Hyperbolic t-SNE with a Quadtree Splitting in the Cartesian Coordinate System

Yehor Kozyr[1]

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

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

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

Name of the student: Yehor Kozyr
Final project course: CSE3000 Research Project
Thesis committee: Elmar Eisemann, Martin Skrodzki, Gosia Migut

Abstract

With the rapid growth in data collection, efficient data processing is critical. Dimensionality reduction methods, like t-distributed stochastic neighbour embedding (t-SNE), compress high-dimensional data into embeddings that preserve the key features of the datasets making data less sparse and easier to process further. Recent improvements suggest that using hyperbolic space for data representation can benefit embeddings. As it is a new technique, it remains computationally expensive. Previous work suggests that a Barnes-Hut approximation with a polar quadtree can be applied to the Poincaré disk model to approximate the result of hyperbolic t-SNE and accelerate its calculation. However, the polar quadtree is proposed as a solution to accelerate the calculation without exploring alternative approaches. Aiming to close this gap, we propose an acceleration method using Barnes-Hut approximation with a Cartesian quadtree. We experimentally compare our acceleration method to a polar quadtree and showcase its lower execution time without the loss of quality of the embeddings. Implementation and scripts for the experiments and plots are available at https://github.com/Sne4kers/hyperbolic-tsne.

1 Introduction

With an increasing amount and type of data being collected yearly, the need to efficiently process that data rises. High-dimensional data is often problematic for algorithms to process due to the sparsity of the data and the additional computational power required to process it. In order to extract key insights about the datasets, they are compressed into embeddings. In this research, we work on t-distributed stochastic neighbour embedding (t-SNE). In the embeddings formed by t-SNE, the locality of the neighbourhoods is preserved – similar data points have a high probability of being closer to each other while different further away from each other [1].

With the rising use of embeddings, hyperbolic space has been shown to benefit the embedding of hierarchical data [6], trees [2], and additionally, new methods of dimensionality reduction in the hyperbolic space were proposed based on the t-SNE in the Euclidean space [7].

Hyperbolic embeddings are still computationally expensive when compared to the methods for solving equivalent problems in the Euclidean space. To tackle this problem, a method using a Barnes-Hut approximation was proposed by Martin Skrodzki et al. [8] that takes advantage of a polar quadtree. The quadtree is a data structure that recursively splits the 2-dimensional space into 4 regions. The original paper builds such a tree in the polar coordinate system – the splitting is done by splitting the region in polar quadrilaterals, based on the distance from the center and angle. Such a design choice is very intuitive as it splits a circle of the Poincaré disk model into similar regions, only covering regions within it. There is a downside to this method, as it forces the developer to regularly recalculate coordinates for the entire datasets to convert from one coordinate system to another, as building a polar quadtree requires constantly converting coordinates from the Euclidean space of the Poincaré disk model into the polar coordinate system.

With this research, we aim to compare the polar quadtree acceleration method to a variation of the quadtree with splitting in the Cartesian coordinate system. A quadtree with such splitting allows us to maintain only the coordinates of data points in the Poincaré disk model and will remove the need to convert them regularly to the intermediate polar coordinate system, alleviating some computations per iteration. Even though this way of building a tree is more widespread in Euclidean space applications, it presents additional challenges when applying the Barnes-Hut approximation in the hyperbolic space due to the difficulties of fitting squares into the circle of the Poincaré disk model.

In the following section, we discuss background information about adapting t-SNE to hyperbolic space in Section 2, specifications of the proposed quadtree with splitting in the Cartesian coordinate system in Section 3, evaluation of the proposed quadtree comparing it to the polar quadtree and exact computation in Section 4, ethical implications of this research in Section 5, and conclusions with suggestions of future work in Section 6.

2 Background

2.1 t-distributed Stochastic Neighbor Embedding

t-distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reduction technique that was proposed for better visualization of the similarities within the data by Laurens van der Maaten and Geoffrey Hinton in 2008 [1]. The algorithm focuses on preserving local neighbourhoods from the high-dimensional data in the low-dimensional representation. It is done by minimising the Kullback–Leibler (KL) divergence on 2 distributions. The high-dimensional input $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subseteq \mathbb{R}^d$, where d is the number of dimensions, is interpreted as a set of conditional probabilities:

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

where σ_i is the variance of the Gaussian centered on datapoint $\mathbf{x}_i$, and $p_{i|i} = 0$. In the low-dimensional embedding $\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n\} \subseteq \mathbb{R}^{d'}$ similarity of points $\mathbf{y}_i$ and $\mathbf{y}_j$ is modelled as a probability:

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

To compute the low-dimensional embedding of t-SNE, principal component analysis (PCA) is first applied to the original data, followed by the procedure of gradient descent to minimise the KL divergence:

$$\mathrm{KL}\left(P \parallel Q\right) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3}$$

The gradient of KL divergence:

$$\frac{\delta C}{\delta \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j) \quad (4)$$

As a result, the local minimum of dissimilarity between the low-dimensional and high-dimensional representation of the data will be found. A regular implementation of t-SNE would have a run time of $O(n^2)$ as a pair-wise combination of each point is required for the computation.

2.2 Barnes-Hut approximation for t-SNE

In order to accelerate the quadratic calculation of all of the pairwise combinations, the approximation is used – the Barnes-Hut algorithm.

The obtained gradient formula from the KL divergence can be rewritten as a negative and positive-sum which can be viewed as negative and positive forces in a n-body system, so the Barnes-Hut algorithm can be applied to negative forces. In this approximation, the space is split into regions using a quadtree, after which it is queried with stopping criteria:

$$\frac{r_{cell}}{\|\mathbf{y}_i - \mathbf{y}_{cell}\|} < \theta \quad (5)$$

where r_{cell} is the maximum distance within the cell, $\mathbf{y}_{cell}$ is a barycenter of the cell, and θ is a hyperparameter regulating how much of cells will get approximated. The impact of change in points that are far away is small on the target point. Using this condition they are grouped in a single cell that will be later used in a calculation as a single point with a bigger multiplier, grouping multiple points into one, reducing the computational complexity of the stage to $O(n \log n)$ [3].

2.3 Hyperbolic space and Poincaré disk model

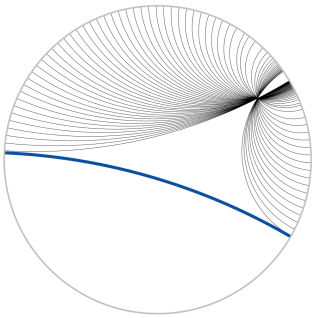

Figure 1: The Poincaré disk model of hyperbolic space with straight lines that are parallel to a blue straight line. Straight lines in hyperbolic space appear to be curved in the Poincaré disk model.

While there are many ways to model hyperbolic space geometry behaviour, we will use the Poincaré disk model. In this model, the entire space is represented by a disk. A straight line in hyperbolic space would be modelled in a Poincaré disk model as a circular arc contained within a disk. Formally expressed, the Poincaré disk model is the space $\mathbb{D} = \{\mathbf{y} \in \mathbb{R}^2 : ||\mathbf{y}|| < 1\}$ with a metric:

$$g^{\mathbb{D}}_{\mathbf{y}} = \lambda^2_{\mathbf{y}} g^E \quad (6)$$

where $\lambda_{\mathbf{y}} = \frac{2}{1-||\mathbf{y}||^2}$, g^E – standard scalar product of $\mathbb{R}^2$ and $||.||$ – standard norm of $\mathbb{R}^2$. The hyperbolic distance $d^{\mathcal{H}}(\mathbf{y}_i, \mathbf{y}_j)$ between 2 points in the Poincaré model is given by:

$$d^{\mathcal{H}}_{ij} = \cosh^{-1}\left(1 + 2\frac{||\mathbf{y}_i - \mathbf{y}_j)||^2}{(1 - ||\mathbf{y}_i||^2)\,(1 - ||\mathbf{y}_j||^2)}\right) \quad (7)$$

2.4 Approximation of t-SNE for hyperbolic space

As we expand on the idea of acceleration of hyperbolic t-SNE introduced by Martin Skrodzki et. al. and compare the Cartesian quadtree to the polar quadtree in this scenario, the same formula of gradient was used for finding local minima of KL-divergence:

$$\frac{\delta C^{\mathcal{H}}}{\delta \mathbf{y}_i} = 4\left(\sum_{j \neq i} p_{ij} q^{\mathcal{H}}_{ij} Z^{\mathcal{H}} \frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i} - \sum_{j \neq i} \left(q^{\mathcal{H}}_{ij}\right)^2 Z^{\mathcal{H}} \frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i}\right), \quad (8)$$

where $Z^{\mathcal{H}} = \sum_{j \neq i} \left(1 + {d^{\mathcal{H}}_{ij}}^2\right)^{-1}$ [8, Eq. 15]. Similarly, as in polar quadtree acceleration, we will approximate the negative part of the gradient, based on the assumption that points with a small effect on the given point can be grouped. Therefore, far away from point i points k_1, k_2, ... k_n in a single cell, can be approximated as:

$$n\left(q^{\mathcal{H}}_{i,cell}\right)^2 Z^{\mathcal{H}} \frac{\delta d^{\mathcal{H}}_{i,cell}}{\delta \mathbf{y}_i} \quad (9)$$

instead of $\sum_{j \in k_1,..,k_n} \left(q^{\mathcal{H}}_{i,j}\right)^2 Z^{\mathcal{H}} \frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i}$. Barnes-Hut condition is also modified for hyperbolic space: $r_{cell}/d^{\mathcal{H}}_{i,cell} < \theta$.

To accelerate the algorithm, the depth-first traversal is performed on the quadtree that is used, with a stopping condition as described for the Barnes-Hut algorithm. At the point of stopping, all data points that are bounded by the subtree are approximated to their Einstein midpoint and used to reduce the number of sum elements according to the Expression 9.

3 Methodology

3.1 Basic quadtree implementation

We propose a quadtree splitting in the Cartesian coordinate system for a Barnes-Hut approximation in a Poincaré disk model of hyperbolic space. The key difference from a regular quadtree splitting in the Cartesian is the fact, that for the approximation algorithm, the maximum distance within each bounding box should be known. While it is a trivial task for a tree spanning 2-dimensional Euclidean space, it has some issues when applied to a Poincaré disk model. With an unmodified implementation of the quadtree, the nodes can result in the bounding boxes that include regions beyond the infinity line of the Poincaré disk. This makes it impossible to determine the maximum distance within a cell, which leads to the Barnes-Hut algorithm not being able to stop in cells that include regions beyond the infinity line. As a fix for this issue, we introduce a change to stopping criteria for when building a tree - additionally, to requiring a single point within a cell,

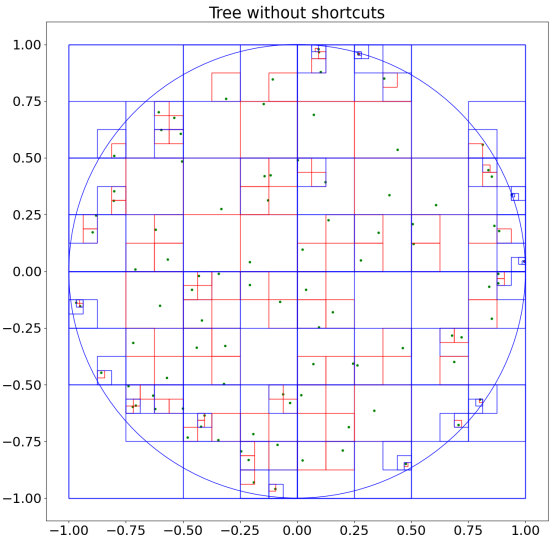

Figure 2: Cartesian tree before shortcuts were introduced applied to the Poincaré disk model.

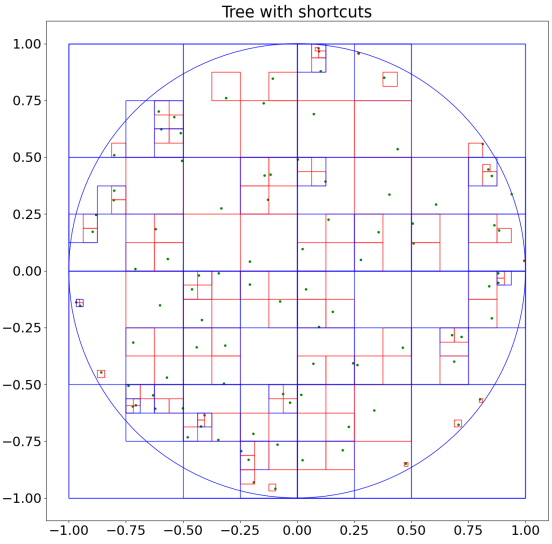

Figure 3: Cartesian tree after shortcuts were introduced applied to the Poincaré disk model.

we pose an additional condition that the leaf cell should not contain any region beyond the infinity line. This way, in the limit, we approximate the exact shape of the disk and still maintain the ability to make a set of bounding boxes with known distances containing all of the points.

With this structure, the depth of the tree in certain areas is increased substantially, due to our stopping criteria, but this can be easily fixed - we introduce shortcuts to the tree, by removing all nodes that have a single child, reducing the query time and the depth of the tree.

Visualization of the tree without shortcuts (Fig. 2) and with shortcuts (Fig. 3) are compared on the same set of points.

All of the code was developed based on the original implementation of the algorithm of hyperbolic t-SNE by Martin Skrodzki et. al. The algorithm with the original implementation of the quadtree in the Cartesian coordinate system, with shortcuts, was used to collect data for multiple experiments. It resulted in sensible results when testing the preservation of the local neighbourhood (Fig. 4) and the effect of Barnes-Hut parameter θ on execution time (Fig. 5).

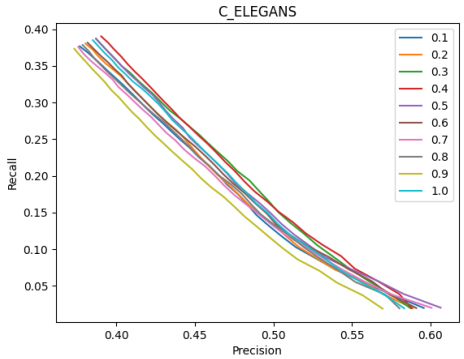

Figure 4: Recall and precision graph for a 10k points sample of C_ELEGANS dataset, exploring the dependency of theta on the resulting neighbourhoods

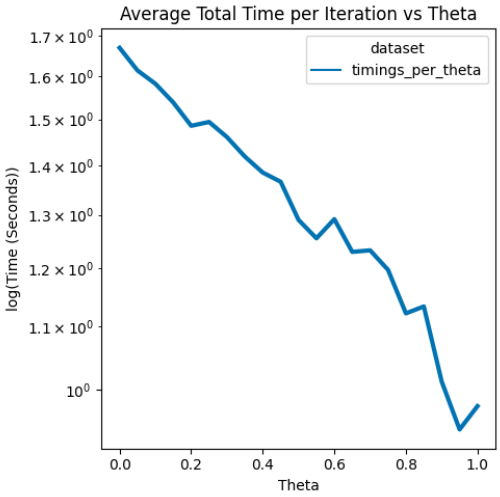

Figure 5: Effect of theta on time spent on a single iteration of the original quadtree implementation on a 10k sample of the C_ELEGANS dataset

Although the algorithm worked with data and showed similar to the polar quadtree results, its timing measurements were horrible. In time per iteration, it is comparable to the exact

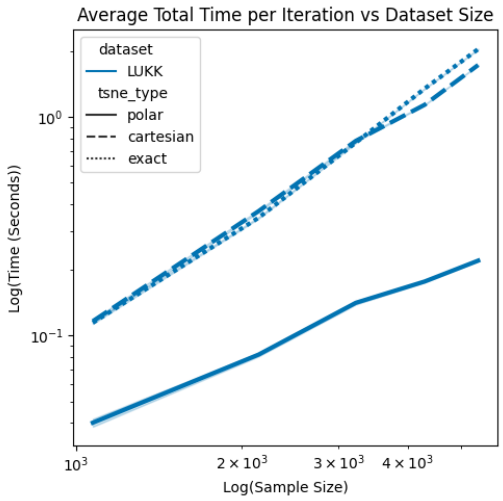

Figure 6: Time per iteration on samples of LUKK dataset. Before the change in the calculation of maximum distance within cells, the quadtree with splitting in the Cartesian coordinate system is comparable in performance to the exact computation, losing significantly to the polar quadtree.

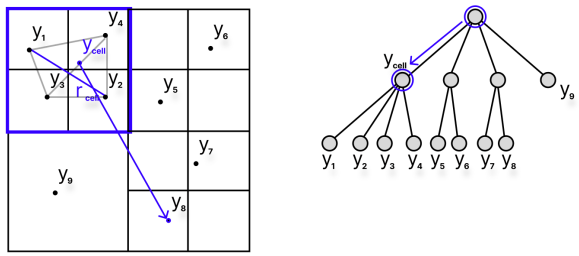

Figure 7: Modified Cartesian quadtree and visualisation of the Barnes-Hut algorithm using it. The points $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_3$, $\mathbf{y}_4$ are grouped together as $\mathbf{y}_{cell}$ when approximating gradient for $\mathbf{y}_8$. Instead of using the maximum distance within the selected cell (indicated with blue), r_{cell} is used which is the maximum distance between $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_3$, $\mathbf{y}_4$. Modified from [8] and [4].

computation (Fig. 6). Most of the time was spent on the calculation of negative forces, despite using the same code as the polar quadtree for that section of the algorithm. As the most amount of time was spent in the querying of the tree, we arrived at the assumption that it is the structure of the tree that drags the performance down. As our stopping condition was modified to also account for the fact that quadtree with splitting in the Cartesian coordinate system would include regions beyond infinity boundary, it required way more queries to approximate the forces, as it would go deeper in some cases than polar quadtree.

To test this assumption, the code behind polar quadtree implementation and Cartesian quadtree was modified to count the average depth that is reached at each iteration of the t-SNE. At each iteration, when negative forces are computed, for each of the points the number of centres of masses used to calculate negative forces for that point will be recorded. The experiment was run on a 10k sample of the C_ELEGANS dataset. The polar quadtree used on average 418.98 points in the exaggeration stage and 280.7 in the second stage. The Cartesian quadtree used on average 7313.55 points in the exaggeration stage and 7882.49 in the second stage. This clearly explains the difference in performance.

3.2 Quadtree modifications

After the issue was identified, we tried to eliminate it. What is important here is to not over-engineer the solution. The issue lies in the stopping condition that forces the algorithm to go deeper – because of the way the tree is built, many cells include regions beyond the infinity border or near it. Such cells are queried deeper as we can not stop in them due to not having the metric of maximum possible distance within them or it is huge within them. This can be changed by better approximating distance within a cell in such cells, allowing for the algorithm to stop in them despite being close to the infinity region. Any complex ideas, for example, some heuristic on top of a constructed convex-hull, are unacceptable as the solution must be $O(n)$ or lower in computational complexity to compete with the polar quadtree implementation.

Polar quadtree uses $O(1)$ to calculate such distance – it takes the maximum possible distance between vertices of the quadrilateral, so we decided to also stick to $O(1)$ in the updated version of the Cartesian quadtree. As a way to approximate the maximum distance within a cell, we propose a heuristic of maximum distance between its children's barycenters. Figure 7 shows the Barnes-Hut algorithm applied with such a heuristic. It showcased itself with outstanding performance on the same experiment with a 10k sample of C_ELEGANS dataset, on average using only 121.17 points per point when calculating negative forces in the exaggeration step and 97.98 in the second stage.

As the Cartesian quadtree with the heuristic of maximum distance that uses children's barycenters showed a good performance in the summarization experiment, we decided to stick to it in all further experiments.

When comparing theoretical estimates of the polar and Cartesian quadtrees, it is difficult to estimate the time complexity of querying the tree, as it depends on the hyperparameters set by the developer used directly during querying – θ, as well as on the distribution of points in the space, that can be influenced, for example, by the momentum parameters, number of iterations in the early exaggeration stage, learning rate. When it comes to building the tree, it is evident that the Cartesian tree is built in $O(n \log n)$ time, as it recursively splits the available points into 4 regions and executes the same linear set of operations on each level on the identified subsets of points, which is equivalent to $O(n \log n)$ complexity of the entire algorithm, according to the Master theorem.

4 Evaluation

All of the experiments were conducted on the same machine with an AMD Ryzen 5 5600H on Linux. All of the code for experiments is publicly available in the GitHub repository with code used for this project.

Table 1: Data sets used in the experiments with their main metrics – the number of points, dimension of the data set, the number of labelled classes.

Name	Data type	Points	Dim	Cl.
LUKK	single-cell	5372	369	4
MYELOID8000	single-cell	8000	11	5
PLANARIA	single-cell	21612	50	51
MNSIT	images	70000	784	10
WORDNET	lexical	82115	11	n/a
C_ELEGANS	single-cell	89701	20222	37

Experimental setup

For the experiments, the same procedure was used for most of the experiments as in the paper by Martin Skrodzki et. al. described in Section 5 [8], as we used the same codebase. The only different part of the algorithm that was replaced, is the computation of negative forces, as it directly depends on the quadtree implementation. As in the original paper [8], the initial learning rate was set to:

$$\eta = \frac{n}{12 \cdot 1000} \tag{10}$$

where n is the number of points. Replicating the original paper on accelerating the hyperbolic t-SNE, the parameters that were used are $\theta = 0.5$, momentum in the early exaggeration of 0.5, and momentum in the non-exaggerated stage of 0.8. Each setup would use 250 iterations of the exaggeration stage and 750 steps of the non-exaggerated stage, as well as the same early stopping condition during the non-exaggerated stage – stopping when any point is within 10^{-4} from the boundary of the Poincaré disk boundary. The settings were replicated to reflect as close as possible to the implementation of the algorithm with the polar quadtree presented in the original paper.

4.1 Comparison of the embedding quality

As in the paper on the polar quadtree, we will compare the quality of the embedding via how nearest neighbours are preserved using a precision/recall metric [5]. Keeping the parameters the same, we will use $k_{max} = 30$. For each $k \in \{1, 2, ..., k_{max}\}$ true positive rate TP_k will be computed, formally $TP_k = N_k(X) \cap N_k(Y)$, representing the number of points from the high-dimensional neighbourhood of size k that are also present in the low-dimensional neighbourhood of size k in the embedding. Using this, we can calculate the metrics of precision $PR_k = \frac{|TP_k|}{k}$ and recall $RC_k = \frac{|TP_k|}{k_{max}}$, according to their definition. While the perfect neighbourhood preservation scenarios would imply $PR_k = 1$ and $RC_k = \frac{k}{k_{max}}$, that is not always feasible for t-SNE, we strive to show the difference in the quality of the embeddings between exact and accelerated versions using these metrics. Figures 8-13 show the precision/recall curves for all of the datasets we have tested on.

It is clear from the graphs, that hyperbolic t-SNE with an acceleration using quadtree with splitting in the Cartesian coordinate system does not lose significantly in quality of the embeddings, mostly following trends of both the exact method and accelerated implementation using a polar quadtree. Depending on the data set, the algorithm using a Cartesian quadtree can outperform (Fig. 8, 11, 13) or underperform (Fig. 12) relative to the implementation using a polar quadtree. Figures 14, 15, 16 show the embeddings constructed after a single run of the algorithm on the MNIST dataset.

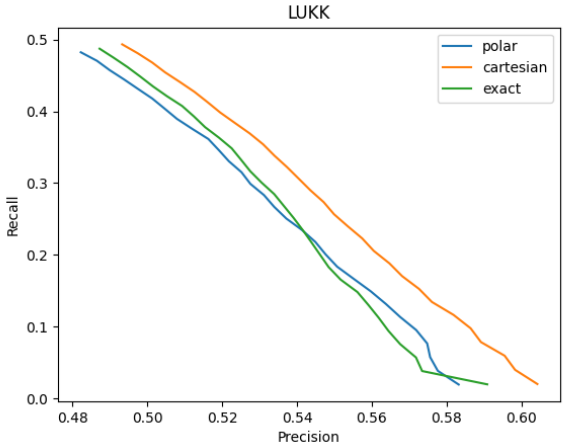

Figure 8: Precision/recall graph comparing algorithms on the LUKK dataset.

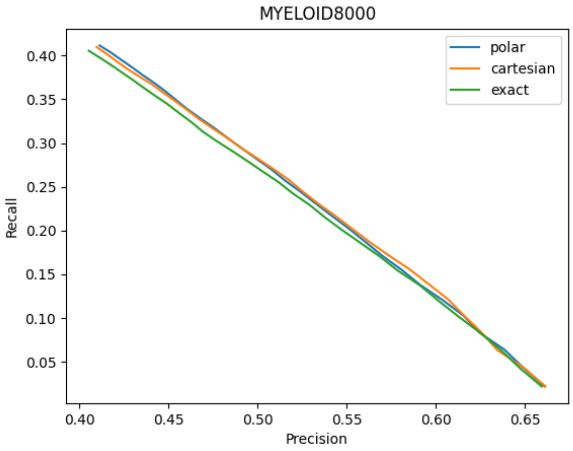

Figure 9: Precision/recall graph comparing algorithms on the MYELOID8000 dataset.

4.2 Time comparison

As we are focusing on the Cartesian quadtree as the acceleration data structure, the key point of comparison is the execution time. Table 2 shows the measurements collected during 5 runs of each implementation on each dataset.

It is evident from the table that a Cartesian quadtree executes faster on average than a polar quadtree. Additionally, for all datasets with the exception of LUKK, the average time

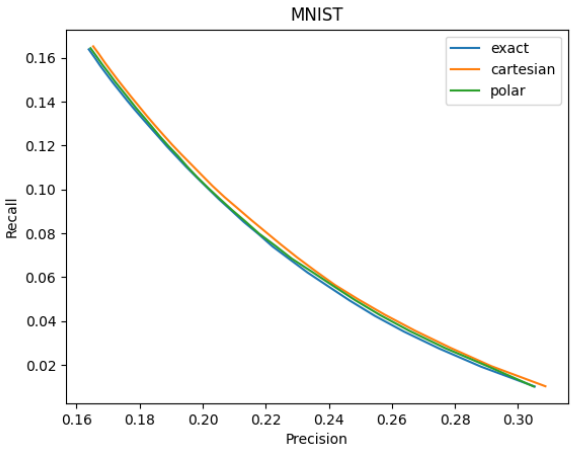

Figure 10: Precision/recall graph comparing algorithms on the MNIST dataset.

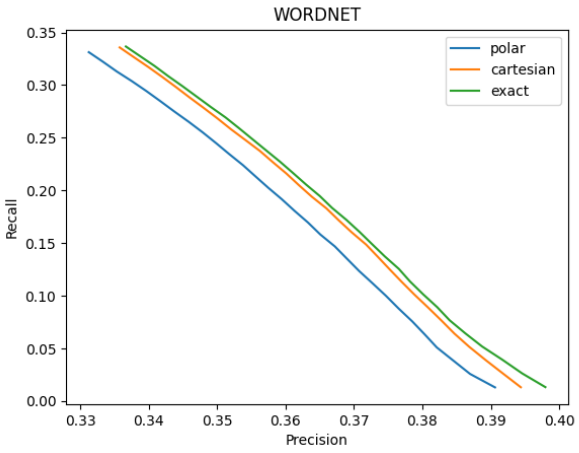

Figure 11: Precision/recall graph comparing algorithms on the WORDNET dataset.

Table 2: Execution time of a single iteration recorded in 5 runs on each dataset. table includes the average of the measurements, as well as the minimum and maximum recorded time per iteration and standard deviation of the measurements.

Name	Cartesian[s] avg / min / max / std	Polar [s] avg / min / max / std
LUKK	0.17 / 0.06 / 0.70 / 0.08	0.24 / 0.84 / 0.07 / 0.06
MYELOID8000	0.15 / 0.09 / 0.59 / 0.05	0.40 / 0.03 / 1.44 / 0.15
MNIST	1.91 / 1.72 / 2.68 / 0.12	6.53 / 3.45 / 13.9 / 0.42
WORDNET	2.02 / 1.81 / 5.91 / 0.21	6.84 / 2.90 / 14.9 / 0.83
PLANARIA	0.65 / 0.56 / 1.09 / 0.05	2.05 / 1.07 / 4.76 / 0.43
C_ELEGANS	2.64 / 2.35 / 40.9 / 1.83	8.56 / 7.31 / 10.2 / 0.40

of iteration for the Cartesian quadtree is different from the polar quadtree by more than a standard deviation of time per iteration of the Cartesian quadtree, indicating a strong advantage over the polar quadtree implementation.

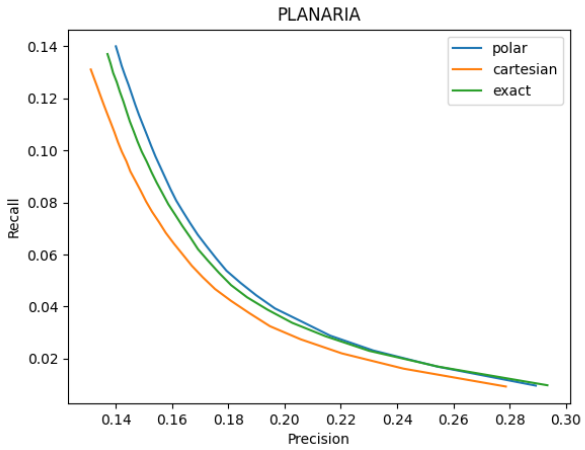

Figure 12: Precision/recall graph comparing algorithms on the PLANARIA dataset.

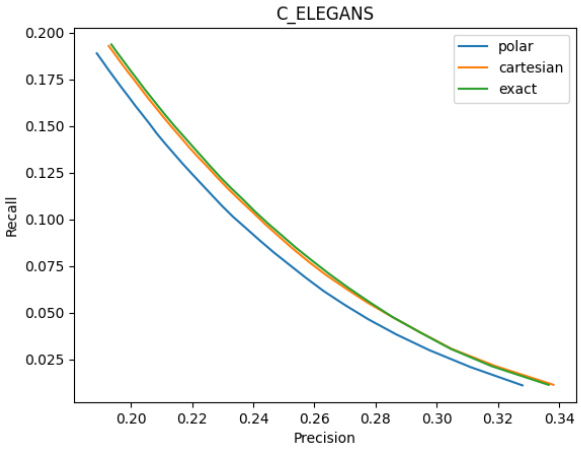

Figure 13: Precision/recall graph comparing algorithms on the C_ELEGANS dataset.

Additionally, a series of experiments were conducted to observe the performance of both polar and Cartesian quadtrees on different sample sizes of the datasets. Figure 17 showcases a comparison of both of the quadtrees and the exact computation. Both of the implementations outperform exact computation. Cartesian quadtree has a bit of a disadvantage on small sample sizes over polar quadtree while outperforming polar quadtree on bigger sample sizes.

As it was shown in Figure 17 that both acceleration data structures outperform exact computation, a series of runs were conducted on all datasets without exact computation due to its long execution time. Multiple runs were used to reduce the variance of the collected measurements and to get more accurate results. For each data set a sample of size $n/5$, $2n/5$, $3n/5$, $4n/5$ and n points was constructed to explore the relation between the size of the dataset and performance

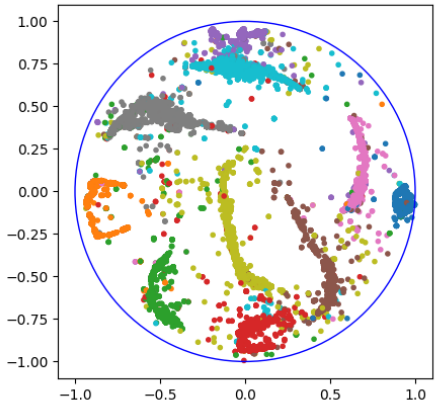

Figure 14: Embedding produced by the polar quadtree algorithm on the MNIST dataset.

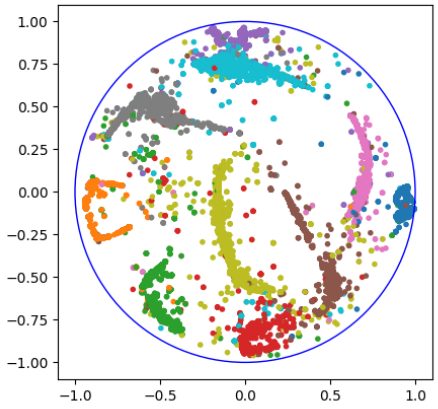

Figure 15: Embedding produced by the Cartesian quadtree algorithm on the MNIST dataset.

of the algorithms. Hyperbolic t-SNE using a polar quadtree and Cartesian quadtree were run on the same samples 5 times on each sample. As the result of the experiment, a plot of time per iteration with a logarithmic scale was produced. It is clear from Figure 18 that an implementation using a Cartesian quadtree has an advantage over a polar quadtree implementation, especially as the number of points increases.

As the scale is logarithmic, plots from both polar and Cartesian quadtrees have a similar incline of the lines, indicating their similar time complexity but with different constant coefficients, introduced by the difference in the tree building.

5 Responsible Research

In this research, we expand on the idea of accelerating the hyperbolic t-SNE. This work provides an improvement of the previous work on the acceleration of the hyperbolic t-SNE. However, it does not cancel out the ethical implications of

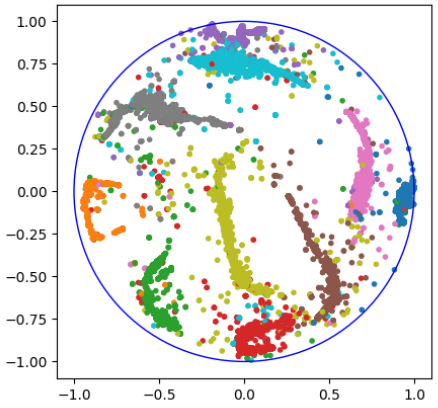

Figure 16: Embedding produced by the exact algorithm on the MNIST dataset.

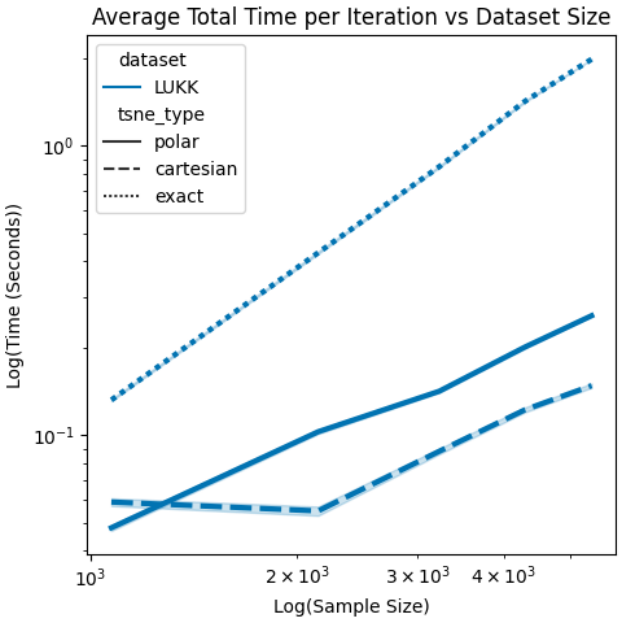

Figure 17: Time per iteration of hyperbolic t-SNE on 5 sample sizes of LUKK dataset. 5 runs of each algorithm on each sample size. Cartesian quadtree implementation loses on a very small sample size, while has an advantage over polar quadtree as the size of the sample increases.

possible use cases of the t-SNE algorithm and applications that are made possible with proposed improvements in speed. The algorithm can be widely used in different areas for data processing. Non-ethical results can be produced to manipulate original data or amplify biased opinions during the analysis of the data after the application of the algorithm. All experiments are performed in a reproducible manner – they replicate experiments done in previous works and all of the code used for experiments is publicly available.

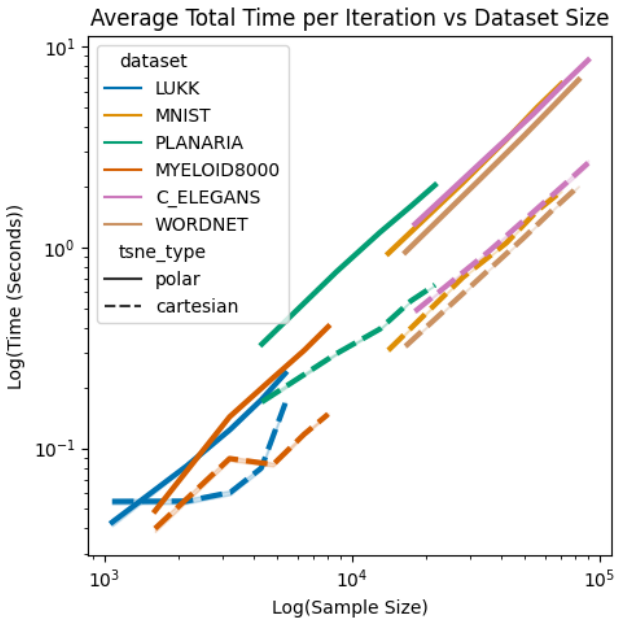

Figure 18: Time per iteration of hyperbolic t-SNE on 5 sample sizes of all the datasets. 5 runs of the algorithm using polar quadtree and Cartesian quadtree on each sample size. Cartesian quadtree implementation has an advantage over polar quadtree, as the size of the sample increases.

6 Conclusions and Future Work

6.1 Conclusions

With this research, we have shown that the use of the quadtree with splitting in the Cartesian coordinate system for hyperbolic t-SNE is possible. Although we were not able to reach positive results using a similar technique for computation of maximum distance within a cell as in the original research on the application of the polar quadtree, using a proposed approximation using the maximum distance between the cell's children's barycenters the algorithm was shown to be faster than the exact computation. Additionally, through a series of experiments, it was shown that the Cartesian quadtree outperforms the polar quadtree on the metric of time spent per iteration of the algorithm, without a sacrifice in the quality of the embeddings, due to a greater degree of approximation – using fewer points on average to approximate the gradient for each point.

6.2 Future work

It remains unclear, whether a selected set of parameters used in the experiments is optimal for the Cartesian tree. Additional investigation needs to be conducted on other possible ways to approximate distance within a cell other than the maximum distance between the cell's children's barycenters. Additionally, the previous approach of approximating the maximum distance should be investigated further, as it is possible that similar execution time results can be reached with a different θ parameter.

References

[1] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html.

[2] R. Sarkar, "Low distortion delaunay embedding of trees in hyperbolic plane," in *Graph Drawing*, M. van Kreveld and B. Speckmann, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 355–366, ISBN: 978-3-642-25878-7.

[3] L. van der Maaten, *Barnes-hut-sne*, 2013. arXiv: 1301.3342.

[4] L. van der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 93, pp. 3221–3245, 2014. [Online]. Available: http://jmlr.org/papers/v15/vandermaaten14a.html.

[5] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova, "Hierarchical stochastic neighbor embedding," *Computer Graphics Forum*, vol. 35, Jun. 2016. DOI: 10.1111/cgf.12878.

[6] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/59dfa2df42d9e3d41f5b02bfc32229dd-Paper.pdf.

[7] Y. Zhou and T. O. Sharpee, "Hyperbolic geometry of gene expression," *Iscience*, vol. 24, no. 3, 2021.

[8] M. Skrodzki, H. van Geffen, N. F. Chaves-de-Plaza, T. Höllt, E. Eisemann, and K. Hildebrandt, *Accelerating hyperbolic t-sne*, 2024. arXiv: 2401.13708.