lastly, the `README` file in the repository includes all the necessary information about the versions of the used tools and the hardware specifications that the experiments were run on. It also includes a step-by-step guide for replicating these results.

	Exact [s]				Poincaré [s]				Lorentz [s]			
Data Set	min	avg	std	max	min	avg	std	max	min	avg	std	max
MYELOID8000	2.15	2.83	0.67	5.96	0.085	0.388	0.635	3.357	0.125	0.349	0.717	3.581
PLANARIA	15.37	21.23	5.20	28.82	1.07	1.59	0.39	4.34	0.85	1.76	0.91	4.35
MNIST	147.39	219.32	54.02	278.17	3.18	5.36	1.14	8.95	2.93	5.96	1.29	9.42
C_ELEGANS	239.06	328.46	79.92	450.51	5.11	6.26	0.82	9.08	3.81	6.13	1.09	9.27

Table 2: Absolute run-time statistics for the exact and accelerated Lorentz and Poincaré versions of Ht-SNE

6.2 Misrepresenting Data

Data visualization algorithms aid people in representing complex data (that has little to no significance for a human) in a way that it retains as much of its original properties as possible, while making it understandable. Because this paper deals with one such algorithm, an important topic to discuss is the cases in which this visualization technique could possibly steer scientists (or other people) to making erroneous hypotheses about the data.

Firstly, because our algorithm for Ht-SNE using the Lorentz Hyperboloid model represents Euclidean high-dimensional data to 2-dimensional hyperbolic space, it runs the risk of failing to capture relevant properties of said data. This is why all representations created by this algorithm are just approximations of the actual data. If we consider the trivial case of 4 points in 3-dimensional space, placed at equal distances from each other (e.g., in the shape of a triangle-based pyramid), trying to embed them in 2-dimensional space, we quickly realize that we cannot maintain the property of equal pairwise distances any more. Situations like these might lead scientists and users to making erroneous hypothesis about the underlying structure of the data that is being represented.

Secondly, visualization techniques can be used to purposefully mislead or persuade an audience. This is a studied aspect in the domain of data visualization, and various solutions have been proposed to detect and counteract attempts at malicious misrepresentation of the data by using visualization algorithms [21; 27]. Because of this, it is important to note that our proposed solution for hyperbolic t-SNE does not fall outside this sphere of concern. Being a dimensionality reduction algorithm for high-dimensional data visualization, it serves as the starting point of a demonstration, not as its main argument. Therefore, any representation using this algorithm should be accompanied by a well-structured and sane argument for the point being made. Otherwise, due to the nature of high-dimensional data and hyperbolic space, certain properties of the data might be under-represented (or over-represented) in the visualization, leading to a skewed perception of the actual data.

7 Conclusions & Future Work

In summary, we investigated the effect of using the Lorentz Hyperboloid model as the embedding space for hyperbolic t-SNE. After performing qualitative experiments, we showed that using this model improves the embedding quality of the embeddings, by making use of its properties, such as its high numerical stability. By using an octree as an acceleration data-structure (as part of the Barnes-Hut scheme), we improved the runtime of the algorithm from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$ compared to the non-accelerated version. After performing experiments on the run time of the algorithm, we proved that our implementation has a similar run-time, but a higher quality performance compared to the Poincaré variant. Moreover, we showed that, by modifying the hyperparameters, the user can choose the trade-off between better quality or lower run-times, making our variant a robust alternative to existing Ht-SNE implementations.

A drawback of using this variant is that the computation of the maximum width of the cells in the octree relies on an approximation (as discussed in Subsection 4.1). This might lead to the computed width being smaller than the actual maximum width, leading to incorrect summarizations. Because showing the exact implications of this fact are purely mathematical, it falls outside the scope of this paper.

Another limitation of using the octree as an acceleration structure for the hyperboloid is that the hyperboloid is a flat surface, while the cube is a volume. Therefore, most of the volume of the cube would be empty. A new splitting criterion that takes into account the shape of the hyperboloid could be introduced, thus minimizing the amount of empty space in each cell.

Acknowledgements

I want to express my gratitude to my supervisor, Martin Skrodzki, and to my responsible professor, Dr. Elmar Eisemann, for their continuous support through valuable feedback and insights. Moreover, I want to thank my teammates for the help provided and for the valuable discussions.

A Additional Experimental Results

In the following, we include several images and tables that were the outcome of experiments.

References

[1] James W. Anderson. Hyperbolic geometry. 1999.

[2] J. H. Barnes and Piet Hut. A hierarchical o(n log n) force-calculation algorithm. *Nature*, 324:446–449, 1986.

[3] Anna C. Belkina, Chris Ciccolella, Rina Anno, Richard L. Halpert, Josef Spidlen, and Jennifer E. Snyder-Cappione. Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature Communications*, 10, 2019.

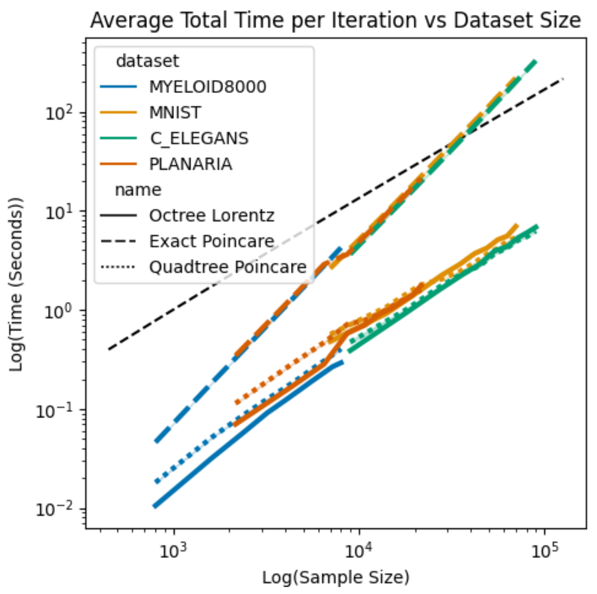

Figure 11: Time per iteration versus data-set size for all the data-sets.

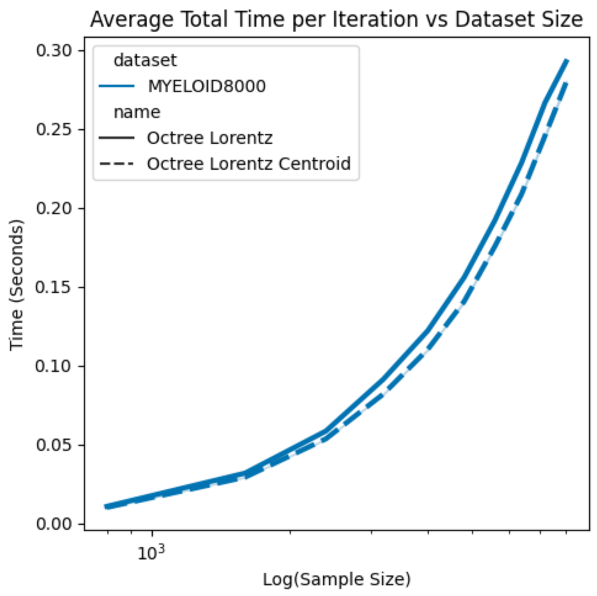

Figure 12: Comparison between Lorentz Ht-SNE using the pseudo-Fréchet mean and the Lorentz centroid

[4] Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri V. Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1:62, 2010.

[5] Christopher J. C. Burges. Dimension reduction: A guided tour. *Found. Trends Mach. Learn.*, 2, 2010.

[6] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. *ArXiv*, abs/2105.14686, 2021.

[7] Leo Corry. Hermann minkowski and the postulate of relativity. *Archive for History of Exact Sciences*, 51:273–314, 1997.

[8] Andrej Cvetkovski and Mark Crovella. Multidimensional scaling in the poincaré disk. *ArXiv*, abs/1105.5332, 2011.

[9] P. Demartines and J. Herault. Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, 1997.

[10] José M. Figueroa-O'Farrill, Emil Have, Stefan Prohazka, and Jakob Salzer. Carrollian and celestial spaces at infinity. *Journal of High Energy Physics*, 2022, 2021.

[11] Alexander G. Gray and Andrew W. Moore. 'n-body' problems in statistical learning. In *Neural Information Processing Systems*, 2000.

[12] Yunhui Guo, Hao-Tian Guo, and Stella X. Yu. Co-sne: Dimensionality reduction and visualization for hyperbolic data. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20, 2021.

[13] Marc Hammarlund, Oliver Hobert, David M. Miller, and Nenad Sestan. The cengen project: The complete gene expression map of an entire nervous system. *Neuron*, 99:430–433, 2018.

[14] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[15] Anna Klimovskaia, David Lopez-Paz, Léon Bottou, and Maximilian Nickel. Poincaré maps for analyzing complex hierarchies in single-cell data. *Nature Communications*, 11, 2019.

[16] Dmitry Kobak and Philipp Berens. The art of using t-sne for single-cell transcriptomics. *Nature Communications*, 10, 2018.

[17] Dmitri V. Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 82 3 Pt 2:036106, 2010.

[18] Jan Krumsiek, Carsten Marr, Timm Schroeder, and Fabian J Theis. Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network. *PLoS ONE*, 6, 2011.

[19] Marc Teva Law, Renjie Liao, Jake Snell, and Richard S. Zemel. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, 2019.

[20] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

[21] Leo Yu-Ho Lo, Ayush Gupta, Kento Shigyo, Aoyu Wu, Enrico Bertini, and Huamin Qu. Misinformed by visualization: What do we learn from misinformative visualizations? *Computer Graphics Forum*, 41, 2022.

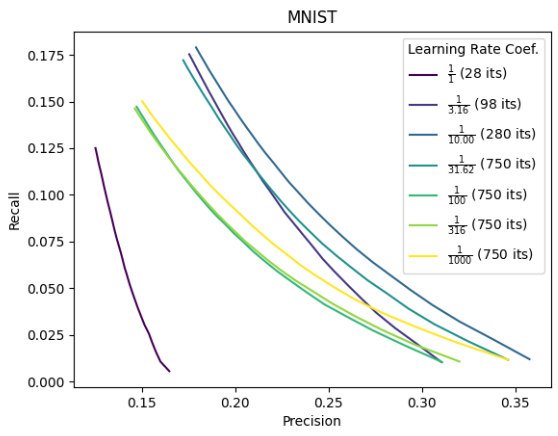

Figure 13: Effect on the precision vs. recall of the learning rate on the MNIST data-set

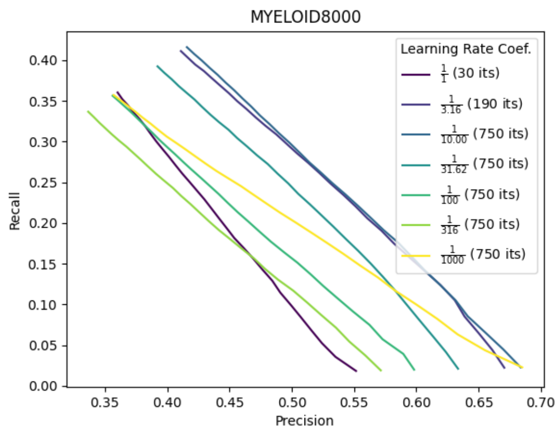

Figure 14: Effect on the precision vs. recall of the learning rate on the MYELOID8000 data-set

[22] H. Martini and K.J. Swanepoel. The geometry of minkowski spaces — a survey. part ii. *Expositiones Mathematicae*, 22(2):93–144, 2004.

[23] Horst Martini, Konrad J. Swanepoel, and Gunter Weiss. The geometry of minkowski spaces — a survey. part i. *Expositiones Mathematicae*, 19:97–142, 2007.

[24] Donald Meagher. *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. 1980.

[25] Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry, 2018.

[26] Victor Onclinx, John A. Lee, Vincent Wertz, and Michel Verleysen. Dimensionality reduction by rank preservation. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[27] Anshul Vikram Pandey, Anjali Manivannan, Oded Nov, Margaret L. Satterthwaite, and Enrico Bertini. The persuasive power of data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 20:2211–2220, 2014.

[28] Nicola Pezzotti, Thomas Höllt, Boudewijn P. F. Lelieveldt, Elmar Eisemann, and Anna Vilanova. Hierarchical stochastic neighbor embedding. *Computer Graphics Forum*, 35, 2016.

[29] Mireya Plass, Jordi Solana, F. Alexander Wolf, Salah Ayoub, Aristotelis Misios, Petar Glazar, Benedikt Obermayer, Fabian J Theis, Christine Kocks, and Nikolaus Rajewsky. Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics. *Science*, 360, 2018.

[30] Martin Skrodzki. personal communication.

[31] Martin Skrodzki, Hunter van Geffen, Nicolas F. Chaves-de Plaza, Thomas Hollt, Elmar Eisemann, and Klaus Hildebrandt. Accelerating hyperbolic t-sne. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–13, 2024.

[32] Victoria Stodden. Enabling reproducible research: Open licensing for scientific innovation. *Intellectual Property Law eJournal*, 2009.

[33] Victoria Stodden. Reproducible research: Addressing the need for data and code sharing in computational science. *Computing in Science and Engineering*, 12:8–12, 2010.

[34] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245, jan 2014.

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

[36] Kevin Verbeek and Subhash Suri. Metric embedding, hyperbolic space, and social networks. *Proceedings of the thirtieth annual symposium on Computational geometry*, 2014.

[37] Moritz von Looz, Henning Meyerhenke, and Roman Prutkin. Generating random hyperbolic graphs in subquadratic time. In *International Symposium on Algorithms and Computation*, 2015.

[38] Scott A. Walter. The non-euclidean style of minkowskian relativity. 1998.

[39] Menglin Yang, Min Zhou, Zhihao Li, Jiahong Liu, Lujia Pan, Hui Xiong, and Irwin King. Hyperbolic graph neural networks: A review of methods and applications. *ArXiv*, abs/2202.13852, 2022.

[40] Yuansheng Zhou and Tatyana O. Sharpee. Hyperbolic geometry of gene expression. *iScience*, 24, 2020.

[41] Çaglar Gülçehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter W. Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic attention networks. *ArXiv*, abs/1805.09786, 2018.