Accelerating t-SNE using Fast Fourier Transforms and the Particle-Mesh Algorithm from Physics

Delchevalerie, Valentin; Mayer, Alexandre; Bibal, Adrien; Frénay, Benoît

Published in:
IJCNN 2021 - International Joint Conference on Neural Networks, Proceedings

DOI:
10.1109/ijcnn52387.2021.9534334

Publication date:
2021

Document Version
Peer reviewed version

Accelerating t-SNE using Fast Fourier Transforms and the Particle-Mesh Algorithm from Physics

Valentin Delchevalerie
Fac. of Computer Science
NaDI & naXys institutes
University of Namur
Namur, Belgium
valentin.delchevalerie@unamur.be

Alexandre Mayer
Department of Physics
naXys institute
University of Namur
Namur, Belgium
alexandre.mayer@unamur.be

Adrien Bibal
Fac. of Computer Science
NaDI institute
University of Namur
Namur, Belgium
adrien.bibal@unamur.be

Benoît Frénay
Fac. of Computer Science
NaDI institute
University of Namur
Namur, Belgium
benoit.frenay@unamur.be

***Abstract*—t-Distributed Stochastic Neighbor Embedding (t-SNE) is a well-known dimensionality reduction technique used for the visualization of high-dimensional data. However, despite several improvements, t-SNE is not well-suited to handle large datasets. Indeed, for large datasets, the computation time required to obtain the visualizations is still too high to incorporate it in an interactive data exploration process. Since t-SNE can be seen as an N-body problem in physics, we present a new variant of t-SNE based on a popular algorithm used to solve the N-body problem in physics called Particle-Mesh (PM). The problem is solved by first computing a potential in space and deriving from it the force exerted on each body. As the potential can be computed efficiently using Fast Fourier Transforms (FFTs), this leads to a significant speed up. The mathematical correspondence between t-SNE and PM presented in this work could also lead to other future improvements since more advanced PM algorithms have been developed in physics for decades.**

***Index Terms*—Machine learning, Numerical Physics, Dimensionality Reduction, Visualization, t-SNE, Particle-Mesh**

I. Introduction

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique well-suited for the visualization of high-dimensional data. SNE was first introduced by Hinton and Roweis [1] and then improved by van der Maaten and Hinton [2] to achieve t-SNE. Given a dataset $\mathcal{P} \equiv \{\mathbf{x}_i \in \mathbb{R}^p\}_{i=1}^N$, dimensionality reduction aims to find a mapping $\mathcal{P} \rightarrow \mathcal{Q} \equiv \{\mathbf{y}_i \in \mathbb{R}^q\}_{i=1}^N$ such that $q \ll p$, while leading to the smallest possible loss of information. If $q \leq 3$, $\mathcal{Q}$ can be used to produce a visualization of the dataset on a scatter plot. For data scientists, such visualizations are a common data analysis tool. It can help to reveal structures in data, and, in a more general way, help to get better insights.

As the aim of visualization is to quickly get data insights, it should be possible to incorporate it into an interactive data exploration process. In other words, the computation of visualizations should be fast enough to be incorporated in online tools. However, t-SNE exhibits an $\mathcal{O}\left(N^2\right)$ complexity in both time and memory that makes it unusable when N is too large. This typically happens when N becomes greater than a few thousands, i.e. most of the time. Several improvements have been proposed to tackle this issue. Van der Maaten [3] was the first to show the similarity between t-SNE and the well-known N-body problem in physics. He proposed to use tree-based algorithms as it was already proposed by Barnes and Hut [4] in astrophysics. This leads to an $\mathcal{O}\left(N \log N\right)$ numerical complexity, which is much better than the original $\mathcal{O}\left(N^2\right)$. Surprisingly, there has been no other attempt to take advantage of algorithms developed in physics for t-SNE.

The goal of this work is to describe t-SNE with another formalism borrowed from physics. This approach highlights the fact that t-SNE can be implemented using an algorithm called Particle-Mesh (PM) [5]. The idea is to compute a scalar potential $\phi\left(\mathbf{r}\right)$ in the low-dimensional space and to derive the forces exerted on each instance from it, similarly to what can be done for conservative forces in physics. The advantage lies in the fact that $\phi\left(\mathbf{r}\right)$ can be computed efficiently using the FFT algorithm. The fact that PM involves a well-known formalism in physics actually motivated our work. It will be possible indeed to bring new improvements to t-SNE in parallel to improvements of PM in physics.

The main contributions of this work are the following:

- A new formulation of t-SNE is presented inspired by the PM algorithm used in physics to solve the N-body problem. It leads to a significant acceleration and a new way to interpret t-SNE, with more physical insights.
- A quantitative study to evaluate the quality of visualizations is proposed, which is, to the best of our knowledge, not performed in other similar papers.

Section II describes the t-SNE algorithm and reveals its mathematical similarity with the N-body problem. Section III presents state-of-the-art means for accelerating t-SNE. Section IV describes the PM algorithm and shows how it can be transposed for t-SNE. Section V presents the experimental setup and results of our comparative study. Results are then discussed in Section VI, before concluding in Section VII.

II. Background on t-SNE

Let $\mathcal{P} \equiv \{\mathbf{x}_i\}_{i=1}^N$ be a set of instances in a high-dimensional (HD) space $\mathbb{R}^p$. The first step of t-SNE is to compute a pairwise similarity matrix $\mathbf{P}$ such that $p_{i,j} = \frac{p_{j|i} + p_{i|j}}{2N}$ with

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

and $p_{i,i} = 0$. The widths σ_i are determined using a meta-parameter u called the perplexity, which defines how large the neighborhood of each instance is. Good values for the perplexity are usually between 30 and 50. For each instance, σ_i is linked to u by $u = 2^{H(P_i)}$ with

$$H(P_i) = -\sum_j p_{j|i} \log p_{j|i} \tag{2}$$

and binary search can be used to approximate σ_i.

Now, let $\mathcal{Q} \equiv \{\mathbf{y}_i\}_{i=1}^N$ be the instance positions in a low-dimensional (LD) space $\mathbb{R}^q$ that are randomly initialized. Again, a pairwise similarity matrix $\mathbf{Q}$ can be computed. However, instead of the Gaussian distribution used in Equation (1), van der Maaten and Hinton [2] show that using a Student t-distribution in the LD space is preferable, leading to

$$q_{i,j} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}}, q_{i,i} = 0. \tag{3}$$

In order to make the $\mathbf{Q}$ similarities as close as possible to $\mathbf{P}$, t-SNE minimizes the Kullback-Leibler divergence

$$C = KL(\mathbf{P} \| \mathbf{Q}) = \sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}. \tag{4}$$

The pairwise similarities in the LD space are then as similar as possible to those in the HD space. It is then possible to draw conclusions about data by visualizing $\mathcal{Q}$. t-SNE minimizes C with a gradient descent (GD) by computing

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j \left(p_{i,j} - q_{i,j}\right) q_{i,j} Z \left(\mathbf{y}_i - \mathbf{y}_j\right), \tag{5}$$

where $Z = \sum_k \sum_{l \neq k} \left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}$ is a normalization factor. At step k of GD, Equation (5) is used in the update

$$\mathbf{y}_i^{(k)} = \mathbf{y}_i^{(k-1)} - \lambda \frac{\partial C}{\partial \mathbf{y}_i^{(k-1)}} + \eta(k) \left(\mathbf{y}_i^{(k-1)} - \mathbf{y}_i^{(k-2)}\right) \tag{6}$$

that iteratively moves each point $\mathbf{y}_i$ to minimize C, where λ is the learning rate and $\eta(k)$ is a momentum factor.

In order to reveal the similarity between t-SNE and the N-body problem, Equation (5) can be split in two terms as

$$\begin{aligned} \frac{-1}{4} \frac{\partial C}{\partial \mathbf{y}_i} &= \sum_j p_{i,j} \frac{\mathbf{y}_j - \mathbf{y}_i}{1 + d_{i,j}^2} + \sum_j \frac{1}{Z} \frac{\mathbf{y}_i - \mathbf{y}_j}{\left(1 + d_{i,j}^2\right)^2} \\ &= \sum_j \mathbf{F}_{i,j}^{attr} + \sum_j \mathbf{F}_{i,j}^{rep}, \end{aligned} \tag{7}$$

where $d_{i,j} = \|\mathbf{y}_i - \mathbf{y}_j\|$. For more mathematical details on how to obtain the equations in this section, see van der Maaten and Hinton [2]. The first term is defined as an attractive force. By analogy with physics, $-\frac{\partial C}{\partial \mathbf{y}_i}$ represents the force exerted on the instance i. Since $\mathbf{y}_j - \mathbf{y}_i$ is a vector directed from i to j, instance i will move closer to j according to this term. Similar reasons explain why the second term can be defined as a repulsive force. The factor $\frac{1}{Z}$ can be seen as a balancing factor between the attractive and repulsive forces. The next section explains how these two forces are usually computed.

III. State of the Art on Accelerating t-SNE

This section presents the most often used techniques to accelerate t-SNE. The acceleration performed for the attractive term and for the repulsive term are presented.

A. Computation of the Attractive Term

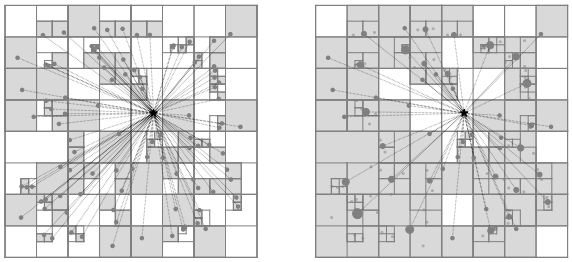
Computing the attractive force in Equation (7) takes an $\mathcal{O}\left(N^2\right)$ numerical complexity in time and memory. Indeed, $\mathbf{P} \in \Re^{N \times N}$ must be computed, albeit only once, since only the $\mathbf{y}_i$'s move in LD, not the $\mathbf{x}_i$'s in HD. Fortunately, $p_{i,j}$ quickly tends to 0 when the distance between the instances i and j increases. This behavior is accentuated by the fact that in HD spaces, the euclidean distance between two points increases faster. It is therefore acceptable to approximate $\mathbf{P}$ by a sparse matrix by considering only the k nearest neighbours of each instance i, other elements being really close to 0. Most of the time, the k nearest neighbor search is performed with $k = 3u$ using trees such as kd-trees or ball-trees.

Some authors, such as Linderman et al. [6], Pezzotti et al. [7] and Tang et al. [8], show that using approximate nearest neighbor search algorithms does not significantly impact the quality of the obtained visualizations. An example of such algorithm is implemented in the *Annoy* library [9], which uses random projection trees [10] to search for points in space that are close to a given query point. Computing the attractive forces can then be done efficiently in $\mathcal{O}(uN)$.

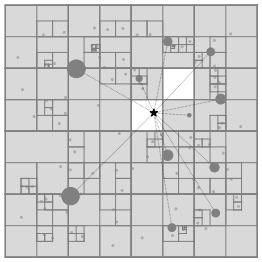
B. Computation of the Repulsive Term

There are two issues when computing the repulsive term in Equation (7). The first issue is that it requires to fully compute $\mathbf{Q}$, and the second issue is that it also requires to compute the normalizing factor Z, i.e. to compute each $d_{i,j} = \|\mathbf{y}_i - \mathbf{y}_j\|$. These two issues lead to an $\mathcal{O}\left(N^2\right)$ complexity. A method able to speed up the computation of the repulsive forces, while also building a correct approximation of Z, is thus needed. Contrary to what can be done for the attractive term, the first issue cannot be solved by only using the k nearest neighbors of each instance. Indeed, the instances in the LD space are constantly moving and it would therefore require to perform the k nearest neighbor search at each iteration, which would drastically slow down the algorithm. Furthermore, in the LD space, a Student t-distribution is used instead of a Gaussian, which means that the $q_{i,j}$ tend more slowly to 0 as the Student t-distribution has thicker tails. Therefore, only considering the k nearest neighbors may lead to greater errors.

To solve these two issues, the Barnes and Hut version of t-SNE [3] uses kd-trees to approximate the repulsive forces, while at the same time obtaining an approximation of Z. This exploits the fact that, when computing the repulsive force exerted on instance i by all other instances, groups of instances close to each other but far away from instance i can be reduced to only one fictive instance that resumes them all. These groups of instances are found using kd-trees, which are used to iteratively structure space into smaller groups. The goal of these trees is then to go from the root to the leaves and use a group when $\frac{r_{cell}}{\|\mathbf{y}_i - \mathbf{y}_{cell}\|} < \theta$, where r_{cell} is the length



(a) Computation with $\theta = 0$ (b) Computation with $\theta = 0.5$



(c) Computation with $\theta = 1.5$

Fig. 1: These figures, inspired by Heer [11], show the subdivision obtained with a kd-tree in a 2-d space. While iteratively subdividing space into smaller cells, a tree is constructed. Each node of this tree stores the number of instances in each cell and their center of mass. The black star represents the instance on which the force exerted by all other instances is computed. In Figure (a), no group is accepted, which is equivalent to a brute force computation. In Figure (b), the number of instances considered is significantly reduced, but points too close to the black star are not grouped together. In Figure (c), too many instances are grouped, which results in approximation errors.

of the diagonal of the cell that forms the considered group, y_{cell} is its center of mass and θ is a threshold parameter that balances speed and accuracy. Figure 1 illustrates the impact of θ on the approximation.

The work of Linderman et al. [6] constitutes another possibility to accelerate the computation of the repulsive term. Their algorithm, the FFT-accelerated Interpolation-based t-SNE (FIt-SNE), is based on polynomial interpolations combined with the use of FFT to speed up t-SNE. However, they do not exploit the formalism of the well-known PM algorithm in physics, whereas we will be able to use the improvements of PM developed in physics for many years. This can lead to better insights and has the potential for further improvements.

IV. Description of PM-t-SNE

This section presents the Particle-Mesh (PM) algorithm and how it can be used for the computation of the repulsive forces in t-SNE. It begins with a brief introduction to the N-body problem in electrostatics. This will allow us to (i) introduce the objective of PM and (ii) introduce the key concept of charge (or instance) distribution in space. The two following sections then describe the solution proposed by PM, from a mathematical and a numerical point of view respectively. As the key concepts of PM will be transposed to t-SNE in the last section, these sections make it possible to understand where the mathematical developments of PM-t-SNE comes from.

A. The N-body Problem in Electrostatics

The N-body problem arises in physics as soon as multiple bodies interact with each other. This happens in multiple applications such as celestial mechanics, plasma physics, fluid dynamics, electrostatics, etc. However, the N-body problem has no analytical solution as soon as $N > 2$. Furthermore, a brute force algorithm to solve the problem requires to evaluate $\frac{1}{2}N\left(N-1\right)$ interactions, which is intractable when N increases. Numerical physics has therefore been an area of active research and multiple algorithms appeared. PM is one of these algorithms used in physics to solve the N-body problem for large systems (usually $N \sim 10^5 - 10^7$).

To understand the idea behind PM, this section presents the particular case of electrostatics, but developments are similar for other interactions. We consider a set of N punctual charges at positions $\{\mathbf{r}_i\}_{i=1}^N$ in a 3-d space that exert forces on each other given their charge $\{\mathbf{q}_i\}_{i=1}^N$ according to Coulomb's law

$$\mathbf{F}\left(\mathbf{r}_i\right) = \frac{q_i}{4\pi\epsilon_0}\sum_{j=1}^{N}\frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|^3}q_j, \tag{8}$$

where ϵ_0 is the electric constant. In a more general way, if charges are distributed continuously in space, Coulomb's law can take the more general integral form

$$\mathbf{F}\left(\mathbf{r}_i\right) = \frac{q_i}{4\pi\epsilon_0}\iiint\frac{\mathbf{r}_i - \mathbf{r}'}{\|\mathbf{r}_i - \mathbf{r}'\|^3}\rho\left(\mathbf{r}'\right)d^3\mathbf{r}', \tag{9}$$

where $\rho\left(\mathbf{r}\right)$ is the charge distribution in space. This $\rho\left(\mathbf{r}\right)$, representing here the amount of charge at position $\mathbf{r}$, can be related to the number of instances at the position $\mathbf{r}$ in t-SNE.

B. Background on Particle-Mesh (PM)

Now that the N-body problem has been presented in the previous section, one classical way to solve it is by using PM. The main idea of PM is to use the electrostatic potential

$$\phi\left(\mathbf{r}\right) = \frac{1}{4\pi\epsilon_0}\iiint\frac{\rho\left(\mathbf{r}'\right)}{\|\mathbf{r} - \mathbf{r}'\|}d^3\mathbf{r}' \tag{10}$$

to compute the forces exerted on each particle, given that $\mathbf{F}\left(\mathbf{r}\right) = -q_i\nabla\phi\left(\mathbf{r}\right)$. Thus, evaluating $\phi\left(\mathbf{r}\right)$ allows us to compute $\{\mathbf{F}\left(\mathbf{r}_i\right)\}_{i=1}^N$. This way of computing forces is the key idea that we want to bring back to t-SNE. The advantage is now that the computation of $\phi\left(\mathbf{r}\right)$ involves a convolutional product that can be computed in an efficient way using Fourier Transforms (FTs). Indeed, Equation (10) can be written as

$$\phi\left(\mathbf{r}\right) = \frac{1}{4\pi\epsilon_0}\left(\rho\left(\mathbf{r}\right) * h\left(\mathbf{r}\right)\right), \tag{11}$$

where $*$ denotes a convolutional product and $h\left(\mathbf{r}\right) = \frac{1}{\|\mathbf{r}\|}$. Since $\phi\left(\mathbf{r}\right) \propto \rho\left(\mathbf{r}\right) * h\left(\mathbf{r}\right) \iff \hat{\phi}\left(\mathbf{k}\right) \propto \hat{\rho}(\mathbf{k})\hat{h}(\mathbf{k})$, where $\hat{\cdot}$

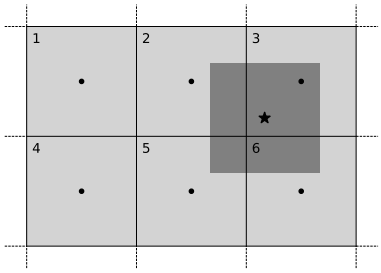


Fig. 2: Black points are grid points. The star represents a particle and the cloud surrounding it is the region of the space where the charge is uniformly distributed. With NGP, the charge is attributed to the grid point 3. With CIC, the charge is distributed between grid points 2, 3, 5 and 6, with a bigger fraction of the charge added to 3, and a smaller one to 5.

denotes the FT of the function, it is possible to trade the hard-to-compute convolutional product for a simple product by first computing $\hat{\rho}(\mathbf{k})$ and $\hat{h}(\mathbf{k})$. It is then possible to get back to $\phi(\mathbf{r})$ by taking the inverse FT of $\hat{\phi}(\mathbf{k})$. For more mathematical details, see Arfken et al. [12].

Without using FTs, $\phi(\mathbf{r})$ is directly obtained by computing the convolutional product described in Equation (11) and PM exhibits an $\mathcal{O}(M^2)$ numerical complexity, where M is the number of points in the grid used to discretize space. PM can therefore constitute an improvement regarding the brute force algorithm if the grid resolution is such that $M < N$. However, the use of FTs reduces the numerical complexity to $\mathcal{O}(M \log M)$. Since, as presented in Section V, M can be much lower than N for large datasets, this leads to a significant speed up compared to the solution of Barnes and Hut [4].

C. Discretizing Space

The previous section describes the mathematical developments of PM. Now, in order to use them from a numerical point of view, PM needs to discretize space. There are multiple ways to build a grid and obtain $\rho(\mathbf{r})$ from $\{\mathbf{y}_i\}_{i=1}^N$ in order to compute $\{\mathbf{F}(\mathbf{r}_i)\}_{i=1}^N$. The easiest way is to assign q_i to the point of the grid that is the closest to $\mathbf{y}_i$ (i.e., building a 2-dimensional histogram). This corresponds to a first-order interpolation scheme, also called Nearest Grid Point (NGP). If the resolution is low, this can bring significant errors. However, when N is large, they tend to compensate each other. Another commonly used solution is to consider a second-order interpolation scheme. In that case, the 4 points of the grid that are the closest to $\mathbf{y}_i$ get a fraction of q_i (these fractions are inversely proportional to the distance between each point and $\mathbf{y}_i$). This method, known as Cloud In Cell (CIC), is more accurate. Figure 2 illustrates how these two methods work.

Once the potential is computed for each point of the grid, its gradient is computed using a first- or second-order method depending on the interpolation method used to build $\rho(\mathbf{r})$. Indeed, it does not make sense to use a second-order method to evaluate $\nabla\phi(\mathbf{r})$ if $\rho(\mathbf{r})$ is estimated using a first-order approximation. The same kind of reasoning applies when it comes to evaluating the force exerted on each particle.

D. Closing the Gap Between t-SNE and PM

Now that PM has been presented, this section aims to transpose it to t-SNE. As the attractive force in t-SNE can be computed efficiently (see Section III-A), this section focuses on the computation of the more problematic repulsive term in Equation (7). It seems already similar to Equation (8) if we suppose that $q_i = 1, \forall i$. This means that there is no privileged instance in the dataset. Inspired by PM, this force can then be computed by discretizing space, i.e. by using a grid such that

$$\mathbf{F}_i^{rep} = \mathbf{F}^{rep}(\mathbf{y}_i) = \frac{1}{Z}\sum_{g=1}^{M} \rho_g \frac{\mathbf{y}_i - \mathbf{y}_g}{\left(1 + \|\mathbf{y}_i - \mathbf{y}_g\|^2\right)^2}, \tag{12}$$

where g is a super-index that runs through the M points of the grid, and ρ_g is the number of instances assigned to the g^{th} grid's point. When $M \to \infty$ in Equation (12), the finite summation becomes an integral and the equation becomes

$$\mathbf{F}^{rep}(\mathbf{y}_i) = \frac{1}{Z}\iint \rho(\mathbf{r}') \frac{\mathbf{y}_i - \mathbf{r}'}{\left(1 + \|\mathbf{y}_i - \mathbf{r}'\|^2\right)^2} d^2\mathbf{r}'. \tag{13}$$

To be able to use PM for t-SNE, a potential such that $-\nabla\phi(\mathbf{r}) \propto \mathbf{F}^{rep}(\mathbf{r})$ is needed. If it is expressed as

$$\phi(\mathbf{r}) = \rho(\mathbf{r}) * w(\mathbf{r}), \tag{14}$$

where $w(\mathbf{r}) = \frac{1}{1+\|\mathbf{r}\|^2}$, one can show that

$$\mathbf{F}^{rep}(\mathbf{y}_i) = \frac{-1}{2Z}\nabla\phi(\mathbf{r})\,|_{\mathbf{r}=\mathbf{y}_i}. \tag{15}$$

Indeed,

$$\begin{aligned}
\frac{-1}{2Z}\nabla\phi(\mathbf{r}) &= \frac{-1}{2Z}\nabla\left(\rho(\mathbf{r}) * \frac{1}{1+\|\mathbf{r}\|^2}\right) \\
&= \frac{-1}{2Z}\iint \rho(\mathbf{r}')\,\nabla_r \frac{1}{1+\|\mathbf{r}-\mathbf{r}'\|^2} d^2\mathbf{r}' \\
&\overset{1}{=} \frac{1}{Z}\iint \rho(\mathbf{r}') \frac{\mathbf{r}-\mathbf{r}'}{\left(1+\|\mathbf{r}-\mathbf{r}'\|^2\right)^2} d^2\mathbf{r}' \\
&= \mathbf{F}^{rep}(\mathbf{r}).
\end{aligned} \tag{16}$$

Equation (14) involves a convolutional product that can be computed efficiently using FTs. The gap between t-SNE and PM is then almost closed, only Z still needs to be computed. Fortunately, the expression

$$\phi(\mathbf{r}) = \iint \rho(\mathbf{r}') \frac{1}{1+\|\mathbf{r}-\mathbf{r}'\|^2} d^2\mathbf{r}' \tag{17}$$

can be used to compute Z, as it is linked to $\phi(\mathbf{r})$ by,

$$Z = \sum_k \phi(\mathbf{y}_k), \tag{18}$$

[1] given that $\nabla\frac{1}{1+\|\mathbf{r}\|^2} = \frac{\partial\frac{1}{1+\|\mathbf{r}\|^2}}{\partial\|\mathbf{r}\|}\frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{-2\mathbf{r}}{\left(1+\|\mathbf{r}\|^2\right)^2}$.

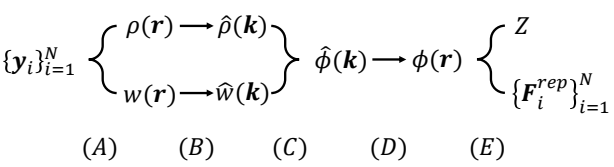


Fig. 3: Scheme for the computation of the repulsive term. Step (A) discretizes space and evaluates ρ and w. Step (B) consists of using FTs in order to compute ϕ efficiently. Step (C) is a simple product between $\hat{\rho}$ and $\hat{w}$. Step (D) consists of using inverse FTs to obtain ϕ and step (E) consists of using it to get both the repulsive forces and Z with Equations (15) and (18).

since, by considering a grid of infinite resolution,

$$\begin{aligned} Z &= \sum_k \sum_{l \neq k} \frac{1}{1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2} \\ &= \sum_k \sum_{g=1}^{M} \rho_g \frac{1}{1 + \|\mathbf{y}_k - \mathbf{y}_g\|^2} \\ &= \sum_k \iint \rho(\mathbf{r}') \frac{1}{1 + \|\mathbf{y}_k - \mathbf{r}'\|^2} d^2\mathbf{r}', \end{aligned} \tag{19}$$

where g is a super-index that runs through the M grid's points.

Computing the potential hence allows us to compute the repulsive term completely. Since the potential is expressed as a convolutional product, it can be computed efficiently using FTs and inverse FTs. Figure 3 sums up the scheme for the computation of the repulsive term.

V. Comparative Study

This section carries out a comparative study to assess the quality of the visualizations obtained for a given computation time, and for different t-SNE algorithms. Several datasets of different sizes are used to make this comparative study.

A. Experimental Datasets

For the comparative study, experiments are performed on multiple datasets of different sizes. For a clear comparison, datasets and preprocessing steps are close to those of van der Maaten [3]: (1) the CIFAR-10 dataset of Krizhevsky [13], (2) the MNIST dataset of LeCun and Cortes [14], (3) the SVHN dataset of Netzer et al. [15] and (4) the TIMIT dataset of Garofolo et al. [16]. This section briefly presents them. For each one of them, PCA was performed after preprocessing to reduce them to 50 dimensions. This dimension for the HD embedding is a classical choice proposed by van der Maaten [3], and is used because t-SNE performs less well when the initial dimension is too high.

CIFAR-10 consists of $60,000$ tiny 32×32 images with 3 channels. There are 10 classes such as airplane, automobile, bird, cat, etc. with $6,000$ instances each. As $32 \times 32 \times 3$ dimensions is too high for t-SNE, we used a Convolutional Neural Network (CNN) to extract a 512-d embedding.

MNIST consists of $70,000$ 28×28 greyscale images of 10 handwritten digits evenly represented in the dataset.

(a) DSC=60.15%, DC=77.34% (b) DSC=88.81%, DC=93.99%

Fig. 4: Two visualizations obtained with PM-t-SNE on SVHN.

SVHN consists of $630,420$ real-world 32×32 color images obtained from house numbers in Google Street View. Each image is labeled as a single digit. Again, a CNN is used to extract a 64-d embedding.

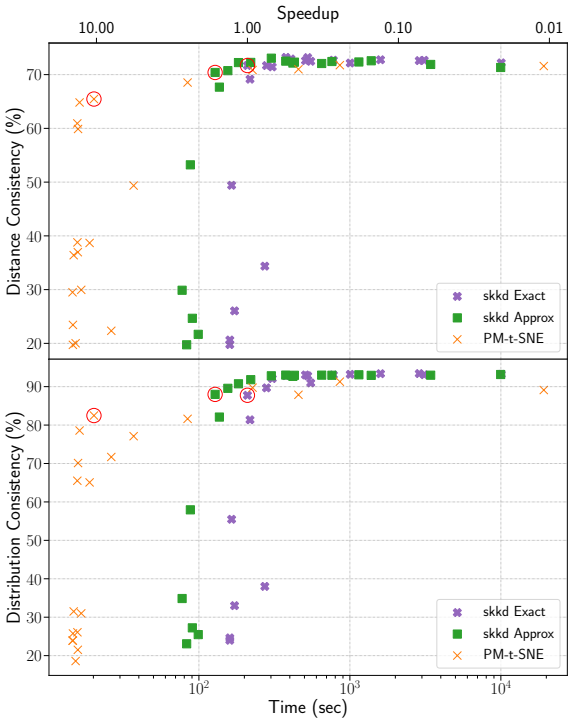
TIMIT contains $3,696$ spoken utterance for a total of $1,105,455$ frames labelled with the 39 phonemes. We used 13 mel-frequency cepstral coefficients (MFCC), delta features and delta-delta features to obtain a 273-d embedding.

Contrary to CIFAR-10 and MNIST, SVHN and TIMIT are more complex real-world datasets where classes are not evenly represented. In SVHN, some images may be of bad quality, and in TIMIT, data is so complex that it is difficult to visualize.

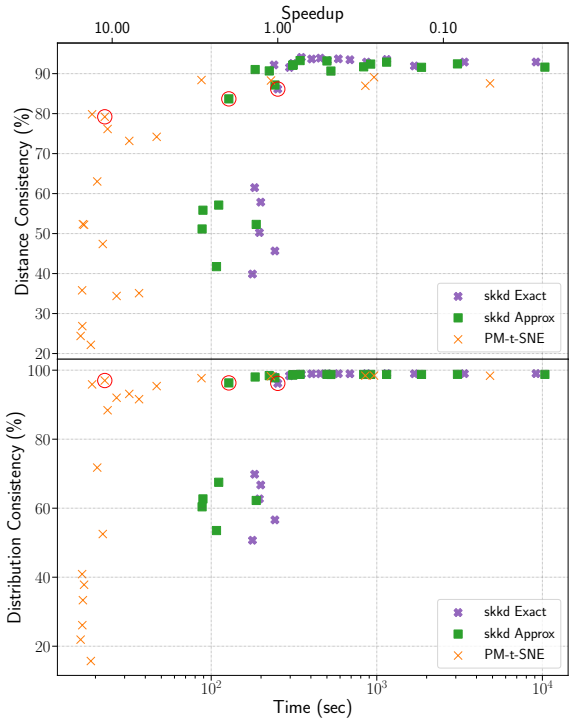
B. Experimental Setup

In this comparative study, 3 different algorithms are used. The first one, called skkd Exact for this study, comes from *scikit-learn* [17] and uses the Barnes and Hut approximation explained in Section III-B. The second one, that we called skkd Approx, is almost the same, but the approximated nearest neighbor search is performed using *Annoy* instead of kd-trees. The last one, PM-t-SNE, is our implementation that uses a Cloud In Cell interpolation method and the same approximated nearest neighbor search than skkd Approx.

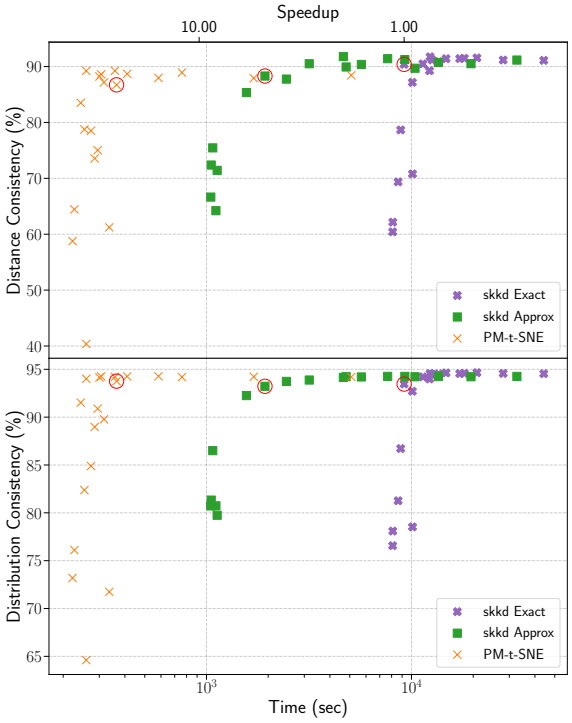
For the 3 algorithms, a similar experimental setup as the one described by van der Maaten [3] is used, including the use of early exaggeration in the first few hundreds steps. The number of iterations for GD is fixed to 750. The learning rate λ is also set to 750. The perplexity u is set to 30, and the number of nearest neighbors to consider for the computation of the attractive term is set to $1.5u$, as no significant loss in quality was observed compared to using $3u$ nearest neighbors. The early exaggeration is performed during the first 200 iterations with $\alpha = 12$. The momentum $\eta(k)$ is initially set to 0.8 and decays exponentially. For the *Annoy* library, the number of trees to use is set to 10 as it is the best value according to all our preliminary experiments. For skkd Exact and skkd Approx, 20 different values between 0.1 and 8 are tested for the meta-parameter θ (called *angle* in *scikit-learn*), except for SVHN and TIMIT where it was too long to test them with $\theta = 0.1$. For PM-t-SNE, 20 different values for the number of grid points M between 4^2 and $4,096^2$ in a log-scale are tested.



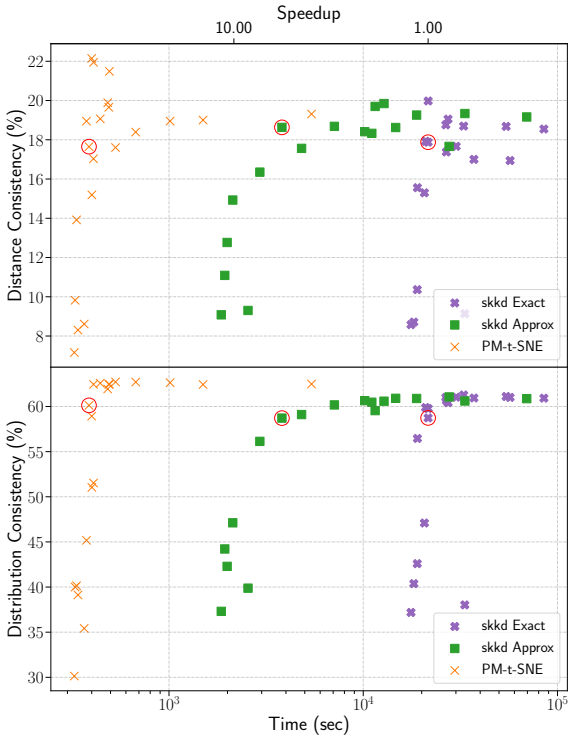
(a) CIFAR-10



(b) MNIST



(c) SVHN



(d) TIMIT

Fig. 5: Results obtained on the CIFAR-10, MNIST, SVHN and TIMIT datasets. Notice that $N_{\text{CIFAR-10}} < N_{\text{MNIST}} < N_{\text{SVHN}} < N_{\text{TIMIT}}$. Red circles point out points in the elbows, which is discussed in Section VI. The speedup axis represents the ratio $\frac{\tau}{t}$ where t is the computation time, and τ is a reference time arbitrarily chosen as the red circle for the skkd Exact method. Note that the brute t-SNE implementation is not used as a baseline since it takes too long to run. For PM-t-SNE, the meta-parameter M increases from left to right. For skkd Exact and skkd Approx, the meta-parameter θ decreases from left to right. The low DSC obtained by all methods for TIMIT can be explained by the complexity of visualizing this dataset of audio recordings.

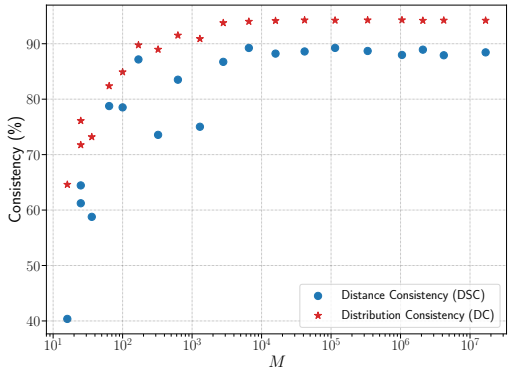
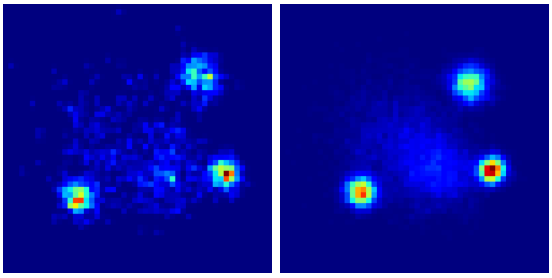


Fig. 6: DSC and DC obtained as a function of M for the SVHN dataset. Results are similar for the other datasets.



(a) For 10^3 instances (b) For 10^4 instances

Fig. 7: These figures show the density of instances obtained using a Cloud In Cell interpolation method on a 50×50 grid.

When the size of the dataset N increases, the time needed for the computation of the attractive term also increases as it exhibits a numerical complexity of $\mathcal{O}(uN)$. However, the computation of this term can be easily parallelized. Therefore, for the 3 algorithms and each run, the computation of the attractive term was spread between 8 cores of a 32-core AMD Epyc 7551P CPU clocked at 2 GHz. On the other side, only one core was used for the computation of the repulsive term.

The distance consistency (DSC) and distribution consistency (DC) introduced by Sips et al. [18] are used to assess the quality of the visualizations. This choice is motivated by the work of Sedlmair et al. [19], which shows that these metrics are among the best to assess the quality of a visualization. To illustrate this, Figure 4 presents two visualizations with their DSC and DC scores. DSC is defined as the proportion of points in the LD space that are closer to the centroid of their class than to another one. DC aims to extend DSC to more general spacial distributions. Pixels are considered instead of clusters, and a measure of entropy in each pixel is used to evaluate the quality. If n points of any class overlap on a particular pixel, the associated entropy for that pixel is $-\sum_c \frac{n_c}{n} \log_2\left(\frac{n_c}{n}\right)$, where n_c is the number of points of class c overlapping in the pixel. If $n_c = n$, the entropy is equal to 0 and the consistency is maximal. When datasets contain a large number of instances, a lot of points overlap each other in the visualization and pixel entropy increases. Therefore, considering pixels to assess the quality is particularly meaningful. For our study, images of our visualizations with 300×300 pixels were used to evaluate the DC. As the visualizations obtained from t-SNE can vary w.r.t. the initialization of $\mathcal{Q}$, 8 visualizations are used to compute the mean quality.

C. Results

Results on the different datasets are presented in Figure 5. Note that DSC and DC are built such that a larger percentage reflects a better visualization. For PM-t-SNE, points from the left to the right are obtained by increasing the grid resolution (M). For skkd Exact and skkd Approx, θ decreases for points from the left to the right. When increasing M or reducing θ, the computation time required to obtain the visualization increases, but quality also increases until reaching a plateau. When the dataset size is larger, the difference in time between the 3 algorithms for a visualization of a given quality increases (the gap between the elbows increases).

Figure 6 shows how the quality of the visualizations changes with M for PM-t-SNE. Only the results for SVHN are presented, but those for the other datasets are similar.

VI. Discussions

In Figure 5, all the curves contain an elbow. The optimal choice in a particular curve, for the meta-parameters M or θ, is the one that leads to a good balance between visualization quality and computation time. These optimal choices are represented by points in the elbows, and are highlighted with red circles. The computation time for these particular points can be used to characterize how efficient algorithms are. For small datasets, the curves obtained with PM-t-SNE present more fluctuations and this choice of reference point may then be discussed. Nevertheless, PM-t-SNE aims to be applied on large datasets, where these fluctuations tend to disappear.

By looking at these points in the elbows, we can see that PM-t-SNE is able to produce visualizations with a quality similar to the two others. We can also see that PM-t-SNE performs better for datasets with a larger number of instances, i.e. a smaller computation time is required by PM-t-SNE in order to obtain visualizations of similar quality.

For the CIFAR-10 dataset, PM-t-SNE allows us to obtain visualizations with a smaller computation time but at the cost of a small loss in quality. As we will see with other datasets, this is due to the fact that the number of instances in CIFAR-10 is too low for PM-t-SNE. Notice also that using an approximated nearest neighbors search does not lead to a significant speed up as the computation time for skkd Exact is really close to the one of skkd Approx. Yet, it does not lead to a decrease in quality either. For the highlighted points, the value for M is $6,561$ and for θ it is 1.8.

For MNIST, conclusions are similar to those for CIFAR-10. For the red points, M is $6,561$, and θ it is 1.8.

For the SVHN dataset, PM-t-SNE is ~ 9 times faster than skkd Approx and the loss in quality almost vanished. The interest of using an approximated nearest neighbors clearly appears as skkd Exact is nearly one order of magnitude slower

than skkd Approx. For the highlighted points, the value for M is now $2,809$, which is really low compared to the number of instances in the dataset, which is $630,420$. The value for θ for the two other algorithms is still 1.8. For such big datasets, PM-t-SNE is quite efficient, as it is faster and it allows us to build visualizations of equivalent qualities.

Finally, for the TIMIT dataset, PM-t-SNE is even faster as it is now ~ 12 times faster than skkd Approx and ~ 18 times faster than skkd Exact. In terms of DC, PM-t-SNE slightly overtakes other implementations. For the highlighted points, the value for M is $1,296$ points, and for θ it is still 1.8.

In conclusion, it is always possible to build a grid such that PM-t-SNE is faster than an implementation based on the Barnes and Hut approximation. In order to obtain visualizations of good quality from PM-t-SNE, it requires a large number of instances. This makes sense as it relies on the discretization of the distribution of instances in space (i.e., a 2-d histogram). This discretization leads then to errors as points are moved on the grid. If there are a small number of instances, this 2-d histogram is not good enough to reflect the positions of the instances. For a larger number of instances (typically, $\sim 10^5 - 10^6$ or even more), these errors tend to compensate for each other and the 2-d histogram is better at describing the disposition of instances in space. Figure 7 illustrates this. For such datasets, a grid between 50×50 and 100×100 can already be sufficient even if it leads to $M \lll N$.

VII. Conclusion

This work presents a new formalism for t-SNE called PM-t-SNE, based on the well-known PM algorithm used to solve the N-body problem in physics. Combined with the use of FFT, it allows us to obtain a significant speed up. We found that this method is well-suited for large datasets ($N \sim 10^5 - 10^6$, or even more). In that case, PM-t-SNE overtakes other implementations based on the Barnes and Hut approximation in terms of computation time, while maintaining the quality of the visualizations. From a numerical point of view, PM-t-SNE exhibits a complexity of $\mathcal{O}(M \log M)$, whereas Barnes and Hut implementations exhibit a complexity of $\sim \mathcal{O}(N \log N)$. For large datasets, $M \lll N$ and this leads to a significant reduction of the numerical complexity.

The PM algorithm has been used in physics for decades, and has been improved several times. Now that PM can be used to perform t-SNE in a more efficient way, more sophisticated versions of PM could be transposed as well. As an example, we only considered equispaced grids in this work, but Splinter [20] shows that it is possible to use a system of nested grids. In this way, it is possible to increase the grid's resolution in regions where the density of instances is higher, and decrease it in less dense regions. It would also be interesting to parallelize the computation of the repulsive term, as it is not as trivial as for the attractive term, and also consider other interpolation methods. Finally, it could be interesting to consider an hybrid algorithm that uses both the PM methods and the Barnes and Hut approximation, as it is already done for the N-body problem in physics by Bode and Ostriker [21].

Acknowledgments

A.M. is funded by the Fund for Scientific Research (F.R.S.-FNRS) of Belgium. V.D. was supported by the EOS VeriLearn project n. 30992574 and benefits from the support of the Walloon region with a Ph.D. grant from FRIA (F.R.S.-FNRS). This research used resources of PTCI at UNamur, supported by the F.R.S.-FNRS under the convention n. 2.5020.11. The authors thank Jérôme Fink and Géraldin Nanfack for their comments and the fruitful discussions on this paper.

References

[1] G. E. Hinton and S. Roweis, "Stochastic Neighbor Embedding," in *Proceedings of NeurIPS*, vol. 15, 2003, pp. 857–864.

[2] L. van der Maaten and G. E. Hinton, "Visualizing Data using t-SNE," *JMLR*, vol. 9, pp. 2579–2605, 2008.

[3] L. van der Maaten, "Accelerating t-SNE using Tree-Based Algorithms," *JMLR*, vol. 15, no. 93, pp. 3221–3245, 2014.

[4] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, 1986.

[5] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*. A. Hilger, 1988.

[6] G. Linderman, M. Rachh, J. Hoskins, S. Steinerberger, and Y. Kluger, "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data," *Nat Methods*, vol. 16, no. 3, pp. 243–245, 2019.

[7] N. Pezzotti, B. P. F. Lelieveldt, L. van der Maaten, T. Hollt, E. Eisemann, and A. Vilanova, "Approximated and User Steerable tSNE for Progressive Visual Analytics," *IEEE TVCG*, vol. 23, no. 7, pp. 1739–1752, 2017.

[8] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing Large-scale and High-dimensional Data," 2016, pp. 287–297.

[9] E. Bernhardsson, "Annoy: Approximate Nearest Neighbors in C++/Python," https://pypi.org/project/annoy, 2018.

[10] S. Dasgupta and Y. Freund, "Random projection trees and low dimensional manifolds," in *Proceedings of the ACM STOC*, Victoria, British Columbia, Canada, 2008, p. 537.

[11] J. Heer, "The Barnes-Hut Approximation: Efficient computation of N-body forces," https://jheer.github.io/barnes-hut/, 2021, Accessed on 2021-01-05.

[12] G. B. Arfken, H.-J. Weber, and F. E. Harris, *Mathematical methods for physicists: a comprehensive guide*, 7th ed. Elsevier, 2013.

[13] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Master's thesis, University of Toronto, 2009.

[14] Y. LeCun and C. Cortes, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, 2010, Accessed on 2016-01-14.

[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[16] J. S. Garofolo, L. F. Lamel, W. M. Fisher, D. S. Pallett, N. L. Dahlgren, V. Zue, and J. G. Fiscus, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," https://catalog.ldc.upenn.edu/LDC93S1, Accessed on 2020-12-23.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *JMLR*, vol. 12, no. 85, pp. 2825–2830, 2011.

[18] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan, "Selecting good views of high-dimensional data using class consistency," *Computer Graphics Forum*, vol. 28, no. 3, pp. 831–838, 2009.

[19] M. Sedlmair and M. Aupetit, "Data-driven Evaluation of Visual Quality Measures," *Computer Graphics Forum*, vol. 34, no. 3, pp. 201–210, 2015.

[20] R. J. Splinter, "A nested-grid particle-mesh code for high-resolution simulations of gravitational instability in cosmology," *MNRAS*, vol. 281, no. 1, pp. 281–293, 1996.

[21] P. Bode and J. P. Ostriker, "Tree Particle-Mesh: An Adaptive, Efficient, and Parallel Code for Collisionless Cosmological Simulation," *The ApJS*, vol. 145, no. 1, pp. 1–13, 2003.