HCt-SNE

Vu, Viet Minh; Bibal, Adrien; Frénay, Benoît

Published in:
International Joint Conference on Neural Networks

Publication date:
2021

Document Version
Peer reviewed version

HCt-SNE: Hierarchical Constraints with t-SNE

Viet Minh Vu
Faculty of Computer Science - NaDI
University of Namur
Namur, Belgium
vuvietminh@unamur.be

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

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

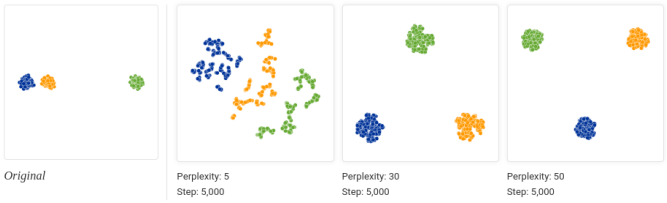
***Abstract*—Dimensionality reduction (DR) methods are useful when analyzing high dimensional data, in particular, if one wants to visualize them. t-distributed stochastic neighbor embedding (t-SNE), one of the most widely used DR methods, can preserve neighborhood information and reveal groups in embeddings. However, it may not preserve the global structure and fail to reveal the semantic information in the visualization. From a user point-of-view, a DR visualization is useful if it not only reveals hidden structures in the data but also corresponds to the user knowledge. This paper addresses these problems by proposing *Hierarchical Constraint* t-*SNE* (HCt-SNE), a method that allows users to integrate hierarchical constraints directly into t-SNE embeddings. The user constraints are encoded in an explicit tree. We transform the hierarchical information in this tree into a novel regularization term based on triplet constraints among the nodes at different levels in the tree. Our method takes advantage of semantic information provided in class labels and outperforms the original t-SNE and two other supervised DR methods in terms of both visual assessment and quality metrics on three classic image datasets: MNIST, Fashion-MNIST and CIFAR10.**

***Index Terms*—Dimensionality Reduction, Visualization, t-SNE, Hierarchical Constraint, Triplet loss.**

I. Introduction

Dimensionality reduction (DR) methods aim at transforming data in a high dimensional (HD) space into a low dimensional (LD) space while preserving relevant meaningful properties of these data. t-distributed stochastic neighbor embedding (t-SNE), one of the most widely used methods, preserves local structures (the neighborhood information) and can reveal the separation of groups of instances in the embedding [1].

Although t-SNE and other state-of-the-art methods provide interesting visualizations for data exploration, they have several important issues. First, in many cases, visualizations given by traditional DR methods are not useful because of the overlap of different groups or the blending of clouds of points. This prevent users to efficiently explore data as they cannot consider each group separately. Second, t-SNE does not preserve global structures in the data. Wattenberg et al. [2] point out common misreadings when interpreting t-SNE plots, in particular that "distances between well-separated clusters in a t-SNE plot may mean nothing". Figure 1a shows an example of three groups, two of them being close to each other and far from the third one. In different t-SNE visualizations, the global structure of these groups is not represented properly. Third, t-SNE may not represent *semantic* information in the LD embedding, as it only preserves neighborhood information



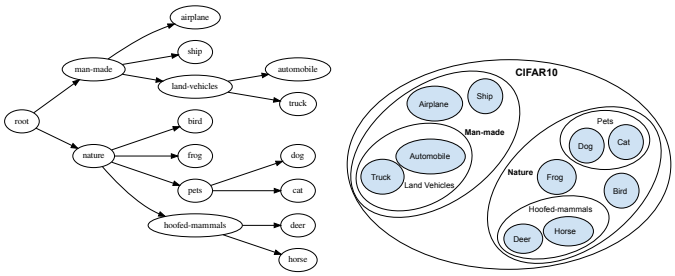
(a) Example where t-SNE does not preserve the distance between separated clusters in several visualizations, reproduced from [2].

(b) Background effect on a t-SNE embedding of CIFAR10.

Fig. 1: Two issues of t-SNE are addressed in this work: (a) there may be a lack of global structure in embeddings and (b) semantic information may not be taken into account.

based on a similarity measure in the HD space. For example, the Euclidean distance between images ($L2$-pixel distance) can put a too strong focus on the background. As an illustration, let us consider the three images  ,  and  from the CIFAR10 dataset. The airplane in the blue sky is around three times closer to the bird in the blue background than to the airplane on a gray background. This effect leads to a poor neighborhood estimation (around 38% accuracy for KNN classification in the LD space with $K = 10$). Figure 1b illustrates this issue with a t-SNE embedding of the CIFAR10 dataset. Images with the same background are placed close together regardless of the object represented in each image.

The goal of this work is to improve the preservation of global structure in the visualization by injecting semantic information in the form of hierarchical constraints, as shown in Fig 2. Hierarchical constraints are used to express expert knowledge about a dataset since groups in t-SNE embeddings do not always match user expectations. For example, the



(a) Hierarchical constraints expressed as a tree by the user.

(b) Hierarchical structure as enforced in the visualization.

Fig. 2: Example of hierarchical constraints to embed human knowledge in a visualization of the CIFAR10 dataset.

hierarchical structure of the CIFAR10 dataset in Figure 2a is not reflected in Figure 1b. Figure 2b shows a more aligned visualization, where child nodes in the hierarchical tree are represented by subgroups in the visualization. Our idea is to construct a semantic hierarchy with the help of the user, expressed by a tree as in Figure 2a, and transform it to a hierarchical structure in the visualization as in Figure 2b.

In order to improve visualizations, one may also rely on higher-level features extracted from a neural network or on a more suitable distance metric. These two approaches may avoid the shortcomings of t-SNE by using better input, but they do not address them directly. Also, it may not be possible to apply them, what further motivated our work. For example, t-SNE can be used with input pairwise (dis)similarities that are encountered in many applications (text mining, geolocation, string comparison, etc.). In these cases, the distance metrics are predefined and the original features are not accessible to benefit from a neural network for feature extraction.

In this paper, we present hierarchical constraints t-SNE (HCt-SNE), a simple and intuitive way to transform hierarchical constraints in the form of a tree into a differentiable regularization term for t-SNE. This term, based on a triplet loss, can be efficiently estimated and jointly optimized with the original objective function of t-SNE (Section III). We use class labels to define the semantic in the dataset and compare our method with other supervised DR methods (reviewed in Section II). Embedding quality is assessed with both a visualization quality metric [3] and the KNN-based score suggested by [4]–[6]. Experiments on MNIST, FASHION-MNIST and CIFAR10 show that HCt-SNE does not only visually improve the embedding, but it also provides a significant gain in terms of KNN score in comparison to the original t-SNE embedding (Section IV). Finally, in Section V, we discuss the foundation of the proposed tree-based loss term and its extension with other types of contrastive constraints.

II. Background and Related Work

t-SNE is a DR method that is efficient at preserving local structure. From pairwise distances in the HD space, it constructs a neighbor graph, which is then transformed into probabilities using Gaussian kernels. Probabilities are constructed similarly in LD with t-distributed kernels. t-SNE adjusts the position of points in the LD space by minimizing the Kullback-Leibler (KL) divergence between these probabilities. As discussed above, it has two major issues: the fact that the global structure is not always faithfully reproduced and the lack of consideration of the semantic. This section reviews four groups of methods that tackle these issues.

The first group of methods addresses the problem of global structure in t-SNE embeddings. Den-SNE [7] is a density-preserving approach that addresses the cluster size issue. Uniform manifold approximation and projection (UMAP) [8] claims to preserve global structures better than t-SNE. Its n_neighbors hyperparameter controls the trade-off between global and local structure, while min_dist controls the appearance of groups in the embedding. Global t-SNE [9] combines the original KL loss with a global cost function, focuses on large distances in both HD and LD spaces, and shows an improvement on small and simple datasets.

The second group of methods contains supervised DR methods that include class labels in the DR process to address the lack of semantic in visualizations for images. This includes neighborhood component analysis [10], UMAP in a supervised setting [8] and class-aware t-SNE [6], which all use class labels to consider the semantic information in the embedding.

The third group includes methods that combine visual analytic techniques to discover hierarchical structures in the embedding, such as hierarchical stochastic neighbor embedding (HSNE) [11]. This interactive method for real-time analysis is used for massive datasets like cytometry data [12]. HSNE incorporates the principle of *Overview-First, Details-On-Demand* by constructing the hierarchical representation of the data based on user's given landmarks at different scales.

The fourth group uses the triplet loss to obtain similar representations for similar points and vice versa. The triplet loss is widely used in deep learning for face recognition [13], image retrieval with deep metric learning [14] or self-supervised visual representation learning [15]. For visualization, the triplet loss can be used to directly update the embedding, such as in t-distributed stochastic triplet embedding (t-STE) [16] and TriMap [17]. t-STE uses a heavy-tailed Student-t kernel (that focuses on local similarities) to measure triplet satisfaction. TriMap uses a custom contrastive loss based on triplet constraints weighted by pairwise distances in the HD space.

Although the above methods enhance t-SNE, none of them solve the lack of semantic and global structure at the same time. Moreover, they do not allow users to express directly the semantic they expect in the visualization, except HSNE. Yet, it only offers an overview of the global structure and a feedback is given on separate sub-parts of the embedding, whereas our method provides a global solution. Our method differs from TriMap and STE in two points. First, we focus on a small number of informative triplets that encode the hierarchy in the input constraints instead of sampling all possible triplets. Second, we combine the property of preserving both global and local structures instead of focusing only on local structure as in t-STE or only on global structure as in TriMap. In the

next section, we present our method that uses class labels and human knowledge to encode hierarchical constraints into a triplet loss. Its goal is to improve global structure and inject semantic information into t-SNE embeddings.

III. Hierarchical Constraints in Visualizations

This section presents HCt-SNE, which integrates hierarchical constraints into t-SNE to make data exploration easier. To improve the readability and the semantic of visualizations, instances in a group are made closer than instances in other groups. The triplet (x, x^+, x^-) with an anchor x, a positive example x^+ and a negative example x^- are used to indicate that x should be closer to x^+ than x^-. Section III-A presents examples for expressing group-level hierarchical constraints using individual-level triplet constraints. Section III-B introduces a new regularization term for t-SNE using triplet loss. Section III-C presents our iterative algorithm with a level-order tree traversal to encode the hierarchy at different tree levels.

A. Motivating Example

Let us consider a hierarchical tree as in Figure 2b, where each node corresponds to a group of instances. The root node is considered to have the highest level of abstraction and contains the entire dataset.When going down the tree, the level of abstraction decreases as groups are split until the leaf nodes, which correspond to classes.The high-level constraints expressed by users are represented by the relationship between groups in the tree (parents, children, and siblings). For that reason, we call them *group-level* constraints. Triplet constraints are constructed for the instances in each group in such a way that the hierarchical relationships in the tree are respected. Let us take a car object  as a running example when processing instances in the *automobile* group.

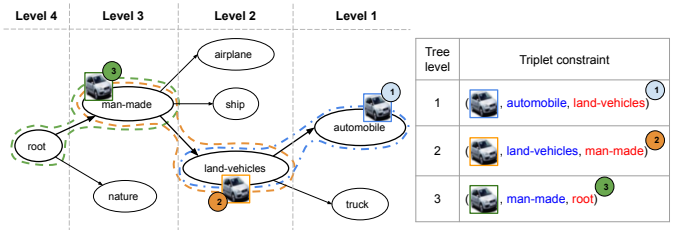


Fig. 3: Examples of triplet constraints to reinforce child-parent relationships. The child and parent groups are annotated on the tree by closed curves with color code. The corresponding triplets are shown on the right.

First, we consider the child-parent relationships illustrated in Figure 3. The *land-vehicles* group includes instances of the *automobile* and *truck* groups. At level 1, the car object  should be closer to other car images in *automobile* rather than being fused into the parent group of *land-vehicles*. This is encoded by the triplet ( , *automobile*, *land-vehicles*) where the group *automobile* can in practice be represented by its centroid. We present here the general idea and discuss this point in greater detail in the next section. At level 2, as *automobile* is a child of *land-vehicles*, the car  is now a member of *land-vehicles* and should be closer to other instances of this group rather than being fused into the parent *man-made* group. By continuing to go up, we end up with the three triplets shown in Figure 3.

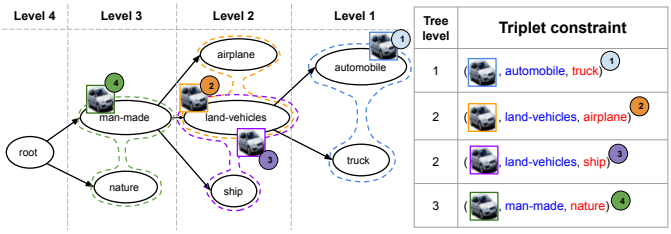


Fig. 4: Examples of triplet constraints to distinguish different child groups in a parent group. Pairs of sibling groups at each level are annotated by closed curves with color code. The corresponding triplets are shown on the right.

Second, we consider the sibling relationships illustrated in Figure 4. At level 1, *automobile* and *truck* are two children of *land-vehicles*. Yet, the car object  should be closer to the other instances of the *automobile* group than to instances of the *truck*group, which leads us to the triplet ( , *automobile*, *truck*). At level 2, the car  now belongs to *land-vehicles* which has two siblings: *airplane* and *ship*. This adds two triplets at this level: ( , *land-vehicles*, *airplane*) and ( , *land-vehicles*, *ship*). Finally, the car  is in *man-made*, whose sibling is *nature*. We thus obtain the fourth triplet ( , *man-made*, *nature*) in Figure 4.

In summary, our idea is that, at each level of the tree, we exploit the child-parent and sibling relationships of each node to construct the individual-level triplet constraints. From this concrete example, we will define more general rules to encode the hierarchical constraints.

B. Encoding the Hierarchical Constraints

Hierarchical Constraints with t-SNE (HCt-SNE) assumes that the user's hierarchical constraints are expressed in the form of a tree, based on class labels and on user prior knowledge about the dataset. First, the group-level hierarchical constraints in the tree are transformed into individual-level triplet constraints. These triplet constraints are then represented by a differentiable regularization term in order to be optimized alongside the objective function of t-SNE.

Let us denote the embedding $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^N$. The triplet constraints are constructed for every instance in each group of the tree. Instead of considering all N^3 possible triplets, we propose to use a triplet of the form $(\boldsymbol{y}_i, G^+, G^-)$, where G^+/G^- denote the group of positive/negative examples for the anchor $\boldsymbol{y}_i$. This triplet indicates that $\boldsymbol{y}_i$ should be closer to G^+ than G^-, where G^+ and G^- are represented by a representative point, here the group centroid (average position of all instances). The number of triplets in the worst case is

$\mathcal{O}(KHN) \ll \mathcal{O}(N^3)$, where K is the number of pairs of sibling nodes and H is the height of the tree. [1]

Let us denote a group and one of its siblings as G_k and $G_{k'}$. The centroids of G_k and its parent group are denoted as $\boldsymbol{c}_k$ and $\boldsymbol{p}_k$. The squared Euclidean distance between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ is denoted as $d(\boldsymbol{y}_i, \boldsymbol{y}_j) = ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2$. The following rules are defined to encode group-level hierarchical constraints.

1) Rule 1: A point $\boldsymbol{y}_i \in G_k$ should be closer to the centroid $\boldsymbol{c}_k$ of its group than to the centroid $\boldsymbol{p}_k$ of its parent group.

This rule means that $\boldsymbol{y}_i$ should be close to other points in G_k rather than being fused in the parent group (see examples in Figure 3). It thus prevents child groups from concentrating on the center of parent group. This rule focuses on the intra-distances within the parent group. The triplet loss for all points in a group G_k according to this rule is

$$\mathcal{L}_{intra} = \frac{1}{|G_k|} \sum_{\boldsymbol{y}_i \in G_k} \Big[d(\boldsymbol{y}_i, \boldsymbol{c}_k) - d(\boldsymbol{y}_i, \boldsymbol{p}_k) + m \cdot d(\boldsymbol{y}_i, \boldsymbol{p}_k) \Big]_+, \tag{1}$$

where $[x]_+ = max(0, x)$ is the maximum operator and $|G_k|$ is the number of instances in G_k. The term $m \cdot d(\boldsymbol{y}_i, \boldsymbol{p}_k)$ plays the role of a margin in the triplet loss, where $m \in [0, 1)$ is called a *relative margin*. If the point $\boldsymbol{y}_i$ violates Rule 1 (when it is too close to $\boldsymbol{p}_k$), the gradient

$$\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{y}_i} = 2\Big[(\boldsymbol{y}_i - \boldsymbol{c}_k) - (1 - m)(\boldsymbol{y}_i - \boldsymbol{p}_k) \Big] \tag{2}$$

moves this point far away from $\boldsymbol{p}_k$. Figure 5 illustrates the gradient without margin ($m = 0$) for the sake of simplicity.

2) Rule 2: A point $\boldsymbol{y}_i \in G_k$ should be closer to the centroid $\boldsymbol{c}_k$ of its group than to the centroid $\boldsymbol{c}_{k'}$ of its sibling group $G_{k'}$.

This rule helps to distinguish between different child groups in a larger parent group (see in Figure 4). It focuses on the inter-distances between child groups and uses the loss

$$\mathcal{L}_{inter} = \frac{1}{|G_k|} \sum_{\boldsymbol{y}_i \in G_k} \Big[d(\boldsymbol{y}_i, \boldsymbol{c}_k) - d(\boldsymbol{y}_i, \boldsymbol{c}_{k'}) + m \cdot d(\boldsymbol{y}_i, \boldsymbol{c}_{k'}) \Big]_+. \tag{3}$$

The gradient for a point $\boldsymbol{y}_i$ that violates the constraint is

$$\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{y}_i} = 2\Big[(\boldsymbol{y}_i - \boldsymbol{c}_k) - (1 - m)(\boldsymbol{y}_i - \boldsymbol{c}_{k'}) \Big]. \tag{4}$$

Figure 6 illustrates the gradient of the point that violates Rule 2 in the case without margin, as in Figure 5.

C. Enforcing Hierarchical Constraints

In Algorithm 1, the loss terms for the individual triplet constraints defined in Eq. 1 and Eq. 3 are used to regularize the original objective function of t-SNE . The proposed iterative algorithm has two steps. In the first step, the position of each point $\boldsymbol{y}_i$ is optimized to minimize the penalty loss (the regularization term) and the KL loss of t-SNE. In the second step, the centroid of each group is updated similarly as in K-Means. The two functions `OptimizeRule1` and `OptimizeRule2` apply Eq. 1 and Eq. 3 for every point $\boldsymbol{y}_i$ in the group G_k and accumulate the loss as well as the gradient of the points that violate the constraints.

[1] In the worst case where all the leaf nodes have the same depth of H, the number of triplets between child nodes and parent nodes is $\mathcal{O}(HN)$, the number of triplets between sibling nodes is $\mathcal{O}(KHN)$, where K is the number of all pairs of sibling nodes. In total, the number of triplets in the worst case is $\mathcal{O}(HN) + \mathcal{O}(KHN) = \mathcal{O}(KHN)$

Algorithm 1: Level-order tree traversal algorithm to integrate hierarchical constraints in t-SNE.

Input : High dimensional data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$,
Original t-SNE embedding $\boldsymbol{Y}_0 = \{\boldsymbol{y}_i\}_{i=1}^N$,
Hierarchical constraints in the form of a tree $\mathcal{T}$,
Weight for each rule $\omega_1 = 0.5$ and $\omega_2 = 0.5$,
Relative margin m, Learning rate η,
Coefficient of the regularization term α.
Output: HCt-SNE embedding $\boldsymbol{Y}$

Initialize $\boldsymbol{Y} = \boldsymbol{Y}_0$
$\mathcal{L}_{intra} = \mathcal{L}_{inter} = 0$ `// Two new loss terms`
$\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{Y}} = \frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{Y}} = \vec{0}$ `// Gradient of each new loss`
for $iter \leftarrow 1$ **to** n_iter **do**
  for *level* $l \leftarrow 1$ **to** $height(\mathcal{T})$ **do**
    `/* Optimize Rule 1 */`
    foreach *node* G_k *at level* l **do**
      $\boldsymbol{c}_k$ = G_k.centroid
      $\boldsymbol{p}_k$ = G_k.parent.centroid
      $loss1 = \texttt{OptimizeRule1}(G_k, \boldsymbol{c}_k, \boldsymbol{p}_k, \boldsymbol{Y}, m)$
      $\mathcal{L}_{intra} = \mathcal{L}_{intra} + \omega_1\, loss1$
      $\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{Y}} = \frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{Y}} + \omega_1 \frac{\partial loss1}{\partial \boldsymbol{Y}}$
    end
    `/* Optimize Rule 2 */`
    foreach *sibling pair* (G_k, $G_{k'}$) *at the same level* l **do**
      $\boldsymbol{c}_k$ = G_k.centroid
      $\boldsymbol{c}_{k'}$ = $G_{k'}$.centroid
      $loss2 = \texttt{OptimizeRule2}(G_k, \boldsymbol{c}_k, \boldsymbol{c}_{k'}, \boldsymbol{Y}, m)$
      $\mathcal{L}_{inter} = \mathcal{L}_{inter} + \omega_2\, loss2$
      $\frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{Y}} = \frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{Y}} + \omega_2 \frac{\partial loss2}{\partial \boldsymbol{Y}}$
    end
  end
  `/* Update new loss and gradient */`
  $\mathcal{L} = KL_{loss} + \alpha\Big(\mathcal{L}_{intra} + \mathcal{L}_{inter}\Big)$
  $\frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}} = \frac{\partial KL_{loss}}{\partial \boldsymbol{Y}} + \alpha\Big(\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{Y}} + \frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{Y}}\Big)$
  $\boldsymbol{Y} = \boldsymbol{Y} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}}$ `// In practice, momentum is used.`
end
return $\boldsymbol{Y}$

In practice, our algorithm can be easily integrated into accelerated variants of t-SNE like Barnes-Hut t-SNE [18] and FIT-t-SNE [19]. The triplet loss has a fast gradient update since centroids remain fixed during the first step of Algorithm 1. Since the embedding of t-SNE is not normalized, the KL loss and the regularization are not well calibrated. The relative margin prevents us from manually tuning the margin value for each dataset. As shown in the following experiments, a relative margin $m = 0.5$ works well for all experimented datasets.

IV. Experiments

In our experiments, three questions are addressed. (1) How are the global structure and the hierarchical information represented in HCt-SNE visualizations? (2) In practice, does our proposed iterative algorithm converge? (3) How do

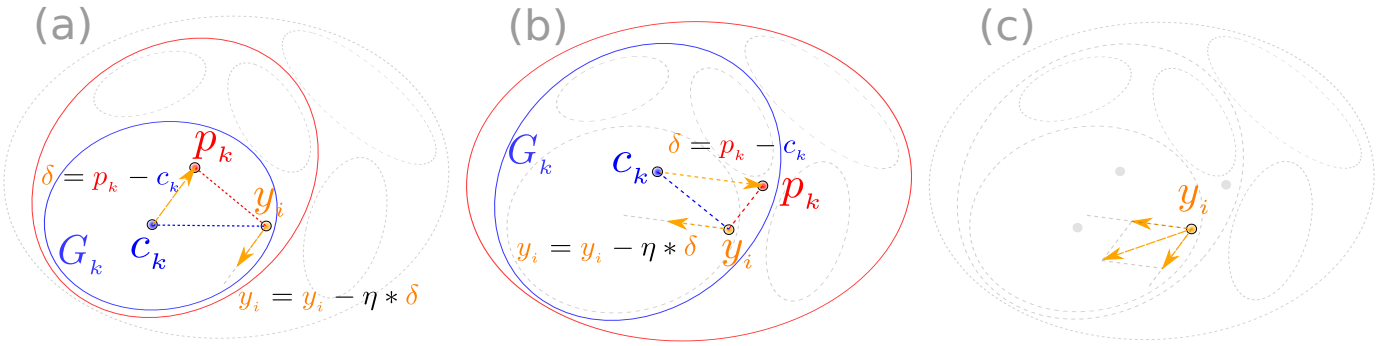


Fig. 5: Illustration of Rule 1 (child-parent relationships) at two levels: (a) a leaf node and its parent and (b) one level higher. For the sake of simplicity and to avoid cluttering the figure, margin m is set to zero to get a simplified gradient $\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{y_i}} = 2(\boldsymbol{p_k} - \boldsymbol{c_k})$. At each level, the instance to constrain $\boldsymbol{y_i}$ is compared to the centroid $\boldsymbol{c_k}$ of its own group and the centroid $\boldsymbol{p_k}$ of its parent group. The adaptation is $-\eta\,\boldsymbol{\delta}$ where $\boldsymbol{\delta} = \boldsymbol{p_k} - \boldsymbol{c_k}$. The sum of the gradients for $\boldsymbol{y_i}$ at the two levels is shown in (c).

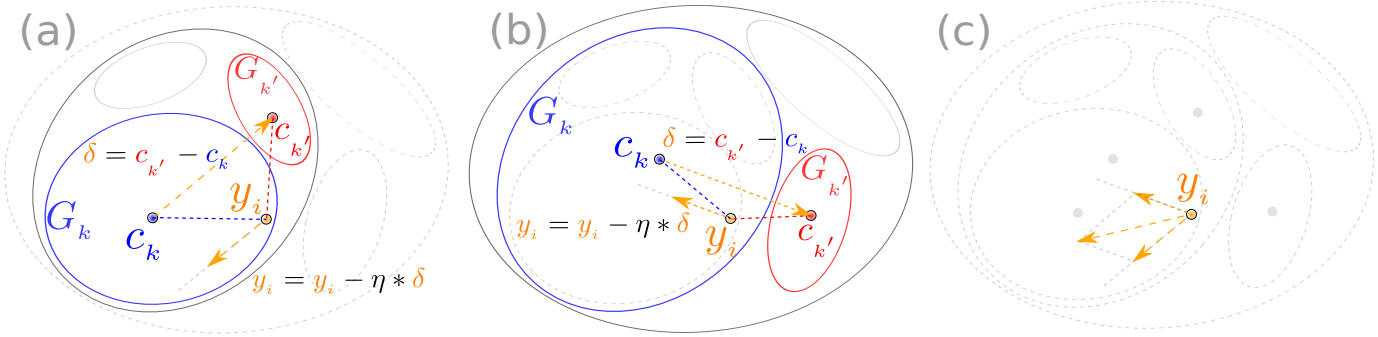


Fig. 6: Illustration of Rule 2 (sibling relationships) with the same conventions as in Figure 5, except that $\boldsymbol{y_i}$ is compared to the centroid $\boldsymbol{c_k}$ of its own group and the centroid $\boldsymbol{c_{k'}}$ of its sibling group. Again, $m = 0$ simplifies the gradient to $\frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{y_i}} = 2(\boldsymbol{c_{k'}} - \boldsymbol{c_k})$. Here, there is only one sibling group, but in more complex cases, one has to sum the contributions of all siblings.

the HCt-SNE embeddings compare with embeddings given by other methods? HCt-SNE is compared with the original unsupervised t-SNE (perplexity=50) and two other supervised DR methods with common hyperparameter values: UMAP in the supervised setting (n_neighbors=10 and min_dist=0.1) and class-aware t-SNE (*cat*-SNE, with its best setting with $\theta = 0.9$ to expand the neighborhood size in the HD space to capture at least 90% of data points with the same label).

A. Experimental Setup

Three standard image datasets MNIST [20], Fashion-MNIST [21] and CIFAR10 [22] are used in our experiments. Input pixels are normalized in $[0, 1]$, and PCA is then applied to keep 95% of the variance. Like t-SNE, HCt-SNE can work with other kinds of data. However, our experiments are performed on image datasets since users can create the hierarchical constraints visually by looking at the images as shown in Sec. III-A.

In our experiments, HCt-SNE takes a t-SNE embedding as the initial state and therefore skips the exaggeration phase. This setting allows us to compare directly the t-SNE visualization without constraints (initial state) and the HCt-SNE visualization with hierarchical constraints. The input hierarchical tree is constructed with the leaf nodes corresponding to the classes in the dataset. The intermediate nodes are created manually according to the desired hierarchical structure of users. For instance, in CIFAR10, *truck* and *automobile* are grouped into *land-vehicles*; this higher-level group is then grouped with *ship* and *airplane* to get *man-made*, etc.

UMAP and cat-SNE are used with the recommended hyperparameters, as our preliminary experiments have shown that other choices provide similar qualitative results for our datasets. HCt-SNE uses the same hyperparameter values as t-SNE and has two additional hyperparameters. First, the relative margin m determines the separation of the groups in the visualization and can be set to 0.5 to make sure the groups are not too close nor too far away. Second, α determines the contribution of the hierarchical constraints to the new loss. α depends on the specific hierarchical tree of each dataset and can be easily tuned by observing the value of the regularization term, and then by trying several values to make sure this term decreases consistently. The reported results are calculated from the following values of α: $7.5{\times}10^{-4}$ for MNIST and Fashion-MNIST, and 5×10^{-3} for CIFAR10. Our implementation is based on openTSNE [23] with Barnes-Hut acceleration.

In order to quantitatively assess the visualizations, three different scores are used. The co-ranking-based score $AUC[R_{NX}]$ [3] measures how well the neighborhood information in the HD space is preserved in the LD space. The KNN-gain score $AUC[G_{NN}]$ [6] measures how much we gain in terms of KNN accuracy when using the embedding in the LD space instead of the original data in the HD space. $AUC[R_{NX}]$ and $AUC[G_{NN}]$ are in the range of $[-1, 1]$, in which 1 is the best, -1 is the worst, and 0 means that there is no gain (or loss) in the neighborhood preservation or the KNN accuracy with the embedding in LD space. It should be noted that, for these scores, a small positive value is acceptable while a negative value is bad. These two metrics have a complexity of $\mathcal{O}\left(N^2 \log N\right)$, where N is the number of instances in the dataset. Because of this complexity, we use a subset of 10k data points for each dataset in order to facilitate the computation of these metrics. It also helps us to make a fair comparison with cat-SNE since cat-SNE is not optimized for large datasets. Finally, the simple KNN score (with $K = 10$) suggested by [4], [5] is used to measure how useful the 2D embedding is for a classification task.

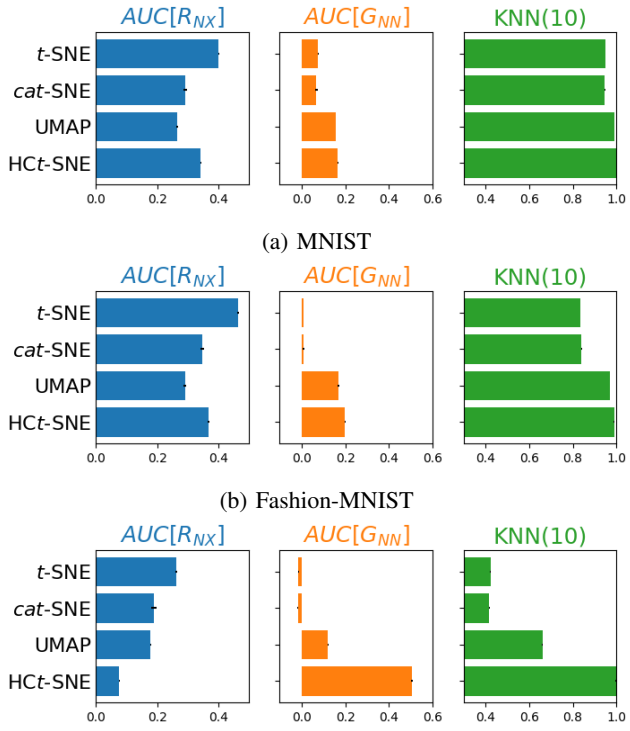
B. Experimental Results

Qualitative results for visual assessment are shown in Table I, where each column corresponds to the visualization of one dataset. The first three rows show the embeddings of t-SNE, cat-SNE, and supervised UMAP, respectively. The fourth row shows the embedding of HCt-SNE, built from the hierarchical tree shown in the fifth row. The last row shows the convergence of the proposed regularization term (in blue) and the new overall loss function (in red).

For MNIST and Fashion-MNIST, HCt-SNE and UMAP can clearly reveal the separation of groups in the visualization. One can also see distinct groups in the visualizations of t-SNE and cat-SNE. However, all competing methods (t-SNE, cat-SNE, and UMAP) fail to reveal distinct groups for CIFAR10. In contrast, HCt-SNE reveals separated groups, gives a global view that helps us to get insights about the relative distances between the instances and about the position of the groups. The visualizations of HCt-SNE can be visually verified with the input hierarchical tree.

Our proposed regularization term is efficient to compute and converges after roughly 50 iterations, as shown in the losses (red lines) of Table I. HCt-SNE is run for 100 iterations to show that the converging state is stable. We do not draw conclusions here from computational time, as it would be unfair since cat-SNE is not optimized for large datasets, whereas HCt-SNE benefits from the computational efficiency of the implementation of openTSNE [2].

Figure 7 shows quantitative results in terms of three metrics (see Section IV-A) for the visualizations in Table I. This figure



(a) MNIST

(b) Fashion-MNIST

(c) CIFAR10

Fig. 7: Average scores of different metrics for three datasets. Each methods are run 10 times with different random initialization. The scores are calculated for each resulting visualizations. The mean values are reported in the bar chart and 95% confidence intervals are shown in the black bars. It should be noted that the scores are stable and confidence intervals are small, which shows that differences are significant.

reports the average scores of 10 different runs with random initialization for 4 experimented methods. In MNIST and Fashion-MNIST, gray-scale images have the same background, and thus there is no background effect. For both datasets, our method performs similarly to supervised UMAP. On the contrary, CIFAR10 contains color images for which the background effect becomes a real problem, as presented in Figure 1b. In a neighborhood, images do not necessarily belong to the same class, what makes the neighborhood preserving score $AUC[R_{NX}]$ and the KNN-based score $AUC[G_{NN}]$ behave oppositely. HCt-SNE preserves semantic information in the class labels and outperforms t-SNE and cat-SNE by a large margin in terms of $AUC[G_{NN}]$. Yet, it has poor $AUC[R_{NX}]$ since it overcomes the background effect and breaks the neighborhood information in HD space. In summary, experiments show that HCt-SNE can integrate user hierarchical constraints to produce useful and informative visualizations.

V. Discussion and Conclusion

This section discusses several technical points in the design of HCt-SNE, its limitations, and several perspectives. The interest of HCt-SNE is three-fold. First, representing the

[2]All our experiments are run on *colab.research.google.com*. For the reader's information, with a dataset of 10k instances, HCtSNE with an initial t-SNE embedding takes less than 35s, UMAP takes around 40s, t-SNE takes 1m20s, and cat-SNE takes around 1h30m. Our implementation and other supplementary materials (visualizations with the full datasets, the effect of the relative margin m) are available online at https://github.com/vu-minh/hc-tsne.

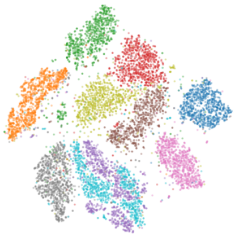
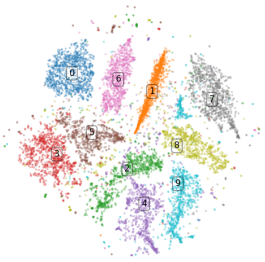
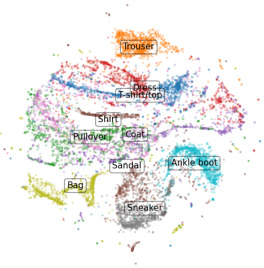
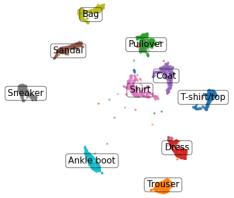
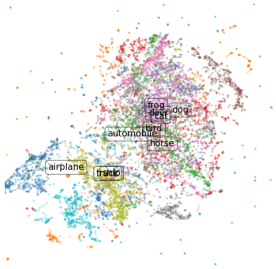
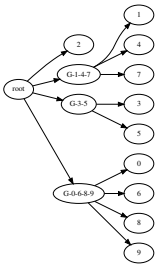
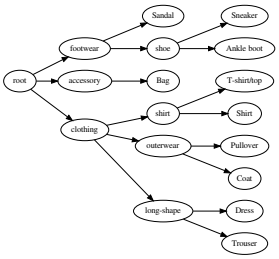
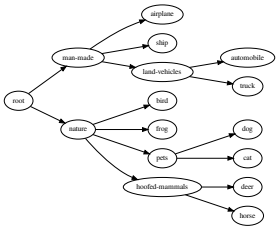
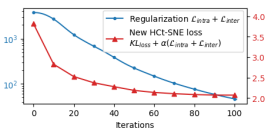
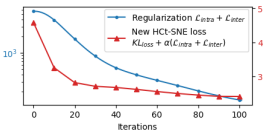
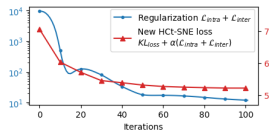
	MNIST (10K)	Fashion-MNIST (10K)	CIFAR10 (10K)
Unsupervised t-SNE			
cat-SNE			
Supervised UMAP			
HCt-SNE (ours)			
Hierarchy for HCt-SNE			
Loss values of HCt-SNE			

TABLE I: Visual assessment for comparing the visualizations of four methods on three datasets. From top to bottom, we show the results of the original unsupervised t-SNE with Barnes-Hut acceleration [18], class-aware t-SNE (cat-SNE) [6], UMAP in a supervised setting [8] and our proposed HCt-SNE. The two last rows show the input hierarchical tree for our method that reflects user knowledge and the convergence of the new regularization term (in blue) and the new overall loss function (in red). The quality of the visualizations is best seen in colors.

hierarchical constraints as a tree is a powerful way to express explicitly the desired global structure. Other supervised DR methods (cat-SNE and UMAP) cannot do this because they only use class labels. Second, the level-order tree traversal algorithm transforms the hierarchy (group level) into triplet constraints (individual level). This is more efficient than naively sampling the N^3 possible triplets. The traversal order does not matter since we update the gradient after visiting all nodes. Third, the triplet loss can be considered as an energy function that minimizes the *compatible settings* (i.e., the anchor and the positive point) and maximizes the *incompatible settings* (i.e., the anchor and the negative point) [24]. This is the same idea as metric learning methods, which try to learn a distance function to make similar points close together and dissimilar points far apart [25]. We can thus extend HCt-SNE by replacing the triplet loss with another contrastive loss like the general function introduced in [26]. The triplet constraints are represented by a regularization term that preserves the global and the hierarchical structure, and thus could also be integrated into other DR methods like UMAP.

In summary, we propose an accessible method for end-users who want to easily explore datasets with the support of a hierarchical representation of constraints for groups of instances. For example, this can be useful when clusters are not clearly separated or even overlap in the visualization computed by t-SNE. It may happen when the distance metric in the HD space is not satisfactory or when the user does not have an appropriate neural network for feature extraction. However, our approach has some limitations. When addressing the problem of global structure in the visualization, we should consider the shape of the groups and the relative distances between them. HCt-SNE currently tackles the second aspect while ignoring the first one. Revealing and preserving the shape of groups with neighborhood embedding methods is still an open problem. Also, in our experiments, we did not consider how to embed new data points. Lastly, HCt-SNE, like t-SNE, is more suitable for visualization but not for general dimensionality reduction tasks.

Our future work will focus on the constraints expressed by users for DR methods. While t-SNE and UMAP are widely used, their results lack global structure preservation, and users have no means to inject their knowledge into the visualization. We plan to work on interactive ways to visually build the hierarchical tree by, e.g., selecting sample images to create nodes. This tree will be passed to HCt-SNE to create a meaningful and useful visualization. Besides, we will also tackle the limitations of our method to expand its usage. For example, the constraints learned from the training set could also help to project new points correctly into their groups.

Acknowledgments

The authors would like to thank the reviewers and our college Géraldin Nanfack for their comments and fruitful discussions on this paper. Viet Minh Vu is supported by a UNamur-CERUNA institutional PhD grant.

References

[1] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *JMLR*, vol. 9, pp. 2579–2605, Nov 2008.
[2] M. Wattenberg *et al.*, "How to use t-SNE effectively," *Distill*, 2016.
[3] J. A. Lee, D. H. Peluffo-Ordóñez, and M. Verleysen, "Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure," *Neurocomputing*, vol. 169, pp. 246–261, 2015.
[4] D. Kobak and P. Berens, "The art of using t-SNE for single-cell transcriptomics," *Nature communications*, vol. 10, no. 5416, pp. 1–14, 2019.
[5] A. C. Belkina *et al.*, "Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets," *Nature communications*, vol. 10, no. 5415, pp. 1–12, 2019.
[6] C. de Bodt *et al.*, "Class-aware t-sne: cat-sne," in *Proc. ESANN*, 2019, pp. 409–414.
[7] A. Narayan, B. Berger, and H. Cho, "Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability," *Nature Biotechnology*, 2021.
[8] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.
[9] Y. Zhou and T. O. Sharpee, "Using global t-SNE to preserve inter-cluster data structure," *bioRxiv*, 2018.
[10] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Proc. NIPS*, 2005, pp. 513–520.
[11] N. Pezzotti, T. Höllt, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "Hierarchical stochastic neighbor embedding," *Computer Graphics Forum*, vol. 35, pp. 21–30, 2016.
[12] V. van Unen, T. Höllt, N. Pezzotti, N. Li, M. J. Reinders, E. Eisemann, F. Koning, A. Vilanova, and B. P. Lelieveldt, "Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types," *Nature communications*, vol. 8, no. 1740, pp. 1–10, 2017.
[13] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, Boston, MA, 2015, pp. 815–823.
[14] W. Ge, W. Huang, D. Dong, and M. R. Scott, "Deep metric learning with hierarchical triplet loss," in *Proc. ECCV*, 2018, pp. 269–285.
[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*, 2020.
[16] L. van der Maaten and K. Weinberger, "Stochastic triplet embedding," in *IEEE International Workshop on Machine Learning for Signal Processing*, Santander, Spain, 2012, pp. 1–6.
[17] E. Amid and M. K. Warmuth, "TriMap: Large-scale Dimensionality Reduction Using Triplets," *arXiv preprint arXiv:1910.00204*, 2019.
[18] L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *JMLR*, vol. 15, pp. 3221–3245, 2014.
[19] G. C. Linderman *et al.*, "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data," *Nature Methods*, vol. 16, no. 3, pp. 243–245, 2019.
[20] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist, 2010.
[21] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747*, 2017.
[22] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.
[23] P. G. Poličar, M. Stražar, and B. Zupan, "openTSNE: a modular python library for t-SNE dimensionality reduction and embedding," *bioRxiv*, 2019.
[24] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," in *Predicting Structured Data*, G. BakIr, T. Hofmann, B. Schölkopf, A. J. Smola, and B. Taskar, Eds. MIT Press, July 2007, pp. 191–246.
[25] B. Kulis *et al.*, "Metric learning: A survey," *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.
[26] S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, and N. Saunshi, "A theoretical analysis of contrastive unsupervised representation learning," in *Proc. ICML*, 2019, pp. 5628–5637.