Making low-dimensional embeddings more informative

Makhin Artem [1] **Kuznetsov Mikhail** [1] **Mumladze Maximelian** [1] **Zelentsov Aleksei** [1]

Abstract

Low-dimensional embeddings might be extremely useful for many applications, such as enhancing the performance of supervised ML models, clustering data, gaining insights from latent features, and exhibiting. However, current state-of-the-art methods sometimes fail to infer many important properties of the data. That's why after low-dimensional transformation some important features of dataset may not preserve. The purpose of this work is to find the ways of solving these problems.

Github repo: https://github.com/Recap-Minutes-of-Meetings/tsne
Presentation file: https://github.com/Recap-Minutes-of-Meetings/tsne/blob/main/preso.pptx

1. Introduction

Low-dimensional embeddings have become an indispensable tool in the field of data analysis. With the vast amounts of high-dimensional datasets being gathered daily, these embeddings help experts explore the structure and relationships encoded within the data. They allow for the identification of general trends, clusters, inter-cluster relationships, as well as extreme-valued samples and outliers.

In natural sciences, such as genomics, low-dimensional embeddings are routinely applied as a first step in data exploration. In core machine learning, they are frequently used as a tool for understanding neural embeddings, such as those given by text or image encoders. The central idea of low-dimensional embeddings is that they approximate the underlying geometry of high-dimensional data. Most high-dimensional data has the property of samples living in a lower dimensional subspace or manifold. Low-dimensional embeddings aim to capture this structure by mapping each sample to a lower dimensional space while preserving important relationships between them.

Although a classical projection onto the first two principal components can provide valuable insights and serve as an initial step of analysis, it often fails to capture the nonlinear regularities that exist in the data. To address this issue, various solutions have been proposed, with the most successful ones utilizing a similarity measure to reconstruct the high-dimensional data in the low-dimensional embedding.

State-of-the-art methods like (van der Maaten & Hinton, 2008), (McInnes et al., 2020), (Tang et al., 2016), and (Artemenkov & Panov, 2020) have improved upon traditional methods like principal component analysis by reconstructing local structures more accurately. However, these methods neglect the distortion of individual cluster sizes and densities in the embedding, which is a major drawback. Another major drawback is that it is impossible to set some condition for low-dimensional embeddings, that's why preservation of any important property of initial dataset is not guaranteed, but sometimes is crucial.

Our research aimed to explore and investigate the newer proposed methods, which are aimed to tackle the abovementioned issues and can be used in certain situations, when certain properties is needed, such as (Kang et al., 2019), (Heiter et al., 2021), (Fischer et al., 2023).

The contributions of this paper are:

- Comprehensive research of applicability of this methods and proof on their concept
- Comparison of results of these methods with the most widespread dimensiolaty reduction methods
- Creating open-source implementation of these methods for further usage in machine learning community

2. Related works

In recent years, embedding high dimensional data into a low dimensional space, particularly 2 or 3 dimensions, has become a crucial tool for unsupervised analysis of modern scientific data. Classical techniques such as (Pearson, 1901), (Torgerson, 1952), (Tenenbaum et al., 2000), (Saul & Roweis, 2003) and (Kohonen, 1990) have focused on maintaining large distances, but high dimensional data typi-

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Alexander Korotin <a.korotin@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

cally lies on a manifold and resembles euclidean space only locally. Therefore, recent research has shifted towards modeling geodesic distances or focusing only on local distances through locally linear embeddings or stochastic neighbor embeddings.

The current state-of-the-art techniques for low-dimensional embeddings, namely (van der Maaten & Hinton, 2008) and (McInnes et al., 2020), focus on correctly modeling local distances, allowing the distortion of long-range distances. They have been successfully applied in various fields, such as genomics and natural language processing. As these techniques usually yield similar embeddings when properly initialized, it is a matter of preference which one to use. Recent theoretical works have confirmed that the clustering revealed by TSNE is correct under certain assumptions about the data.

Several works have proposed interesting approaches in solving the current issues of these algorithms. (Kang et al., 2019), the main idea of which is to factor out prior information by adding parameter to original (van der Maaten & Hinton, 2008), (Heiter et al., 2021), which holds the same idea, but achieves better results by directly operating on the input distance matrices, and (Fischer et al., 2023), which aims to preserve the local densities of clusters.

3. Algorithms and models

We divided our work, together with our experimental setup in tackling 2 problems. The first problem can be formulated as follows "Current state-of-the-art low-dimensional embedding algorithms are not guaranteed to preserve any prior knowledge, which might be important for further applications". The second problem can be forlumated the following way "Current state-of-the-art low-dimensional embedding algorithms in most cases do not preserve relative high-dimensional cluster sizes and densities, which might be important for further applications".

3.1. Datasets

To conduct comprehensive experiments, several datasets were used.

3.1.1. 2D DATA

We generated two 2-dimensional dataset with 3 Gaussian clusters each. The Gaussian cluster had unit variance and were centered at (10, 0),(0, 15), and (-10, 0) respectively. For the first dataset, we drew 300 points from each Gaussian and scaled the spread of the clusters by 1, 2, 4 (i.e., multiply the centered data by this number), respectively. For the second dataset we drew 100, 200, 500, samples from the Gaussians, respectively, keeping the scale the same across clusters.

3.1.2. G3-S

For this dataset we generated 3 Gaussian clusters living in 50 dimensions, each cluster distribution c_i with mean drawn from U(0, 50) (each dimension iid from this uniform) and unit variance. We then draw 200, 400, 600 points from c_1, c_2, c_3, respectively, and scale the spread of the cluster by 2 (i.e., multiply the centered data by 2).

3.1.3. G3-D

For this dataset we generated 3 Gaussian clusters living in 50 dimensions, each cluster distribution c_i with mean drawn from U(0, 50) and unit variance. We then draw 300 points from each of the cluster distributions and scale the spread of the c_1, c_2, c_3 by 2, 4, 8, respectively.

3.1.4. G10-D

To look at data that is not easily projectable, i.e., the inter-cluster distances can be correctly modeled in 2D, for this dataset we generated 10 Gaussian clusters living in 50 dimensions, each cluster distribution c_i again with mean drawn from U(0, 50) and unit variance. We then draw 200 points from each of the cluster distributions and scale the spread of the c_1, ..., c_{10} by 1, ..., 10 respectively.

3.1.5. U5-D

To look at a different distribution and higher dimensional data, we generated 10 Uniform clusters living in 150 dimensions, each cluster distribution ci again with mean drawn from U(0, 50) and unit variance. We then draw 200 points from each of the cluster distributions and scale the spread of the c_1, ..., c_{10} by 1, ..., 10 respectively.

Also, we conducted 2 algorithms of generating new random datasets of certain structure needed for the experiments on problem 1

3.1.6. GEN-1

We consider synthetic data of n = 1000. The data has 14 dimensions in total, where each sample belongs to one of 4 clusters (A1-A4) in dimension 1-8 and one of four clusters in dimension 9-12 (B1-B4). We first draw cluster centers from $\mathcal{N}(0, 2)$ for each of the eight clusters. Then, the feature values for each sample are generated in three steps as in 1

We use Euclidean distance of first 8 dimensions as prior knowledge.

3.1.7. GEN-2

We generate 10-dimensional data of n = 1000 with samples assigned to one of four clusters in the first 4 dimensions, and one of two clusters in dimension 5 and 6. The clusters in the first 4 dimensions are centered at the four unit vectors along

Algorithm 1 GEN-1

1. Pick a cluster a from A1-A4 with probability 0.1, 0.2, 0.3, or 0.4, respectively. Add Gaussian noise to the cluster center a with standard deviation 0.05, 0.1, 0.15, or 0.2, respectively. Noise is drawn and added for each dimension independently.
2. Pick a cluster a from B1-B4 with probability 0.1, 0.2, 0.3, or 0.4, respectively. Add Gaussian noise to the cluster center a with standard deviation 0.05, 0.1, 0.15, or 0.2, respectively. Noise is drawn and added for each dimension independently.
3. The remaining 2 dimensions of every sample are Gaussian noise from $\mathcal{N}(0, 0.1)$.

the four axes of the dimensions. The clusters in the other dimensions are centered at $[\frac{1}{3}; 0]$ and $[0; \frac{1}{3}]$. The remaining four dimensions are Gaussian noise $\mathcal{N}(0, 0.01)$. Each data point is randomly assigned to one cluster of the first four, and one cluster in dimension 5 and 6, and takes the coordinates of the cluster centers plus some Gaussian noise $\mathcal{N}(0, 0.01)$

We use Euclidean distance of first 4 dimensions as prior knowledge.

3.2. Methods

We compared several methods of generating low-dimensional embeddings, and in this subsection is a brief description of them.

First of all, here is the listing of all "classic" algorithms which we compared with. The reader may see their detailed description in cited papers. It includes current state-of-the-art algorihtms for generating low-dimensional embeddings together with classic approaches, which are still highly appcilable to the real-world problems.

- PCA (Pearson, 1901)
- Multidimensional Scaling (Torgerson, 1952)
- Isomap (Tenenbaum et al., 2000)
- Locally Linear Embedding (Saul & Roweis, 2003)
- t-SNE (van der Maaten & Hinton, 2008)
- UMAP (McInnes et al., 2020)

Let's dive a little bit deeper into other algorithms, because they are main interest of this work, and also these algorithms are not common knowledge, even for experienced researcher in machine learning field.

Algorithm 2 dt-SNE

Input: Data X, perplexity k, iterations T, learning rate μ, momentum δ
Output: Embeddings Y
Compute P
Compute γ_{ij}
$Y^{(0)} \leftarrow PCA(X, 2)$
$Y^{(0)} \leftarrow 0.0001 \frac{Y^{(0)}}{std(Y^{(0)})}$
for $t = 1$ **to** T **do**
  Compute Q
  Compute $\frac{\partial KL(P||Q)}{\partial Y}$
  $Y^{(t)} \leftarrow Y^{(t-1)} + \mu \frac{\partial KL(P||Q)}{\partial Y} + \delta(Y^{(t-1)} - Y^{(t-2)})$
end for

3.2.1. DT-SNE

The implementation of this algorithm is similar to basic t-SNE, but contains some modifications. These modifications are

- Scaling factors are computed before everything else, as follows

$$\gamma_{ij} = \frac{(\sigma_i + \sigma_j)^{-2}}{max_{k,l}(\sigma_k + \sigma_l)^{-2}}$$

- Symmetrical sigmas σ_{ij}, which are calculated as

$$\sigma_{ij} = (\frac{1}{2}(\sigma_i + \sigma_j))$$

- Modified way of calculating P-matrix using symmetrical sigmas

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

$$p_{j|i} = \frac{exp(-||x_i - x_j||_2^2/(2\sigma_{ij}^2))}{\sum_{k \neq i} exp(-||x_i - x_k||_2^2/(2\sigma_{ik}^2))}$$

- Modified way of calculating Q-matrix using scaling factors

$$q_{ij} = \frac{(1 + \gamma_{ij}||x_i - x_j||_2^2)^{-1}}{\sum_{k \neq i}(1 + \gamma_{ki}||x_i - x_k||_2^2)^{-1}}$$

Brief description of the algorithm can be found in 2. Full, detailed description can be found in (Fischer et al., 2023). The implementation of this algorithm can be found on our Github page.

3.2.2. JEDI

The implementation of this algorithm is also similar to basic t-SNE, but contains modifications. The idea of this algorithm is to take as input some prior knowledge Z together

Algorithm 3 JEDI

Input: Data X, Factors Z, perplexity k, iterations T, learning rate μ, momentum δ, α, β
Output: Embeddings Y
Compute P
Compute $P^{'}$
$Y^{(0)} \leftarrow PCA(X, 2)$
$Y^{(0)} \leftarrow 0.0001 \frac{Y^{(0)}}{std(Y^{(0)})}$
for $t = 1$ **to** T **do**
  Compute Q
  Compute $\frac{\partial KL(P||Q)}{\partial Y}$
  Compute $\frac{\partial JS^{\alpha}_{\beta}(P^{'}||Q)}{\partial Y}$
  $Y^{(t)} \leftarrow Y^{(t-1)} + \mu(\frac{\partial KL(P||Q)}{\partial Y} - \frac{\partial JS^{\alpha}_{\beta}(P^{'}||Q)}{\partial Y}) + \delta(Y^{(t-1)} - Y^{(t-2)})$
end for

with X, and calculate additional component of optimization objective using it, so that

$$D^X \approx D^Y \not\approx D^Z$$

The additional optimization objective is Jensen-Shannon divergence between $P^{'}$ and Q. So, the total optimization task can be formulated as follows

$$argmin_Y D_{KL}(P||Q) - JS^{\alpha}_{\beta}(P^{'}||Q)$$

Where

$$p^{'}_{ij} = \frac{p^{'}_{i|j} + p^{'}_{j|i}}{2n}$$

$$p^{'}_{j|i} = \frac{exp(-||z_i - z_j||^2_2/(2\sigma^2_i))}{\sum_{k \neq i} exp(-||z_i - z_k||^2_2/(2\sigma^2_i))}$$

And

$$JS^{\alpha}_{\beta}(P^{'}||Q) = \alpha D_{KL}(P^{'}||\beta Q + (1-\beta)P^{'}) + (1-\alpha)D_{KL}(Q||\beta P^{'} + (1-\beta)Q)$$

So, brief description of the algorithm can be found in 3. Full, detailed description can be found in (Heiter et al., 2021). The implementation of this algorithm can be found on our Github page. Also, here it is worth mentioning that for our experiments we used $\alpha = 0.5$, $\beta = 0.95$

4. Experiments and Results

4.1. Experiment 1: dt-SNE cluster density preservation

For this experiment, we perform low-dimension embedding with different algorithms, and see, how cluster density is preserved with certain metrics. We run each experiment 1 time. All computations are made on remote computer with Intel Core i7 - 10700K processor, 32 GB RAM and NVIDIA RTX 2080Ti with 11 GB VRAM.

4.1.1. Metrics

We report Spearman rank correlation between all high- and low-dimensional distances ρ (global reconstruction), correlation between high- and low-dimensional distances of each point with its 100 closest neighbors (local reconstruction), and correlation between radii of balls enclosing the 100 neighbors of each point in high- and low-dimensional space (relative density reconstruction).

4.1.2. Results

The results of this experiment for various algorithms on the abovementioned datasets can be looked up at 1, 2, and 3. As we can see, dt-SNE works significanly better than simple t-SNE in terms of local reconstruction and relative density reconstruction results. Of course, on some datasets, classic methods, such as PCA and MDS work better than anything else, but this issue is connected to relative simplicity of synthetic datasets.

Also, on figures 1 and 2 we can see the visualisation of the fact, that densities were really preserved.

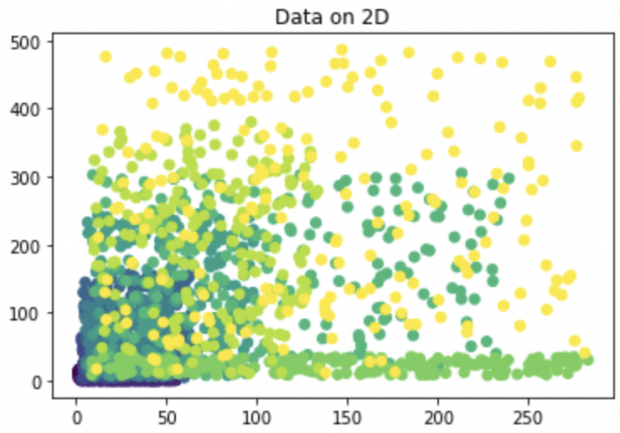

Figure 1. Initial distribution of data on U5-D Dataset

4.2. Experiment 2: JEDI prior knowledge preservation

In this experiment, we tested JEDI, so it can make low-dimensional embeddings given some prior knowledge. In out case, prior knowledge was that our data consists of some number of clusters - and they should be as far as possible from each other on low-dimensional frames. We run each experiment 1 time. All computations are made on remote computer with Intel Core i7 - 10700K processor, 32 GB

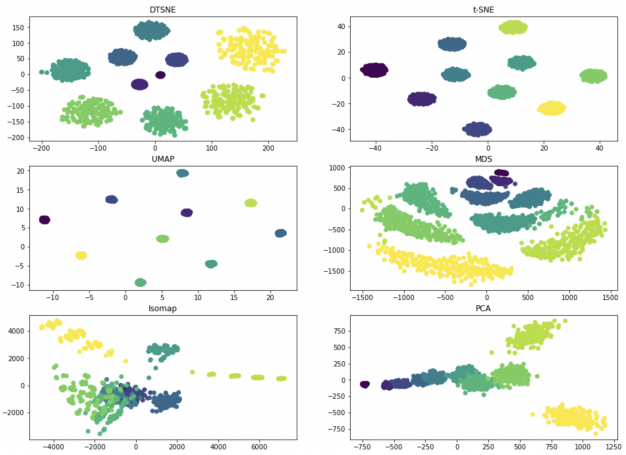

Figure 2. Results of low-dimension transformation with different algorithms

Table 1. Global reconstruction results

DATA SET	2D_1	2D_2	G3-S	G3-D	G10-D	U5-D
LLE	0.442	0.850	0.772	0.447	0.044	-0.068
PCA	1.000	1.000	0.951	0.988	0.961	0.862
ISOMAP	0.990	0.986	0.836	0.950	0.965	0.762
MDS	1.000	0.988	0.827	0.987	0.948	0.865
UMAP	0.597	0.826	0.660	0.813	0.348	0.139
T-SNE	0.755	0.885	0.766	0.641	0.579	0.403
DT-SNE	0.879	0.859	0.737	0.813	0.613	0.616

RAM and NVIDIA RTX 2080Ti with 11 GB VRAM.

4.2.1. METRICS

To objectively quantify how well prior knowledge is factored out from an embedding, we measure the similarity over neighborhoods. For two distance matrices D, D' and neighborhood size k, we define the neighbourhood overlap score (NOS) as

$$NOS(D, D', k) = \frac{1}{k}\frac{1}{n}\sum_{i=1}^{n} |\{kNNof\ \ i\ \in\ \ D\} \cap \{kNNof\ i\ \in\ D'\}|$$

Plotting $NOS(D_X, D_Z,)$ for all neighborhood sizes $k = 1, ..., n$ allows us to asses how well we factor out prior knowledge from an embedding. As the id-line corresponds to a random neighbor encounter, we can measure the area under the NOS curve as a proxy for how well we preserve information, respectively how well we factor out prior knowledge.

4.2.2. RESULTS

The results for this experiments for datasets GEN-1 and GEN-2 can be looked up in table 4. As we can see here, JEDI achieves better results than other algorithms on both datasets.

Table 2. Local reconstruction results

DATA SET	2D_1	2D_2	G3-S	G3-D	G10-D	U5-D
LLE	-0.345	0.785	0.236	-0.480	-0.327	0.084
PCA	1.000	1.000	0.593	0.796	0.801	0.819
ISOMAP	0.966	0.979	0.441	0.556	0.690	0.445
MDS	1.000	0.995	0.727	0.883	0.831	0.839
UMAP	0.543	0.612	0.515	0.226	0.091	0.103
T-SNE	0.570	0.674	0.653	0.256	0.117	0.108
DT-SNE	0.985	0.911	0.716	0.874	0.856	0.848

Table 3. Relative density reconstruction results

DATA SET	2D_1	2D_2	G3-S	G3-D	G10-D	U5-D
LLE	-0.791	0.798	-0.276	-0.644	-0.422	-0.072
PCA	1.000	1.000	0.400	0.856	0.912	0.934
ISOMAP	0.971	0.982	0.274	0.601	0.900	0.624
MDS	1.000	0.999	0.716	0.952	0.947	0.963
UMAP	0.294	0.711	-0.021	0.093	0.003	0.060
T-SNE	0.353	0.707	0.331	0.189	0.062	0.079
DT-SNE	0.985	0.921	0.456	0.930	0.958	0.957

The visualisation on NOS curves is available at figures 3, 4. Also, the results of low-dimensional transformation can be seen at figures 5, 6. As we can see on this figures, JEDI really preserves distances between clusters. Such data can be easily classified with any machine learning classifier

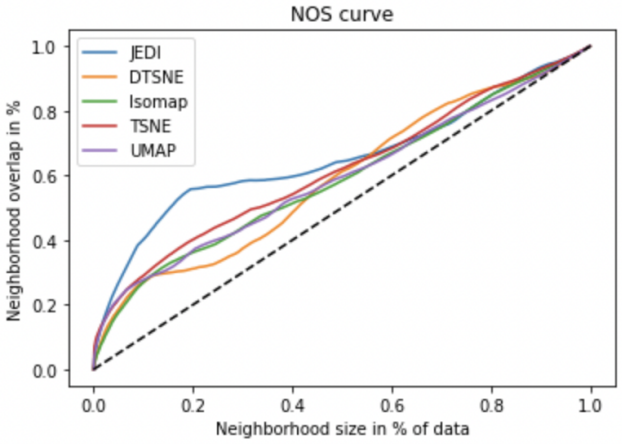

Figure 3. NOS curve on GEN-1 dataset

5. Conclusion

We provided an extensive comparison of various low-dimensional embedding algorithms on our benchmark datasets, and showed that dt-SNE really helps to preserve cluster sizes and densities, as it is said in original paper. Also, we showed that JEDI can work with prior knowledge in order to embed some desired features into low-dimensional embeddings.

Talking about directions of future work, we would like to apply these methods to classifiers on real datasets and see, how

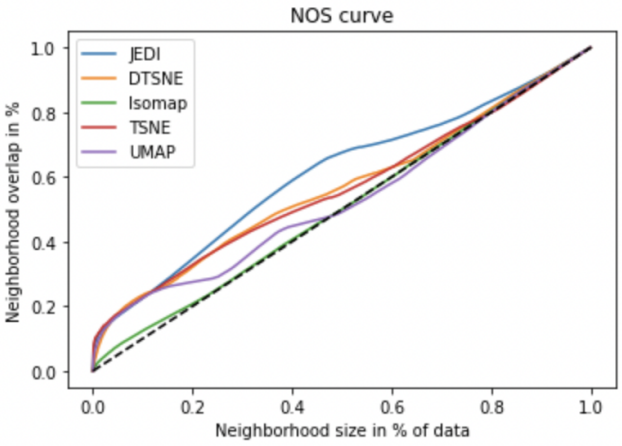

Figure 4. NOS curve on GEN-2 dataset

Table 4. Area under NOS curve

DATA SET	GEN-1	GEN-2
ISOMAP	0.5878	0.5051
UMAP	0.5913	0.5296
T-SNE	0.6149	0.5612
DT-SNE	0.5888	0.5652
JEDI	**0.6545**	**0.6077**

their performance would change given some prior knowledge about train samples, or just preserving its size.

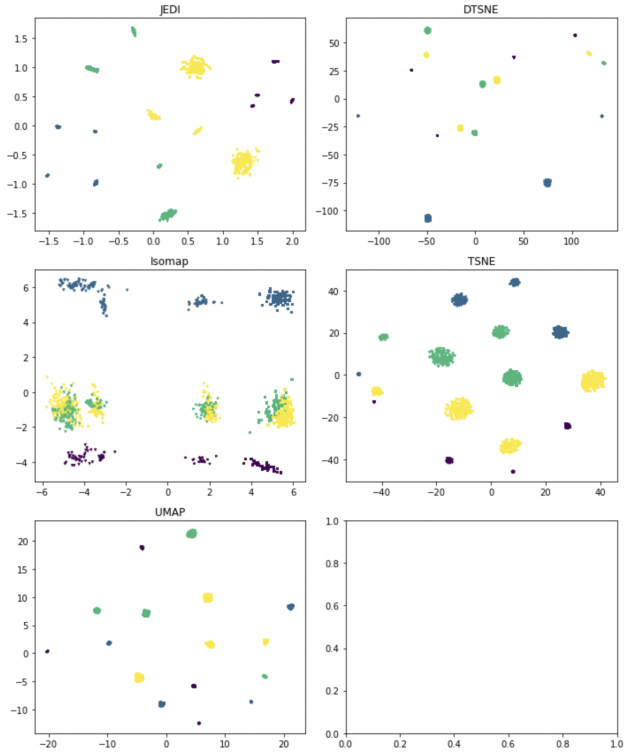

Figure 5. Results of low-dimension transformation on GEN-1 dataset. Different colors are different clusters from prior knowledge

References

Artemenkov, A. and Panov, M. Ncvis: Noise contrastive approach for scalable visualization, 2020.

Fischer, J., Burkholz, R., and Vreeken, J. Preserving local densities in low-dimensional embeddings, 2023.

Heiter, E., Fischer, J., and Vreeken, J. Factoring out prior knowledge from low-dimensional embeddings, 2021.

Kang, B., García, D. G., Lijffijt, J., Santos-Rodríguez, R., and Bie, T. D. Conditional t-sne: Complementary t-sne embeddings through factoring out prior information, 2019.

Kohonen, T. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi: 10.1109/5.58325.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

Pearson, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901. doi: 10.1080/14786440109462720.

Saul, L. K. and Roweis, S. T. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4(Jun):119–155, 2003.

Tang, J., Liu, J., Zhang, M., and Mei, Q. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, apr 2016. doi: 10.1145/2872427.2883041.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Torgerson, W. S. Multidimensional scaling: I. theory and method. *Psychometrika*, (17):401–419, 1952.

van der Maaten, L. and Hinton, G. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

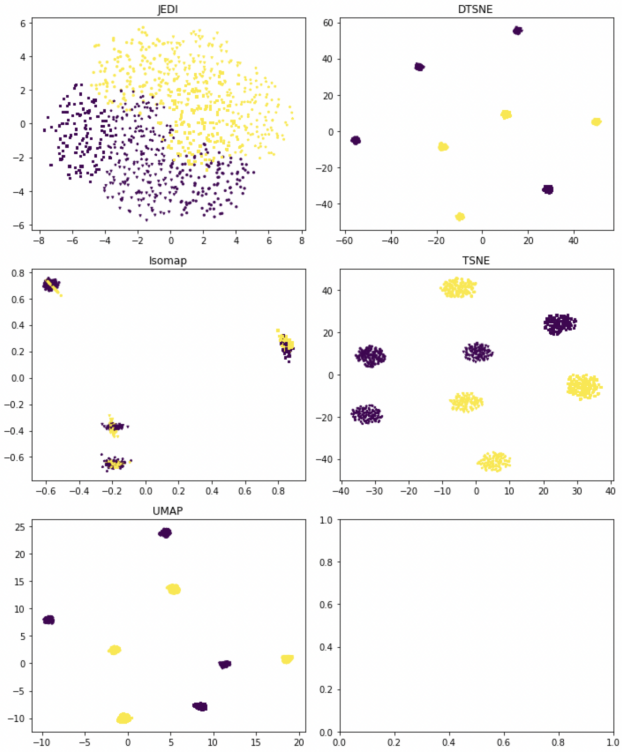

Figure 6. Results of low-dimension transformation on GEN-2 dataset. Different colors are different clusters from prior knowledge

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Makhin Artem (25% of work)

- Preparing the GitHub Repo
- Implementing JEDI
- Presenting the project

Kuznetsov Mikhail (25% of work)

- Implementing dt-SNE
- Adding UMAP to the experiments
- Presenting the project

Mumladze Maximelian (25% of work)

- Implementing most of the experiments
- Making most of the presentation

Zelentsov Aleksei (25% of work)

- Implementing part of JEDI experiment
- Making most of the project report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.
   - ☐ Yes.
   - ☑ No.
   - ☐ Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: None

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.
   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.
   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.
   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.
   - ☐ Yes.
   - ☐ No.
   - ☑ Not applicable.

   Students' comment: Synthetic datasets, code in repository

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.
   - ☐ Yes.
   - ☐ No.
   - ☑ Not applicable.

   Students' comment: Synthetic datasets, code in repository

7. An explanation of how samples were allocated for training, validation and testing is included in the report.
   - ☐ Yes.
   - ☐ No.
   - ☑ Not applicable.

   Students' comment: Unsupervised task, so there is no train/test split

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.
   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.
   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.
    - ☐ Yes.
    - ☐ No.
    - ☑ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

- [x] Yes.
- [ ] No.
- [ ] Not applicable.

Students' comment: None