World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:15, No:8, 2021

Index t-SNE: Tracking Dynamics of High-Dimensional Datasets with Coherent Embeddings

G. Candel, D. Naccache

Open Science Index, Computer and Systems Engineering Vol:15, No:8, 2021 publications.waset.org/10012177.pdf

Abstract—t-SNE is an embedding method that the data science community has widely used. It helps two main tasks: to display results by coloring items according to the item class or feature value; and for forensic, giving a first overview of the dataset distribution. Two interesting characteristics of t-SNE are the structure preservation property and the answer to the crowding problem, where all neighbors in high dimensional space cannot be represented correctly in low dimensional space. t-SNE preserves the local neighborhood, and similar items are nicely spaced by adjusting to the local density. These two characteristics produce a meaningful representation, where the cluster area is proportional to its size in number, and relationships between clusters are materialized by closeness on the embedding. This algorithm is non-parametric. The transformation from a high to low dimensional space is described but not learned. Two initializations of the algorithm would lead to two different embedding. In a forensic approach, analysts would like to compare two or more datasets using their embedding. A naive approach would be to embed all datasets together. However, this process is costly as the complexity of t-SNE is quadratic, and would be infeasible for too many datasets. Another approach would be to learn a parametric model over an embedding built with a subset of data. While this approach is highly scalable, points could be mapped at the same exact position, making them indistinguishable. This type of model would be unable to adapt to new outliers nor concept drift. This paper presents a methodology to reuse an embedding to create a new one, where cluster positions are preserved. The optimization process minimizes two costs, one relative to the embedding shape and the second relative to the support embedding' match. The embedding with the support process can be repeated more than once, with the newly obtained embedding. The successive embedding can be used to study the impact of one variable over the dataset distribution or monitor changes over time. This method has the same complexity as t-SNE per embedding, and memory requirements are only doubled. For a dataset of n elements sorted and split into k subsets, the total embedding complexity would be reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^2/k)$, and the memory requirement from n^2 to $2(n/k)^2$ which enables computation on recent laptops. The method showed promising results on a real-world dataset, allowing to observe the birth, evolution and death of clusters. The proposed approach facilitates identifying significant trends and changes, which empowers the monitoring high dimensional datasets' dynamics.

Keywords—Concept drift, data visualization, dimension reduction, embedding, monitoring, reusability, t-SNE, unsupervised learning.

I. Introduction

HIGH dimensional datasets are very rich sources of information. Because of their wideness, they are difficult to investigate, process, and represent in a simple manner. An appropriate reduction of width would benefit from:

- Storage cost reduction;
- Computational cost reduction;
- Denoising / Information compression;
- Synthetic data visualization.

For a dataset $X \in \mathbb{R}^{n\times d_0}$, with n elements and d_0 dimensions, a reduction into $Y \in \mathbb{R}^{n\times d_1}$ offers a reduction of the memory footprint of $100(1-\frac{d_1}{d_0})$ %, which is non-negligible for large datasets. For linear algorithms, the computational cost is reduced by the same factor. The core information in those wide datasets is hard to identify because of the large number of features and correlation, and redundancy. The use of methods that condense the information enables to obtain a synthetic view of the dataset, which would benefit post-processing algorithms or data analysts' work. The synthetic view can also be used to display results by coloring items according to their predicted value.

A dimension reduction can be performed following different approaches:

- Automatic feature selection;
- Human engineered feature;
- Automatic feature extraction;

Automatic feature selection selects some features among all available according to some characteristics. The feature importance can be evaluated using Shapley values [1] or removing redundant features [2]. These approaches are relatively straightforward to put into practice. Nonetheless, some information may be lost if the signal is too weak, or if the number of selected features is too small.

Feature engineering allows shaping human knowledge into an algorithmic form. This approach is practical when a minimal amount of data is available, preventing the use of automatic algorithms. It is necessary in specific cases to transform human data into processable information. Dates are a good example: a computer cannot understand directly that $\mathbf{x} = [28, 2]$ and $\mathbf{y} = [1, 3]$ represent dates and that $\|\mathbf{x} - \mathbf{y}\| = 1$ or $2\}$ days, depending on this is a leap year or not. Apart from this example with an exact answer, there is no guarantee if the transformation would help or prevent later algorithms from performing good predictions. As these features are not learned from data, they are likely to be stable, i.e., not tricked by outliers which could perturb the learning. They offer some form of *explainability*, as the engineer can describe the meaning of the transformation in

Gaëlle Candel and David Naccache are with the Département d'informatique de l'ENS, ENS, CNRS, PSL University, Paris (France) and with Wordline Labs, Paris (France) (e-mail: gaelle.candel@ens.fr, david.naccache@ens.fr).

the human language. In the absence of previous knowledge, their development is costly in operating time, and on a large dataset, an expert may miss some essential features.

Automatic feature extraction replaces handcrafted features with algorithm-learned features. This is a broad research area, with statistical compression methods such as Principal Component Analysis [3], or methods based on deep neural networks, like autoencoders [4], [5].

Automatic feature extraction can be decomposed into two categories, based on reusability on new data:

- Parametric methods;
- Non-parametric methods.

Parametric methods, such as PCA [3], Self-Organizing Maps [6], learn a mapping function $f : X \rightarrow Y$, which minimize a quantity of the form $\arg_f \min Cost(X, f(X))$. When learned, the function f can be reused on any new input X'. The inference time of most of these methods is linear, such as $f(X) = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_n)]$. For a large input X, the computation can be distributed on several machines, which enhance scalability to large datasets. In contrast, non-parametric methods minimize a specific quantity $\min_Y Cost(X, Y)$ directly by optimizing Y's values. No function is learned, which prevents the reusability of a previous computation for new data. This is the case of ISOMAP [7], UMAP [8] or t-SNE [9]. Despite the non-reusability, the two most recent methods, t-SNE and UMAP, have been extensively used by the machine learning community. Those methods have been successful at representing high dimensional datasets, by adapting to disparate scaling and non-homogeneous densities while letting appear clustering structures.

The strength of t-SNE comes from its ability to deal with heterogeneous scaling and the crowding problem. In high dimensional space, an item may have many neighbors around, all distant from each other. By reducing the number of dimensions, it is impossible to preserve the distance between an item and its neighbors and between the neighbors. If distance to the item is preserved, neighbours' distance will decrease, making them closer than in high dimensional space. This corresponds to the crowding problem.

Instead of preserving all the distances, t-SNE preserves it locally. The algorithm adapts for each item to the local density, taking into account a small group of neighbors. This local adaptation makes the power of t-SNE, as the points in a very dense cluster are separated from each other. Consequently, the number of items in a cluster is proportional to its visual area on the embedding, which helps an analyst look at the dataset composition. The other impact of the local adaptation is on the ability to deal with heterogeneous data scaling. All items fit together on the same visual space regardless their initial distance to the dataset's mean. These characteristics lead to embeddings with excellent visual qualities.

Despite the high quality of the obtained embeddings, t-SNE is a non-parametric method, where no function is learned. The outcome of running t-SNE twice with the same input X leads to two different embeddings $Y^{(0)}$ and $Y^{(1)}$, with no equivalence between the positions. This is due to the initialization, which starts with randomly generated vectors.

The initialisation process can be controlled to improve determinism. Many works such as [10] proposed to initialize the embedding with PCA coefficients. Starting with these positions improves the repeatability, but does not ensure the regeneration of large scale structures for different datasets, as t-SNE preserves local neighborhood only. The work of [10] proposes a method to create large scale structures using two t-SNE steps, which enables to obtain embeddings with large and low scale similarities. The algorithm starts with the PCA coefficients as initial item positions, followed by a t-SNE step adjusted to take into account far-range neighborhood. These positions are reused by another t-SNE step, taking into account small-range neighborhood, letting appear a finer structure. This work was successful for visual analytics, allowing the preservation of relative cluster positions over multiple datasets. The embeddings are visually similar, but the preservation of cluster positions is not exact, preventing the use of the same algorithm on all embeddings.

A naive approach to compare two datasets using their embedding is to compute the joint embedding over the consolidated dataset $X'' = [X, X']$. There are two limitations to this approach. The first one concerns the computational cost, as the complexity of t-SNE is in $\mathcal{O}(n^2)$ for the worst case. The work of [11] proposes an approximation of the different forces, claiming a linear complexity. Nonetheless, the second limitation concerns the data availability. If the two datasets are available now, but a third would arrive later, the embedding corresponding to $[X^{(0)}, X^{(1)}]$ would not share spatial correspondences with the embedding obtained with $[X^{(1)}, X^{(2)}]$.

To obtain consistency in the item positioning, several works [12], [13] proposed the use of deep neural networks to mimic the behavior of t-SNE. As with any trained algorithm with no memory nor update mechanism, the inference results is purely deterministic. The algorithm would be able to map correctly a dataset with a distribution similar to the training dataset. However, for a dataset with a different distribution, the model would not adapt to the new density, leading to overcrowded and/or depleted areas. The neural network would not be able to adapt to concept drift, nor as new outliers as these models' generalizability is limited to their training set.

Instead of learning how to make an embedding, LION t-SNE [14] proposed an answer to *where new points should be put*, taking into account the points already present and the empty areas left. New points are positioned nearby their nearest neighbors without moving existing items from their location. Items that do not have relevant neighbors in the input space are positioned on an empty area of the embedding, filled later with more relevant neighbors. This approach allows adding a few points on the previous embedding, keeping the possible visual quality of the embedding. While this work deals correctly with item's addition, normal or outlier, it does not deal with update nor deletion. Last point concerns the scalability. The addition of a few points is likely to preserve the general aspect of the embedding. However, the shape of the resulting embedding after a massive addition of items is unknown.

Last work to mention is Dynamic t-SNE [15] which updates

Open Science Index, Computer and Systems Engineering Vol:15, No:8, 2021 publications.waset.org/10012177.pdf

the embedding $Y^{(t)}$ into $Y^{(t+1)}$ after a change from $X^{(t)}$ into $X^{(t+1)}$. It assumes that there is a one-to-one correspondence between items of $X^{(t)}$ and $X^{(t+1)}$. This setup corresponds to a monitoring situation, where the data coming from a fixed number of sensors arrives at each time step. To compute $Y^{(t+1)}$, dt-SNE starts with the previous embedding positions $Y^{(t)}$, and tries to minimize the cost defined by t-SNE relatively to $X^{(t+1)}$. A penalty is added on the displacement of $Y^{(t+1)}$ from the initial position, which enables to keep the embedding coherent over multiple time steps. While this work addresses the updatability, as no items can be added nor deleted, the usability is restricted to particular use-cases, such as multidimensional time series.

In this paper, we abort the problem of the reusability of a t-SNE embedding. The proposed approach is inspired by dt-SNE, concerning the idea of using the previous embedding as a support to the new embedding. The support embedding is not used to initialize the new item positions but to guide them towards neighbors location. Compared to dt-SNE, the scope is broadened because there is no constraint on the integrated elements, nor on their number or distribution. By enlarging to the addition, updating and deleting items, our method can be used in many more real-world monitoring situations, such as when some sensors are added to the system or removed due to failure. The approach is not limited to the temporal dataset, but to any *index* variable, a discrete or continuous variable, such as temperature or speed. Dataset can be sorted according to this variable, and successive embeddings can be issued to track the impact of the *index* variable over the data distribution. In other words, it allows to obtain embedding conditional to the index variable of interest.nThe method is called *index* t-SNE, abbreviated it-SNE, for this reason.

In the first section, the main equations governing t-SNE optimization process are introduced. This section is followed by the description of it-SNE, reusing part of the initial t-SNE scheme. Then, the methods section describe the different datasets and evaluation metrics, followed by the experimental results. Last, this article finishes with a discussion followed by its conclusion.

II. t-SNE Formulation

t-SNE [9] is a structure-preserving embedding algorithm trying to preserve the local neighborhood of items in a low dimensional space. Given a dataset $X \in \mathbb{R}^{(n \times d)}$, of n items lying in a d dimensional space, the goal is to generate its corresponding embedding $Y \in \mathbb{R}^{(n \times d_e)}$:

$$Y \leftarrow t\text{-SNE}(X; d_e, perp)$$

where d_e is the number of embedding dimensions, often set to 2, and $perp$ is the perplexity parameter. Two items i and j neighbors in X must be neighbors in Y. The definition of *neighbors* depends of two things: the local density around an item, and the user-defined perplexity parameter which represents the average number of neighbors to consider. Rather than reasoning in terms of distances, the algorithm uses probabilities, computed from pairwise distances, to optimize Y.

A. Interaction Probability

t-SNE tries to adapt the embedding vector Y to X using their respective probability matrices P and Q, both of dimension $n \times n$. These probabilities represent the degree of relatedness of two items in their respective space. A large probability corresponds to a high proximity, while a smaller to a large distance. The input and output probability matrices are computed differently to create a small asymmetry.

1) Input Probability Matrix: The conditional probability of item j with respect to i is defined as:

$$p_{j|i} = \frac{1}{Z_i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right) \tag{1}$$

By convention, $p_{i|i} = 0$ as an item does not interact with itself, and $Z_i = \sum_{j \neq i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)$ is the normalization constant of item i, such as $\sum_j p_{j|i} = 1$.

The standard deviation parameter σ_i adapts the kernel range to the local density around item i. The optimal value of σ_i is obtained by binary search to match the *perplexity*. The perplexity is a user-defined parameter that represents the average number of neighbors of an item, defined formally as:

$$Perp(P_i) = 2^{H(P_i)}$$

where $H(P_i)$ is the Shannon entropy calculated as:

$$H(P_i) = -\sum_{j \neq i} p_{j|i} \log_2(p_{j|i})$$

The joint probability between i and j is defined as $p_{i,j} = \frac{p_{i|j} + p_{j|i}}{2n}$. These equations enable the computation of the symmetric probability matrix P given a particular dataset X and a perplexity target.

2) Output Probability Matrix: The output probability matrix Q is obtained in a similar manner using the embedding vector Y. As the goal is to obtain homogeneous distances between neighbors, there is no adaptation to local neighborhood. Another difference concerns the kernel choice. Instead of an exponential kernel, a t-student kernel with one degree of freedom is used. This kernel asymmetry allows modifying the long-range interactions, which leads to repulsive forces between non-neighbors items.

The joint probability between item i and j is calculated as:

$$q_{i,j} = \frac{1}{V}(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} \tag{2}$$

with $q_{i,i} = 0$ and $V = \sum_{k \neq \ell}(1 + \|\mathbf{y}_k - \mathbf{y}_\ell\|^2)^{-1}$ the global normalization constant, which lead to $\sum_{i,j} q_{i,j} = 1$.

B. Cost Minimization

The dissimilarity between the two probability matrices P and Q is measured using the Kullback-Leibler divergence:

$$KL(P\|Q) = \sum_i C_i = \sum_i \sum_{j \neq i} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} \tag{3}$$

where C_i the cost associated to item i.

By deriving (3), a simple form of the gradient is obtained:

$$\frac{\partial C_i}{\partial y_i} = 4\sum_j (p_{i,j} - q_{i,j})\frac{\mathbf{y}_i - \mathbf{y}_j}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2} \tag{4}$$

The algorithm uses the gradient descent approach to minimize the cost by updating the initial $Y(0)$ solution:

$$\begin{aligned} Y(t) = \quad & Y(t-1) - \alpha(t)\tfrac{\partial C}{\partial Y}(t) \\ & +\eta(t)\left(Y(t-1) - Y(t-2)\right) \end{aligned} \tag{5}$$

with $\alpha(t)$ the learning rate, adjusted over time, and $\eta(t)$ the momentum rate. The matrix Q is recomputed at each update step according to the newly obtained $Y(t)$, while the matrix P is not as the input vector is left unchanged. The computation of Q at each step is the most costly operation, which leads to a complexity in $\mathcal{O}(n^2)$ without optimization.

In the original paper [9], many optimization tricks are used. For instance, *early exaggeration* replaces temporary $(p_{i,j} - q_{i,j})$ by $(kp_{i,j} - q_{i,j})$, where $k > 1$. This trick amplifies the attraction forces between nearest neighbors, which fasten the formation of separated clusters. After some steps, the factor is set back to $k = 1$ which lets nearest neighbors to separate from each others.

Another trick is to add Gaussian noise of small amplitude to the gradient to get out of local minima at start.

The last point to be made is the $\alpha(t)$ learning rate. This rate is updated at each step to boost it in the right directions and slow it down in uncertain situations.

III. Indexed t-SNE

The initial formulation of t-SNE leads to a different embedding for each new initialization. Instead of learning a new embedding from scratch, it-SNE takes advantage of prior embedding to optimize the new embedding.

Given a support dataset $X^{(0)} \in \mathbb{R}^{n_0 \times d}$ of n_0 items and its corresponding embedding $Y^{(0)} \in \mathbb{R}^{n_0 \times d_e}$, the goal is to generate an embedding $Y^{(1)} \in \mathbb{R}^{n_1 \times d_e}$ corresponding to $X^{(1)} \in \mathbb{R}^{n_1 \times d}$ of n_1 items.

$$Y^{(1)} \leftarrow it\text{-SNE}(X^{(1)}, X^{(0)}, Y^{(0)}; perp)$$

Two items i and j neighbors in the input space must be neighbors in the embedding space, regardless of their origin dataset.

A. Cost

To achieve this goal, it-SNE minimizes two independent costs:

- the *intra* cost, $C^{(1)}$, defined as in t-SNE using $(X^{(1)}, Y^{(1)})$;
- the *inter* cost, $C^{(0,1)}$, corresponding to joint interactions between $(X^{(0)}, Y^{(0)})$ and $(X^{(1)}, Y^{(1)})$.

The total cost to minimize is:

$$C^{(1)}_{tot} = C^{(1)} + C^{(0,1)} \tag{6}$$

B. Interaction Probability

Similar to t-SNE, the probability matrices are defined to represent items relationships. $P^{(0,1)}$ denotes the probability matrix between input data $X^{(0)}$ and $X^{(1)}$, and $Q^{(0,1)}$ for their respective embeddings $Y^{(0)}$ and $Y^{(1)}$.

1) Input Interaction: For two items $\mathbf{x}_i \in X^{(0)}$ and $\mathbf{x}_j \in X^{(1)}$, the input probability is defined as:

$$p^{(0,1)}_{i|j} = \frac{1}{Z'_j}\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_j^2}\right) \tag{7}$$

where σ_j corresponds to the optimal parameter for j obtained with t-SNE on $X^{(1)}$, and $Z'_j = \sum_i \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_j^2}\right)$ is the normalization constant to obtain $\sum_i p^{(0,1)}_{i|j} = 1$.

The joint probability between i and j is defined as:

$$p^{(0,1)}_{i,j} = p^{(1,0)}_{j,i} = \frac{1}{2}\left(\frac{p^{(0,1)}_{i|j}}{n_1} + \frac{p^{(1,0)}_{j|i}}{n_0}\right) \tag{8}$$

The symmetrization allows taking into account the density of the two datasets on a given location. Additionally, the normalization allows to equalize the dataset influence. If one dataset is larger than the other, it would contribute more are the total forces from each of its items would be larger than for the smaller dataset. The normalization by dataset size allow to obtain equivalent contribution.

2) Output Interaction Probabilities: The goal of it-SNE is not to place new items $Y^{(1)}$ on existing holes of $Y^{(0)}$, but to have $Y^{(1)}$ on a parallel layer of $Y^{(0)}$. To relax forces, and to take into account embedding separation, a penalty factor ϵ is introduced to artificialy separate points belonging to different embeddings. It could be seen as new embedding dimension $d_e + 1$, such as $\mathbf{y}^{(0)}_i = [y^{(0)}_{i,1}, y^{(0)}_{i,2}, ..., y^{(0)}_{i,d_e}, 0]$ and $\mathbf{y}^{(1)}_j = [y^{(1)}_{j,1}, y^{(1)}_{j,2}, ..., y^{(1)}_{j,d_e}, \epsilon]$. The distance between two items is then $\|\mathbf{y}^{(0)}_i - \mathbf{y}^{(1)}_j\| = \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2$.

The kernel used for the definition of output probabilities is kept unchanged, up to the addition of ϵ:

$$q^{(0,1)}_{i,j} = \frac{1}{V'}\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2 + \epsilon^2\right)^{-1} \tag{9}$$

where $V' = \sum_{i,j}\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2 + \epsilon^2\right)^{-1}$ is the normalization constant, which ensures $\sum_{i,j} q^{(0,1)}_{i,j} = 1$.

3) Cost Minimization: The modification of t-SNE algorithm has a limited impact on the cost derivative formulation. The only change impacts the strength of the gradient by the addition of the term ϵ^2:

$$\frac{\partial C^{(0,1)}}{\partial \mathbf{y}^{(1)}_i} = 4\sum_j (p^{(0,1)}_{i,j} - q^{(0,1)}_{i,j})\frac{\mathbf{y}_j - \mathbf{y}_i}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2 + \epsilon^2} \tag{10}$$

The initial solution $Y^{(1)}(0)$ is optimized following equation (5), replacing $\frac{\partial C}{\partial Y}$ by $\frac{\partial C^{(1)}}{\partial Y^{(1)}} + \frac{\partial C^{(0,1)}}{\partial Y^{(1)}}$.

IV. Experimental Setup

A. Algorithm Parametrization

For all experiments, the target perplexity is set to 30. The initial vector of Y, used to start the optimization process is drawn from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma = 10^{-4}$. At each iteration step, a gaussian noise of standard deviation $\sigma = 10^{-4}$ is added to the gradient.

The initial learning rate $\alpha(0) = 10$ is adapted at each timestep for each item i and dimension k according to the similarity between the gradient and the previous displacement direction $\delta_{i,k}(t) = -\frac{\partial C}{\partial y_{i,k}}(t).(y_{i,k}(t-1) - y_{i,k}(t-2))$:

$$\alpha_{i,k}(t) = \begin{cases} \alpha_{i,k}(t-1) + 0.2 & \text{if} \qquad \delta_{i,k}(t) > 0 \\ \alpha_{i,k}(t-1) \times 0.8 & \text{else.} \end{cases}$$

The momentum rate $\eta(t)$ is adapted over learning, such as $\eta(t) = 0.5$ if $t < 250$ else 0.8.

For t-SNE, the early exaggeration trick is used for the 100 first steps, with an exaggeration factor of 2. For it-SNE, the early exaggeration was disabled.

The number of training steps for a t-SNE embedding and it-SNE is fixed to 300 and 200 respectively. Unless specified, $\epsilon = 1$.

t-SNE and it-SNE were implemented in Python, using NumPy library [16].

B. Datasets

We propose to illustrate the results it-SNE over two types of datasets:

- A synthetic dataset, where all parameters can be controlled at ease;
- A real-world dataset to look at its capabilities on unknown distributions.

1) High Dimensional Gaussians: High dimensional Gaussians are good study candidates, as all dimension are equivalent, preventing the use of renormalization methods. The dataset was constructed inspired from the protocol described in [15].

A dataset of 100 dimensions is created by generating Gaussian clusters. 10 Gaussian center positions $\{\boldsymbol{\mu}_g\}_{g=1:10}$ are generated uniformly at random in $\boldsymbol{\mu}_g \in [-0.5, 0.5]^{100}$. For each Gaussian g, 100 items are sampled from the multivariate normal distribution of mean $\boldsymbol{\mu}_g$ and standard deviation σ. This process leads to a dataset of 1000 items.

In the paper Dynamic t-SNE [15], the authors proposed to build a temporal dataset made of shrinking Gaussians. For each item $\mathbf{x}$ associated with the Gaussian g, the distance between the item and the center is reduced by 10 % at each timestep, such as $\|\mathbf{x}(t) - \boldsymbol{\mu}_g\| = 0.9^t \|\mathbf{x}(0) - \boldsymbol{\mu}_g\|$. Mechanically, $\sigma(t)$ is reduced in the same proportion, such as $\sigma(t) = 0.9^t \sigma(0)$. For this experiment, $\sigma(0) = 1.0$. The shrinking process is followed for 9 steps, which leads to $\sigma(10) \approx 0.4$.

By changing the time direction, a similar experiment with growing Gaussians is generated, starting with $\sigma(0) = 0.4$, and growing at the rate of $\sigma(t) = 0.9^{-t} \sigma(0)$.

While these two introductive experiments keep the total number of point stable and homogeneous density for each Gaussian, we propose to build a temporal dataset where heterogeneity appears over time. For each Gaussian g, an expansion rate is sampled uniformly at random from $r_g \in [0.5, 1.5]$, where 1. leads to an invariance of the element number. In contrast, 0.5 leads to a 50 % step reduction in the number of elements. Each Gaussian starts with $n_g(0) = 100$ points. The number of points for Gaussian g at step t is $n_g(t) = \lfloor n(0) r_g^t \rceil$, but the standard deviation is kept stable with $\sigma = 0.4$. A temporal dataset is generated for 4 successive update steps.

2) Citation Graphs: A citation graph $G = (V, E)$ is a directed acyclic graph (DAG). V represents the a set of documents, like scientific papers, patents, law articles, blog posts, which are supposedly immutable. E is the set of directed edges, with $e = (a, b) \in E$ meaning that the document a is referring to document b, implicitly but assuming that a is newer than b.

Graphs are data structure that are difficult to represent in 2D, because they of their sparsity and the distribution in power-law of their node degree, which leads to a small number of strongly connected nodes, and a large number of weakly connected nodes. Nonetheless, citation graphs, as well as other real-world graphs, organise into local communities which are interesting to study.

We propose to study the evolution of research communities over time by embedding the documents published each year. The embedding obtained for year t is reused for year $t+1$, which would be used in turn to build the following embedding. The DBLP dataset version 12 [17] was used for this purpose. This dataset corresponds to the citation graph of scientific papers around the topic of computer science. It contains 4.894.081 papers and 45.564.149 citing relationships. Metadata are available for the majority of the documents, providing information such as title, publication date, abstract, authors information, conference or journal reference, reference links, and keywords. Keywords also called *field of study*, are automatically extracted according to the method described in [18].

a) Graph preprocessing: A preprocessing removes all existing cycles, as some papers are updated after the official publication date, adding a few citations. This phenomenon concerns a minority of papers, but creates undesirable loops. The cycles are removed using a DFS approach, removing any edges accessed twice by a DFS branch. The main connected component is then kept, removing all papers with no bibliography or belonging to an isolated community.

b) Node Sampling: The citation graph considered is too large to be processed at once. We took advantage of the keywords to select a subset of papers related to a particular topic. We selected all documents related to *cryptography* and related topics, which corresponded to around 100.000 documents published between 1953 and 2020. The documents with less than one reference and less than one citation were removed, which left 70.000 documents published between 1953 and 2020, with the majority published between 2005 and later.

c) Extracting Distance Matrix from a Citation Graph: A graph cannot be converted to tabular data used by t-SNE.

Nonetheless, t-SNE and it-SNE use the distance between items and do not focus on particular features. Even if it is not possible to measure the euclidean distance between nodes on a graph, a distance matrix can be obtained.

The distance between nodes is not a great distance measure. The possible integer values are coarse measures, and the distance is not correctly defined for all pairs in a DAG. Plus, a document may quote unrelated documents from another discipline for illustrating its argumentation with other scientific views. The node could be at distance 2 of many papers on a completely unrelated field, just because of a single example.

A way to measure the document similarity is through bibliographic coupling [19]. Two documents sharing some of their references are coupled, even if there is no direct path from one to the other in the DAG. The strength of the coupling depends on the overlap size. As the graph is sparse, we extend the bibliographic coupling to indirect reference until distance 3. This strategy reduces the sparsity and improves the sensibility of the coupling.

The coupling strength is measured using the Jaccard similarity measure. For two documents, a and b, with respective reference sets A and B, the similarity between these two documents is defined as:

$$Sim(a, b) = \frac{|A \cap B|}{|A \cup B|} \tag{11}$$

A similarity is a value $s \in [0, 1]$, while a distance is a value $d \in \mathbb{R}^+$. A distance close to 0 is equivalent to a similarity of 1, while a large distance to a similarity close to 0. Because the similarity behavior is the opposite of the one of a distance, a form of distance can be obtained by transforming the similarity:

$$D(a, b) = \frac{1}{Sim(a, b) + \xi} - \frac{1}{1 + \xi} \tag{12}$$

where $\xi = 10^{-5}$ is a small constant, which avoids division by zero and limit the maximal distance to 10^5. Using (11) and (12), the citation graph can be appropriately transformed for the embedding algorithms.

C. Metrics

1) Cost: The configuration of the embedding is evaluated in terms of cost, which is the quantity minimized by t-SNE:

$$\sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}$$

where P and Q correspond to the *intra*-probability matrices without considering the interaction with the previous embedding.

2) Distortion: For an experiment where the items in $X^{(0)} \approx X^{(1)}$, a way to measure how well it-SNE places the point is to measure the distance between the initial and new position $Y^{(0)}$ and $Y^{(1)}$ and new position $Y^{(1)}$.

$$Err(Y^{(0)}, Y^{(1)}) = \sum_i |\mathbf{y}_i^{(0)} - \mathbf{y}_i^{(1)}|$$

V. Experimental Results

A. Evolution of Gaussians

a) Shrinking Gaussians: This experiment reproduces the protocol proposed in [15], were points are progressively attracted toward their Gaussian center of reference. The initial embedding $Y^{(0)}$ corresponding to $X^{(0)}$ is obtained with t-SNE, while all following embedding $Y^{(t)}$ for $t \geq 1$ are obtained with it-SNE using as a support the pair $(X^{(t-1)}, Y^{(t-1)})$.

Fig. 1 shows the result with it-SNE, which transforms undistinguishable groups into well-defined groups. Our approach works as well as dt-SNE, but while dt-SNE uses the $Y^{(t-1)}$ position to initialize $Y^{(t)}$, it-SNE restarts from random vectors. This difference frees our model from restrictions on the size and content of the dataset. This experiment was performed on freshly generated points sampled at each time step, leading to the same results as those presented in Fig. 1.

b) Growing Gaussians: By reversing time direction, we get another set of experiments. The dataset starts with 10 Gaussians with $\sigma(0) = 0.4$, progressively increased to $\sigma(9) \approx 1.0$

The results are represented on Fig. 2. This task is easier than the previous, as the first embedding starts with well-separated clusters. The next embedding support is of higher quality than in the previous experiment where the variance was larger. Even after moving to a noisier dataset (the last plot of *growing Gaussians* has almost the same variance as the first plot of *shrinking Gaussians*), the separation between items of different clusters is preserved even if clusters are not spaced from each other. A support embedding of good quality helps to guide items belonging to a noisy dataset, building a better embedding.

c) Change in Density: The last visual result to present with Gaussians focuses on density changes with a fixed σ. The number of samples per Gaussian changes at each step. For Gaussian g at step t, the number of items generated around $\boldsymbol{\mu}_g$ is calculated as $n_g(t) = \lfloor n_0 r_g^t \rceil$.

Fig. 3 illustrates the result of this process. At the start, all clusters have the same size and density and are spaced equally from each other. As time passes, some of the Gaussians grow in size, while others shrink. The area used by shrinking Gaussians decreases while growing Gaussians expand over the space available. The cluster positions are preserved despite the change of density unless too few items are present to allow the cluster aggregation.

B. Influence of ϵ

In this subsection, we discuss the impact of ϵ on the applied forces. For simplicity of the notation, the elements $p_{i,j}$ and $q_{i,j}$ correspond to $p_{i,j}^{(0,1)}$ and $q_{i,j}^{(0,1)}$, with i an element of the support dataset (0) while j an element of the dataset to embed (1). The same simplification is applied to $y_i^{(0)}$ and $y_j^{(1)}$.

1) Forces: The factor ϵ has an impact on Q and the gradient. The strength of the gradient is reduced, as ϵ plays in $\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2\right)^{-1}$. As ϵ grows, the forces coming from the support embedding vanish.

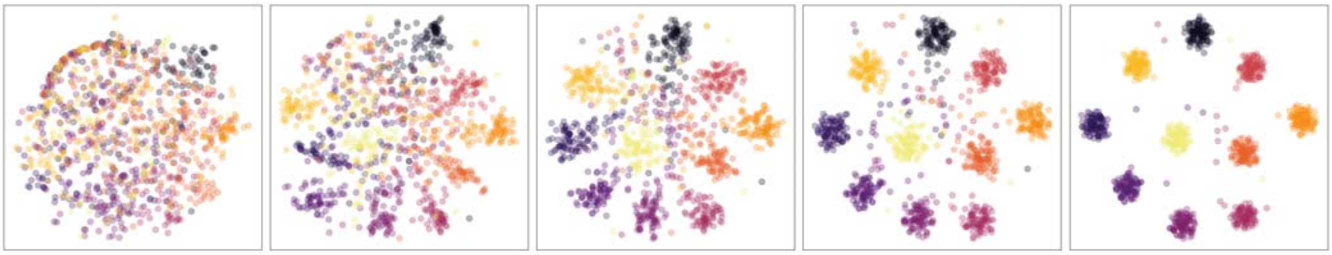

Fig. 1 Shrinking Gaussians. From left to right, steps 1, 3, 5, 7 and 9 are represented. Points are colored according to their Gaussian center of reference. $\epsilon = 1.0$

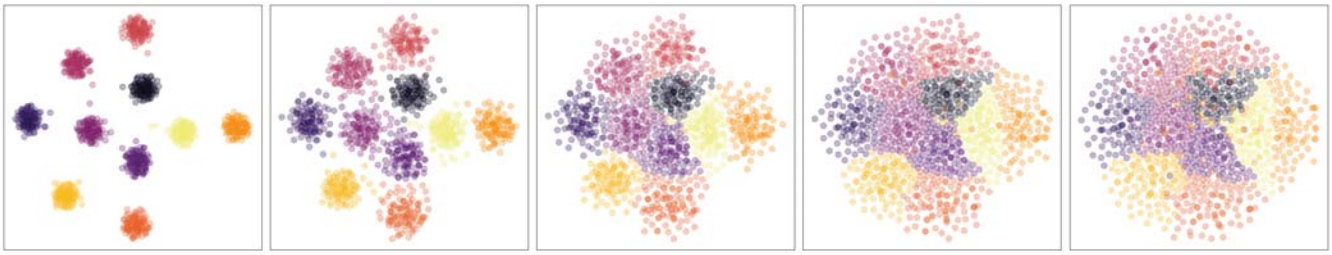

Fig. 2 Growing Gaussians. From left to right, steps 1, 3, 5, 7 and 9 are represented. Points are colored according to their Gaussian center of reference. The penalty factor is set to $\epsilon = 1.0$

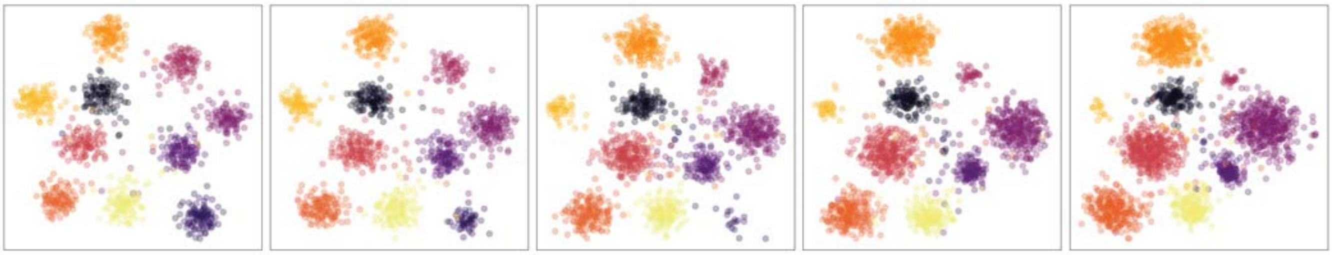

Fig. 3 Evolving Gaussians. 10 Gaussians of various size with $\sigma = 0.4$. From left to right, time steps 0 to 4 are displayed. The penalty parameter is set to $\epsilon = 1.0$. Points are colored according to their Gaussian center of reference

The ϵ factor has an impact on the influence of items by changing the numerator and denominator of Q. When ϵ increases, the numerator $\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2\right)^{-1}$ decreases for all item pairs. This value decays faster for items i and j that are close to each other than those which are not. The denominator of Q in (9), $V' = \sum_{i,j} \left(1 + \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2\right)^{-1}$ decreases when ϵ increases. As both numerator and denominator of Q decrease, the effective variation depends on the item proximity. Q increases for a pair of distant items and decreases for items that are close. The growth of ϵ reduces the variance, converging to $\lim_{\epsilon \to \infty} \mathbb{V}(Q) = 0$, and homogenizes the value of $q_{i,j}$ to $\lim_{\epsilon \to \infty} q_{i,j} = \frac{1}{n_0 n_1}$.

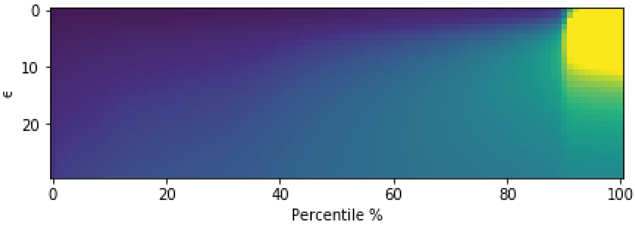

Fig. 4 Percentile values of the $q_{i,j}$ for different values of ϵ. Colors are linear with the value of Q: dark colors correspond to value close to 0 while bright color to high value. The color saturates in yellow at $3n^{-2}$

Fig. 4 illustrates the evolution of Q values with ϵ, using the dataset with 10 Gaussians. For the low value of ϵ, only items belonging to the same Gaussian interact together (i.e., 10% of the points). As ϵ grows, the weights of the nearest neighbors decrease to the profit of more distant neighbors.

Because of the kernel asymmetry between P and Q, the reduction of variance and the convergence to the mean of Q leads to two different behaviors, depending on the closeness of two items. The items are divided into two classes based on the value of $p_{j,i}$. The closer neighbors with $p_{j,i} > \frac{1}{n_0 n_1}$ are considered as the nearest neighbors while the other distant neighbors. For nearest neighbors, the artificial distancing created by ϵ lowers the output probability $q > q^{\epsilon}$, for $q^{\epsilon} = q(\epsilon > 0)$ and $q = q(\epsilon = 0)$. P and Q's difference is $p - q^{\epsilon} > p - q$ is then larger, which leads to larger attractive forces from the close neighborhood. The opposite effect happens for distant neighbors where $q < q^{\epsilon}$, which leads to $p - q > p - q^{\epsilon}$, generating repulsive forces.

To summarize, on the one hand, the forces are globally lowered as ϵ impacts the gradient, and on the other hand, the discrimination between nearest and distant neighbors grows as ϵ amplifies the asymmetry.

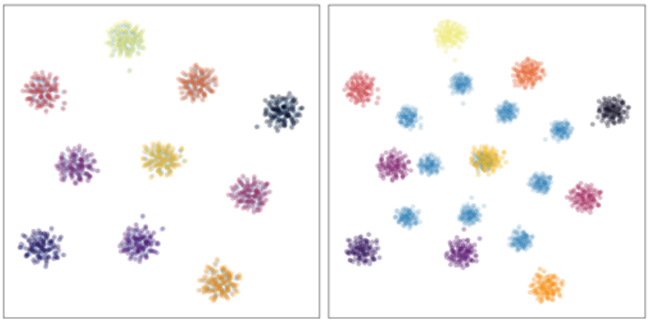

Fig. 5 Mapping of Gaussians with standard deviation $\sigma = 0.4$. The embedding $Y^{(0)}$ is colored in light blue on each plot, while items of $Y^{(1)}$ are colored according to their cluster of reference. Left: $\epsilon = 1$, right: $\epsilon = 25$

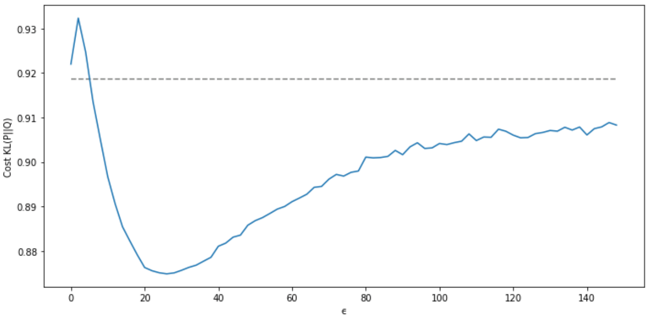

Fig. 7 Evolution of Kullback-Leibler cost with increasing ϵ. Gaussians with standard deviation $\sigma = 0.4$. The dashed line corresponds to the cost of embedding $Y^{(0)}$ obtained with the regular t-SNE

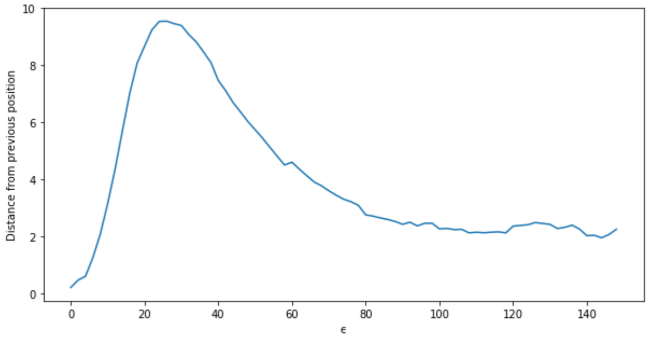

Fig. 6 Evolution of the embedding error $Err(Y^{(0)}, Y^{(1)})$ with ϵ using 10 Gaussians with standard deviation $\sigma = 0.4$

2) Displacement from Origin: A way to study the two contributions can be done by looking at the ability of it-SNE to recover the exact embedding positions. For a dataset $X^{(0)}$, first is computed $Y^{(0)}$ with:

$$Y^{(0)} \leftarrow t\text{-SNE}(X^{(0)})$$

followed by:

$$Y^{(1)} \leftarrow it\text{-SNE}(X^{(0)}, X^{(0)}, Y^{(0)})$$

Fig. 5 shows the cluster conformation for two different ϵ. The clusters are correctly matched, both in classes and in positions for $\epsilon = 1$, as $Y^{(1)}$ masks $Y^{(0)}$. However, for $\epsilon = 25$, while the cluster classes are correctly matched, they are distant from the original position.

To study the transition between the two conformations, we look at the distortion $Err(Y^{(0)}, Y^{(1)})$, which measures the distance between the initial and final position.

Fig. 6 shows the impact of an increase of ϵ over the item locations. For low ϵ, the error is almost 0. The average distance error for $\epsilon = 0$ is 0.204, while the average distance to the first nearest neighbor is 0.218, for an embedding of diameter 32.1. With increasing values of ϵ, the distances between initial and final positions grow. The right part of Fig. 5 illustrates the situation. The repulsive forces between distant neighbors are stronger than the attraction of the nearest neighbors. This pushes the clusters even further away from each other, increasing the distance from the initial position. The maximal distortion is reached around $\epsilon = 25$, a value of the order of the embedding diameter.

After this maximum, the distortion error decreases to stabilize at a value of 2.6. Compared to the diameter of one cluster presented in 5, the value is relatively similar, around 2.88. The clusters overlap, but the forces are not strong enough to accurately bring them to their exact location.

3) Cost: it-SNE tries to minimize two costs at the same time. They might not be compatible, with opposite gradient directions. The intra cost allows assessing the embedding quality to see if the embedding could reach a correct minimum, or if the inter forces constrained the embedding to a non-optimal state.

Fig. 7 presents the results for the same conformations as in Fig. 6. The cost is slightly higher to start with, but it decreases as ϵ increases to arrive at a local cost minimum. This local minimum corresponds to the distortion maxima of Fig. 6. The conformation $\epsilon = 25$ is a more stable configuration than the initial one. For larger values, the forces' strength decrease, but stay below the baseline cost, corresponding to the support configuration. It is to note that the cost difference between the lowest and highest cost value in Fig. 7 is relatively small. All conformations are relatively good, some a little bit more than others.

4) Convergence Speed: In our protocol, the number of training step was fixed to control the computational time. For a support embedding of n_0 items and a new dataset to embed of n_1 items, the t-SNE cost is proportional to n_1^2, while the cost for it-SNE is proportional to $n_1^2 + n_0 n_1$. In our experiments $n_0 = n_1 = 1000$ which means that the number of operations performed by it-SNE is twice the number of t-SNE.

Fig. 8 shows the cost evolution for several values of ϵ. All curves start with a plateau, which corresponds to when the learning rate is not boosted enough to lead to significant changes per steps. The lower ϵ is, the shorter the time spent on the plateau is. A decay part follows the plateau, which smoothly slowed down until convergence.

The dotted orange line allows comparing t-SNE with it-SNE on the number of computational operations. t-SNE is faster than it-SNE, but the difference between the two is not very large.

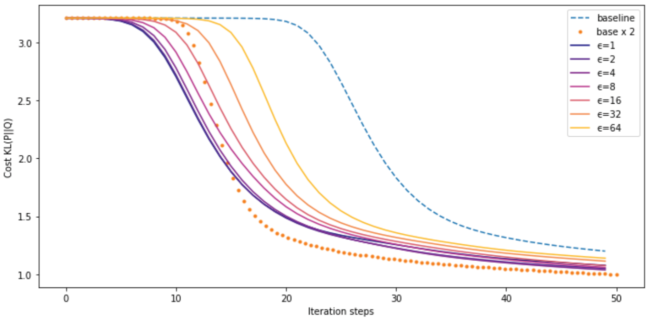

Fig. 8 Evolution of the Kullback-Leibler cost over training time, with 10 Gaussians of standard deviation $\sigma = 0.4$. The yellow to dark lines correspond to it-SNE for several value of ϵ The blue dashed line corresponds to the cost evolution for t-SNE. The dotted line corresponds to baseline with t-SNE, speeded by a factor 2

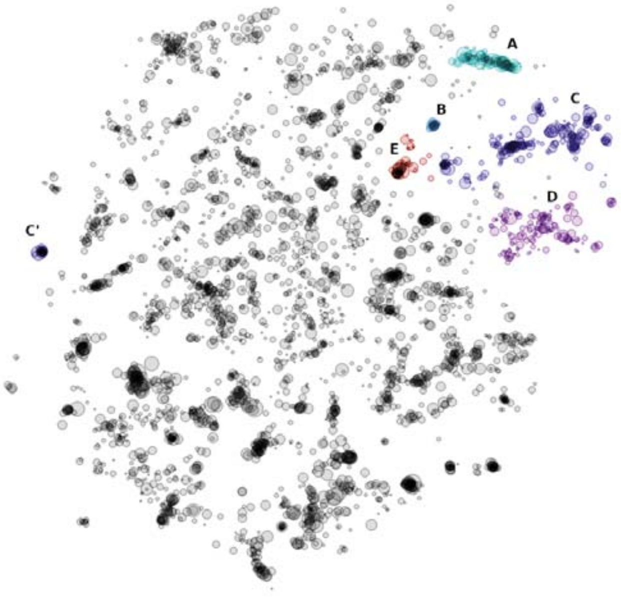

Fig. 9 Citation graph of cryptographic papers in 2010, with some highlighted clusters. A: Hashing, B: Network Code, C and C': Biometry, D: Watermarking/Data Hiding, E: Passwords. The item size is proportional to the number of citations

C. Citation Graph Embedding

Gaussian clusters are easy to study as the parameters are fully controlled. Citation graphs are selected to illustrate a real-world example of it-SNE capabilities.

Because a scientific article refers to relevant papers in its field, this type of dataset presents local community structures. The evolution of the number of researchers is leading to an expansion of knowledge in many fields, and the creation of new ones. Other fields tend to disappear, because of a lack of support from the scientific community or an absence of new discoveries. The study of these phenomena allows to retrace history and reconstruct the science phylogeny [20].

Fig. 9 represents the papers published in the cryptography/security field in 2010. Papers group together to form connected clusters with various shapes, sizes and densities. To the default of a clustering algorithm, some groups have been highlighted and labelled by hand with the help of documents' title and keywords.

Cluster A, with *Hashing*'s general topic, is compact with items well connected to each others. Cluster D about *Watermark* is more diffuse than A, but items are still grouped together. The *Biometry* cluster C is composed of several sub-units of various density. Cluster B and E about *Network Code* and *Password* respectively are much more compact than the other presented.

The papers in each cluster are mostly in phase with the general cluster topic, showing local unity. Nonetheless, the reverse is not valid: a cluster about a particular topic does not enclose all documents related to it. A keyword may correspond to two different ideas, or two distinct communities may work on different aspects of the problem. For instance, this is the *Biometry* field case, which occurs twice in various embedding locations. There is one large cluster on the right and a smaller one on the left (denoted C and C' respectively). While the large cluster C is about general biometric recognition methods, the C' is focused on authentication scheme, mixing *Biometry*, *Password* and *Authentication* topics together.

Scientific communities are dynamic and adapt to new trends. Fields emerge, grow, interact, split, and disappear. it-SNE allows reusing these initial cluster positions for the next subsequent embedding, which allows tracking such dynamics.

Fig. 10 presents the evolution of the embedding 9 from 2010 to 2017, with the same color highlighting. The general shape of the embedding is stable over time, with clusters' positions preserved. A small drift of the clusters occurs, which is noticeable after a few embedding steps. The growth and shrink of some clusters is visible, such as for the second biometry cluster on the left which expands over the years. Merges are also visible, such as for the *Biometry* and *Password* clusters, which first merge in 2013.

To have a better view of the evolution, Fig. 11 shows an enlarged view of the top right area of the embedding, where *Biometry* and *Password* field of study clusters are. In this area, different types of phenomena occur. There are stable clusters present, such as the *Hashing*, *Biometry*, and *Watermark*, with constant size and density. *Password* cluster growths in size, while *Network Coding* disappeared in 2015. The *Biometry* and *Password* groups have been interacting with each other and began merging in 2017.

As the clusters have been human extracted, no metric measure has been tested. Tables listing the most cited paper for each year for the different clusters are presented in the appendix (Tables I, II, III, IV, V, VI). The articles presented in these tables are very consistent with the theme of the cluster from which they originate.

VI. Discussion

A. Complexity

The normal complexity of t-SNE is in $\mathcal{O}(n^2)$, where n is the number of items. While some optimization exists when the dataset is tabular, like the Barnes-Hut optimization [21] which reduces the cost to $\mathcal{O}(n \log n)$, for data like graphs from which a distance matrix can be obtained, this quadratic cost is prohibitive.

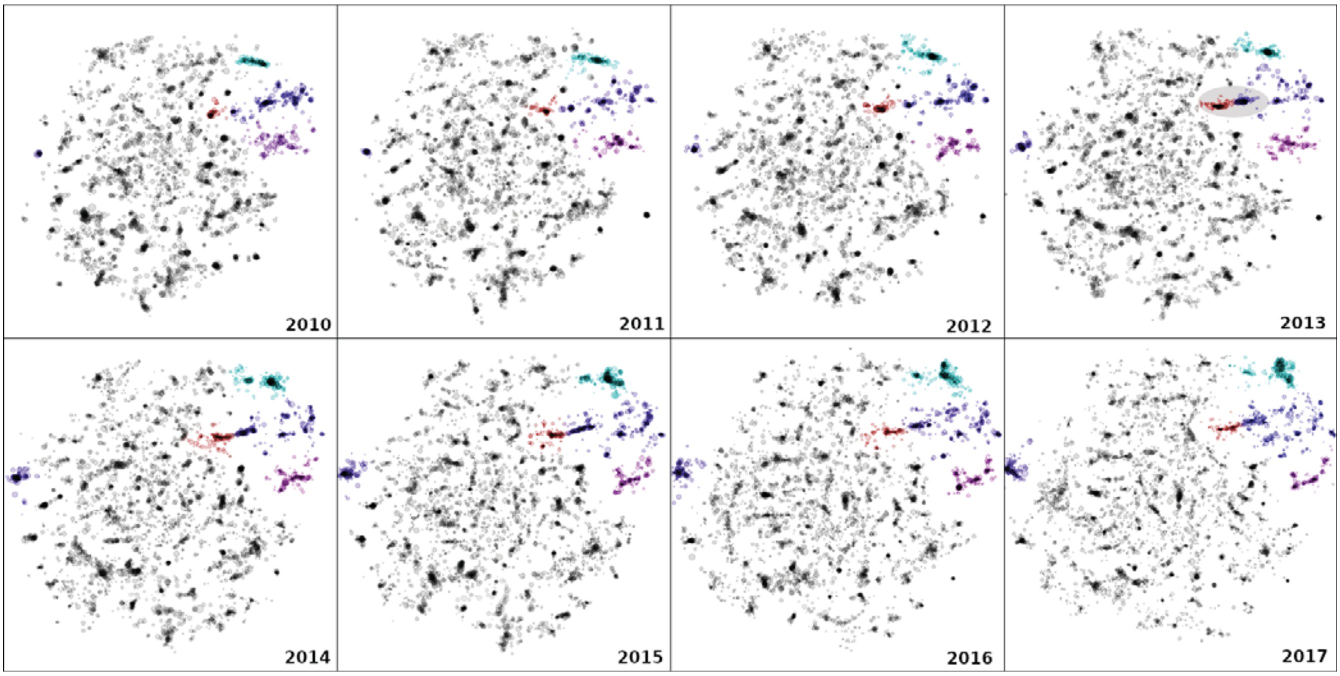

Fig. 10 Citation graph of cryptographic papers, published between 2010 to 2017. Clusters are highlighted according to the previous figure coloring scheme. Size of points is proportional to the logarithmic number of citations. Dark areas correspond to highly grouped papers. The fusion between Biometry and Password clusters in 2013 is shaded in grey

If the dataset is cut into k equivalent pieces of $m = \frac{n}{k}$ pieces, the algorithm would run in $\mathcal{O}(2km^2) = \mathcal{O}(\frac{2}{k}n^2)$, where the $2\times$ stands for the two gradient parts. This reduces by a factor $\frac{k}{2}$ the complexity. The complexity here measures the average number of operations for one run of an iteration step. However, as the algorithm uses gradient descent, a convergence criterion governs the total number of steps. Intuitively, the number of steps required to converge for a large dataset seems larger than for a smaller dataset. The decomposition of the dataset into pieces would reduce the effective computational time.

Concerning the memory requirements, the normal t-SNE requires a storage space of $2n^2$, necessary for the matrix P and Q. Using it-SNE with a dataset split into k pieces, 4 matrix of size m^2, $(P^{(t)}, Q^{(t)}, P^{(t-1,t)}, Q^{(t-1,t)})$ are needed to compute the embedding. The total amount of memory required is then $4m^2 = 4\left(\frac{n}{k}\right)^2$, which is more interesting, as $\frac{k^2}{2}$ reduces its cost. The computational time can be extended on a machine, but not its memory. Our proposed method can be helpful as way to map large datasets by cutting them into smaller pieces.

B. Selecting ϵ

1) Speedup: The parameter ϵ controls the applied forces and the gradient strength. A low value of ϵ creates strong forces which leads to fast convergence. Nonetheless, forces prevent items to move to other locations. An increase of ϵ would relax the system and help an item to arrive on a low energy state. As for the *early exaggeration* trick, it would be benefical to start with a small value of ϵ and then finish with a larger value.

2) ϵ and Perplexity: The perplexity governs the number of neighbors taken into account. The numerator of P in equations (1) and (7) grows with a perplexity increase for all items. The denominator grows too, and leads to a decrease of P for the nearest neighbors. It affects on Q which needs to decrease. The distances between Y's increase with larger perplexity.

If the support embedding has a different perplexity than the target perplexity, a large ϵ may help to adapt to the new perplexity, by relaxing forces strength. The clusters would be attracted to nearby position, and the intra forces would arrange the local shape.

C. Adaptation to Large Changes of Density

Our experiments have been done with temporal datasets with constant or slowly evolving size. The use of a support dataset of highly different size may constrain the system optimization. For a fixed perplexity, the diameter of an embedding grows with the size of the dataset in $(n)^{\frac{1}{d_e}}$. Two datasets of different size would have a radius of $r^{(0)}$ and $r^{(1)}$. The items located in the middle of the embedding can be correctly matched with the support items. However, for peripheral items, there would be a gap of $|r_0 - r_1|$. it-SNE would create a distortion, constraining items of (1) to expand if $r_0 > r_1$, which would increase the interdistances between items in $Y^{(1)}$. For $r_0 < r_1$, a shrinking would occur leading to the same distortion. As intra forces of (0) do not play any role, a trick to free from this constraint would be to scale $Y^{(0)}$ into $Y^{(0*)}$, using the scaling ratio $s = \left(\frac{n_1}{n_0}\right)^{\frac{1}{d_e}}$, such as $Y^{(0*)} = sY^{(0)}$. This would help for datasets of similar densities. For different densities, local distortions would still occur.

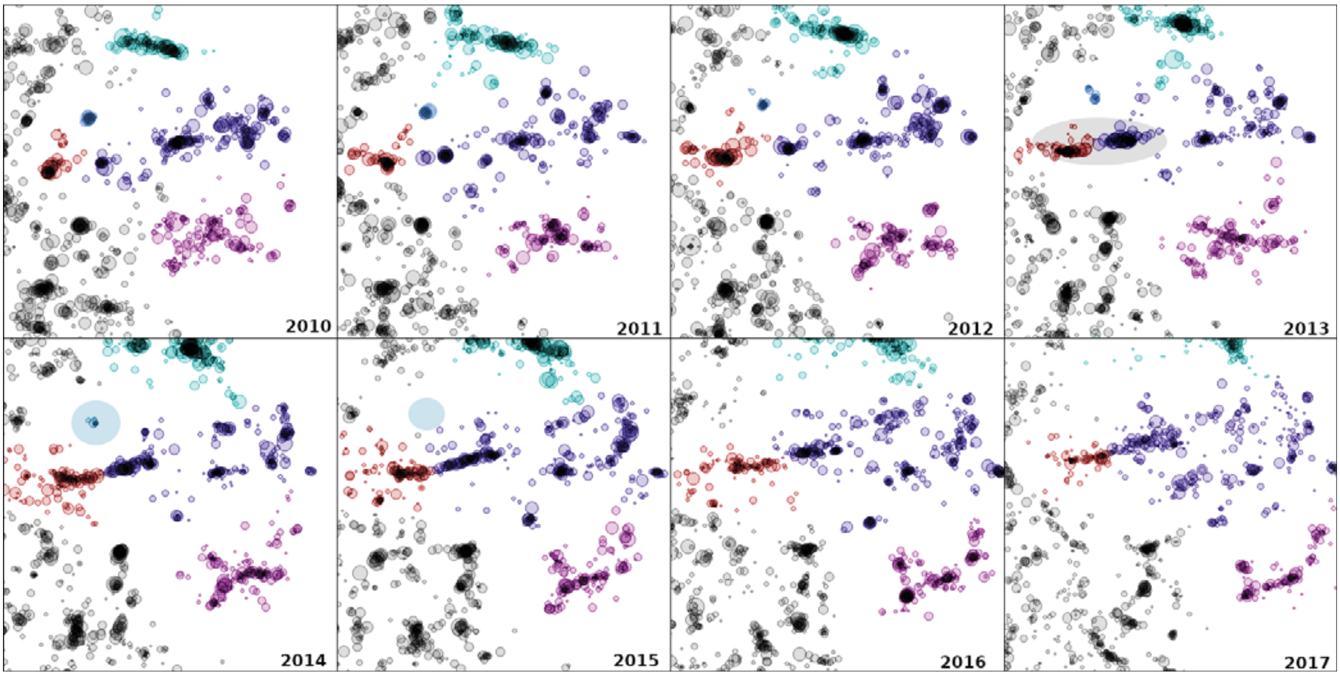

Fig. 11 Citation graph of cryptographic papers, published between 2010 to 2017. Clusters are highlighted according to the previous figure coloring scheme. The size of points is proportional to the logarithmic number of citations. Dark areas correspond to highly grouped papers. The fusion between Biometry and Password clusters in 2013 is shaded in grey

D. Using More than One Support Embedding

The method proposed to use one embedding to support the generation of a new one. A natural question arises about the possibility of using the support of two or more embeddings. This case may happen if two embeddings for $t_0 < t_2$ have been obtained but not for t_1 yet, with $t_0 < t_1 < t_2$. Intuitively, the constrains engendered by the two embeddings might be equivalent to a single one. If more embeddings were to be taken into account, all grouped embedding cost must not prevent the intra forces from playing their role.

For a dataset $X^{(k)}$ taking support of datasets $\mathcal{X} = \{X^{(i)}\}_{i=1:k-1}$ with respective embeddings $\mathcal{Y} = \{Y^{(i)}\}_{i=1:k-1}$, the cost could be rewritten as:

$$C_{tot}^{(k)} = C^{(k)} + \frac{1}{k-1}\sum_{i=1}^{k-1} C^{(i,k)}$$

This adaptation allows generating embedding in between two existing embeddings. Another use of this adaptation is the multivariate case, like for geolocation coordinates, where the new embedding might take the support using multiple non-equivalent datasets. For $\mathcal{X} = \{X^{(i)}\}_{i=1:k-1}$ with relative importance $W = \{w_i | w_i > 0\}_{i=1:k-1}$, the weighted cost would have the form:

$$C_{tot}^{(k)} = C^{(k)} + \frac{1}{\sum_{i=1}^{k-1} w_i}\sum_{i=1}^{k-1} w_i C^{(i,k)}$$

Note that the use of multiple support embeddings increases the cost linearly with the total number of items. Nonetheless, if the nearest datasets are too small to serve as a support, the use of more than one embedding may helps to preserve embedding knowledge and enhance long term coherency.

E. Binary Computation

To create several coherent embeddings for a succession of datasets, the process can be speeded-up by distributing the embedding tasks. If there are k datasets to embed, instead of computing the embeddings in a sequential way, using the support of $t-1$ to compute t, the use of another more distant support would help. The closer the support, the better it would be, as the distribution difference between two neighbor datasets is expected to be lower than for distant datasets.

The computation starts with an initial embedding for $\lfloor \frac{k}{2} \rceil$. Then, the left and right intervals are divided in their middle. An embedding is issued for $\lfloor \frac{k}{4} \rceil$ and $\lfloor \frac{3k}{4} \rceil$. Then, the embedding for subset $\lfloor \frac{1}{8} k \rceil$ can be computed using the support of $\lfloor \frac{k}{4} \rceil$, and $\lfloor \frac{7k}{8} \rceil$ using the support of $\lfloor \frac{3k}{4} \rceil$. For the middle parts $\lfloor \frac{3}{8} k \rceil$ and $\lfloor \frac{5}{8} k \rceil$, their respective embedding is computed using two support embeddings, respectively using $(\lfloor \frac{k}{4} \rceil, \lfloor \frac{k}{2} \rceil)$ and $(\lfloor \frac{k}{2} \rceil, \lfloor \frac{3k}{4} \rceil)$. The procedure is repeated recursively until all embeddings have been obtained. This decomposition allows to speedup the process from $\mathcal{O}(k)$ to $\mathcal{O}(\log_2(k))$.

VII. Conclusion

This paper presents a method adapting t-SNE algorithm to reuse a previous embedding to generate a new one. Compared to the base method t-SNE, an additional cost term is added. This cost links the new items to embed to the support embedding, creating attractive forces. These forces enable the similar items from the support and current datasets to be located on the same embedding area. Clusters are coherent

Open Science Index, Computer and Systems Engineering Vol:15, No:8, 2021 publications.waset.org/10012177.pdf

in the location from one embedding to the other, enabling the reuse of a classification algorithm on both.

it-SNE was tested on two datasets. The first used synthetic Gaussians forming dense clusters, evolving in density and size over time. The second was the scientific citation graph restricted to cryptography related papers, with small, sparse communities. The algorithm was successful at preserving the cluster locations in both experiments, while preserving t-SNE embedding aspect.

Compared to t-SNE, the computational complexity and the memory requirement of it-SNE are doubled. Nonetheless, the use of a support embedding speedups the convergence process of it-SNE. The total number of operations of it-SNE is, in practice, equivalent to t-SNE.

We proposed two extensions: the first to the multivariate case and the second to distribute the embedding computation. One unsolved problem yet is the adaptation to highly different densities, as t-SNE mechanism tries to keep average distance between neighbors constant, which leads to an expansion of the embedding with increasing dataset size.

it-SNE can be used for many purposes, such as monitoring, anomaly detection, network analysis, allowing to track the evolution of clusters in a low dimensional space. The method is not restricted to temporal datasets and could be used to study the impact one variable's impact on the dataset distribution.

Appendix
Most Cited Papers

Tables I-VI list the most cited document per year per topic. Some titles have been truncated.

TABLE I
HASHING

Year	Title
2010	Semi-supervised Hashing for Scalable Image Retrieval
2011	Minimal Loss Hashing for Compact Binary Codes
2012	Image Signature: Highlighting Sparse Salient Regions
2013	Inter-media Hashing for Large Scale Retrieval
2014	Supervised Hashing for Image Retrieval
2015	Supervised Discrete Hashing
2016	Deep Supervised Hashing for Fast Image Retrieval
2017	Learning Discriminative Binary Codes

TABLE II
NETWORK CODING

Year	Title
2010	Secure network coding over the integers
2011	Secure Network Coding on a Wiretap Network
2012	Cooperative Defence Against Pollution Attacks
2013	An Efficient Homomorphic MAC with Small key Size
2014	A Lightweight Encryption Scheme for Network-Coded Mobile

TABLE III
BIOMETRY

Year	Title
2010	Unobtrusive User-Auth on Mobile Phone
2011	A survey on Biometric Cryptosystems and cancelable biometrics
2012	Touch me once and I Know it's you
2013	Touchalytics: On the Applicability of Touchscreen Input
2014	Image quality Assessment for Fake Biometric Detection
2015	Deep Representation for Iris, Face and Fingerprint
2016	Continuous User Authentication on Mobile Devices
2017	MagNet: A Two-Pronged Defense against Adversarial Examples

TABLE IV
BIOMETRY AND AUTHENTICATION SCHEMES

Year	Title
2010	An Efficient Biometrics-based Remote User Authentication Scheme
2011	Cryptanalysis and Improvement of a Biometrics-based Remote
2012	A secure Authentication Scheme for Telecare Medecine
2013	A Temporal-Credential-Based Mutual Authentication
2014	A Novel User Authentication and Key Agreement Scheme
2015	Robust Biometrics-Based Authentication Scheme
2016	An efficient User Authentication and Key Agreement Scheme
2017	Anonymous Authentication for Wireless Body Area Networks

TABLE V
WATERMARK

Year	Title
2010	Review: Digital Image Steganography
2011	Reversible Data Hiding in Encrypted Image
2012	Separable Reversible Data Hiding in Encrypted Image
2013	Digital Image Forgery Detection using Passive Techniques
2014	Reversibility improved data Hiding in Encrypted images
2015	RAISE: a Raw Images Dataset for Digital Image Forensics
2016	Reversible Data Hiding: Advances in the Past Two Decades
2017	Fragile Image Watermarking with Pixel-wise Recovery

TABLE VI
PASSWORD

Year	Title
2010	Encountering Stronger Password Requirements
2011	Of Passwords and People: Measuring the Effect of Passwords
2012	The Quest to Replace Passwords
2013	Patterns in the Wild: a Field Study of the Usability of Pattern
2014	It's a Hard Lock Life: A Field Study of Smartphone Unlock
2015	”... No one Can Hack My Mind”: Comparing Expert
2016	Who are you? A Statistical Approach to Measuring User Auth.
2017	Zipf's Law in Passwords

REFERENCES

[1] S. Cohen, E. Ruppin, and G. Dror, “Feature selection based on the shapley value.” 01 2005, pp. 665–670.

[2] M. A. Hall, “Correlation-based feature selection for machine learning,” Ph.D. dissertation, 1999.

[3] I. Jolliffe, *Principal Component Analysis*. John Wiley & Sons, Ltd, 2005.

[4] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” 06 2011, pp. 52–59.

[5] M. Maggipinto, C. Masiero, A. Beghi, and G. A. Susto, “A convolutional autoencoder approach for feature extraction in virtual metrology,” *Procedia Manufacturing*, vol. 17, pp. 126–133, 2018, 28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USAGlobal Integration of Intelligent Manufacturing and Smart Industry for Good of Humanity.

[6] T. Kohonen, *Self-organizing maps*, 3rd ed., ser. Springer series in information sciences, 30. Berlin: Springer, Dec. 2001.

[7] J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, no. 5500, p. 2319, 2000.

[8] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” 2018, cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap.

[9] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[10] D. Kobak and P. Berens, “The art of using t-sne for single-cell transcriptomics,” *bioRxiv*, 2019.

[11] N. Pezzotti, A. Mordvintsev, T. Höllt, B. P. F. Lelieveldt, E. Eisemann, and A. Vilanova, “Linear tsne optimization for the web,” *CoRR*, vol. abs/1805.10817, 2018. [Online]. Available: http://arxiv.org/abs/1805.10817

[12] L. van der Maaten, “Learning a parametric embedding by preserving local structure,” in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. van Dyk and M. Welling, Eds., vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 384–391. [Online]. Available: http://proceedings.mlr.press/v5/maaten09a.html

[13] M. R. Min, H. Guo, and D. Shen, “Parametric t-distributed stochastic exemplar-centered embedding,” *CoRR*, vol. abs/1710.05128, 2017. [Online]. Available: http://arxiv.org/abs/1710.05128

[14] A. Boytsov, F. Fouquet, T. Hartmann, and Y. L. Traon, “Visualizing and exploring dynamic high-dimensional datasets with lion-tsne,” *CoRR*, vol. abs/1708.04983, 2017. [Online]. Available: http://arxiv.org/abs/1708.04983

[15] P. E. Rauber, A. X. Falcão, and A. C. Telea, “Visualizing time-dependent data using dynamic t-sne,” in *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers*, ser. EuroVis ’16. Goslar, DEU: Eurographics Association, 2016, p. 7377.

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R’ıo, M. Wiebe, P. Peterson, P. G’erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[17] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, “Arnetminer: Extraction and mining of academic social networks,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 990998.

[18] K. Wang, Z. Shen, C. Huang, C.-H. Wu, D. Eide, Y. Dong, J. Qian, A. Kanakia, A. Chen, and R. Rogahn, “A review of microsoft academic services for science of science studies,” *Frontiers in Big Data*, vol. 2, p. 45, 2019.

[19] B. H. Weinberg, “Bibliographic coupling: A review,” *Information Storage and Retrieval*, vol. 10, no. 5, pp. 189–196, 1974.

[20] D. Chavalarias and J.-P. Cointet, “The reconstruction of science phylogeny,” 04 2009.

[21] L. van der Maaten, “Barnes-hut-sne,” in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2013.