COMP 551 - Miniproject 4: Reproducibility in ML - the t-SNE paper

Pauline Riviere - Brendon McGuinness - Shubhika Ranjan
Group 62

Reproducibility Summary

Scope of Reproducibility

The work we chose to reproduce was the classic paper that introduced t-SNE (t-Distributed Stochastic Neighbour Embedding) as a method to visualize high-dimensional data on a low dimensional map [1]. The claims of the paper we wanted to test were as follows:

- Performance claim: t-SNE performs better than three other visualization techniques, i.e. Sammon mapping, Isomap and LLE on the MNIST and Olivetti faces dataset, by obtaining a better separation of classes.
- Model components claim: Early Exaggeration and Perplexity play a major role in the efficiency of t-SNE algorithm. PCA as a part of data preprocessing makes the t-SNE algorithm faster and visualization more interpretable.

Methodology

We employed t-SNE implementation as provided in the author's python script on their Github page. We used Google Colab CPU as t-SNE isn't computationnaly expensive for the small size of data sets as used by the authors.

Results

Overall, our results support the main claims of the original paper as we found that t-SNE visualization was more informative and interpretable than the other visualization techniques we tested. As stated by the authors, perplexity and early exaggeration had a great impact on the algorithm performance. PCA didn't seem as critical for processing time and performance as mentioned by the authors. Our t-SNE plot for MNIST data set wasn't exactly similar to the one in the paper: the 7s were split in 3 different clusters, the 5s & 3s and the 4s & 9s were mixed together.

What was easy

The code provided by the authors for t-SNE was easy to run and easy to follow along to be modified.

What was difficult

The original 6,000 images subset of the MNIST data set used by the authors was not available and they didn't mention their preprocessing method. We had to figure out this looking at an already preprocessed 2,500 images subset they included with their code. We randomly selected 6,000 images from the training set of MNIST dataset and applied a similar preprocessing. However, we couldn't be sure that they selected the images randomly or rather based on their quality and that they applied the same preprocessing methods than we did.

Communication with original authors

We had no communication with the original authors. We used their Github page which has an extensive FAQ page.

1 Introduction

The purpose of this project is to reproduce the work from a published paper by using the exact methods described in that paper. One of the main challenges in machine learning is to ensure that published results are sound and reliable. Reproducibility is an essential step in verifying the reliability of research claims. Additionally, reproducibility is an important step to promote open and accessible science so that new results and methodologies can be quickly and soundly converted into being used in practice. The work we chose to reproduce was the classic paper that introduced t-SNE (t-Distributed Stochastic Neighbour Embedding) as a method to visualize high-dimensional data on a low dimensional map [1].

Visualizing high dimensional datasets has many diverse applications such as cancer biology [2], music analysis [3] and bioinformatics [4] and is often the first line of attack in exploring these high dimensional datasets. At the high-level, t-SNE is a nonlinear algorithm that chooses two similarity measures between pairs of points, one for the high dimensional data and one for the low dimensional (often 2-D) embedding. By minimizing the Kullback-Leibler divergence between the vector of similarities between the mappings, it then attempts to construct a 2-D embedding using gradient descent. In this work, we test four of the claims in the t-SNE paper using the code the authors provided while reproducing their results. Futhermore, we run additional tests varying the model's hyperparameters.

2 Scope of reproducibility

The claims of the paper we wanted to test were as follows:

- Claim 1: t-SNE performs better than three other visualization techniques, i.e. Sammon mapping, Isomap and LLE on the MNIST dataset, by obtaining a better separation of classes. This claim is supported by Experiment 1 in figure 1.
- Claim 2: t-SNE performs better than three other visualization techniques, i.e. Sammon mapping, Isomap and LLE on the Olivetti faces dataset. This claim is supported in Experiment 2 in Figure 2.
- Claim 3: Early Exaggeration and Perplexity play a major role in the efficiency of t-SNE algorithm. Therefore, hyperparameter tuning wrt perplexity and early exaggeration needs to examined. This claim is supported in Experiment 3 in Figure 3 and Experiment 4 in Figure 4
- Claim 4: PCA as a part of data preprocessing makes the t-SNE algorithm faster and visualization more interpretable. This claim is supported in Experiment 5 in Figure 5

3 Methodology

We employed t-SNE implementation as provided in the author's python script on their Github page. Additionally the Isomap and LLE visualization was implemented with the help of scikit learn python-package. For the Sammon mapping implementation, we used the algorithm by Tom Pollard found on their Github page.

3.1 Model descriptions

The **t-SNE** model is built on the Stochastic Embedding Neighbor model. The model starts with appliying Principal Component Analysis to remove the uninformative dimensions of the data, which is then transformed from high-dimensional Euclidean distances into conditional probabilities of similarity using a Gaussian probability density. The model computes a similar conditional probability for corresponding points in the low-dimensional space using a Student-t distribution with one degree of freedom. The two probabilities should be equal to have an accurate representation of the high-dimensionality. The objective of the algorithm is to minimize the mismatch between the two using Kullback-Leibler divergence and gradient descent. Additionally, optimization of t-SNE compared to SNE implies to use a simpler version of the gradient by modifying the Kullback-Leibler divergence function (symmetric SNE) and early compression (L2 penalty) and early exaggeration to help with a fast and efficient optimization.

The **Isomap** aka Isometric mapping looks for a lower dimensional representation to sustain the "geodesic distances" between two datapoints. A geodesic distance is a generalization of distance with respect to the curved surfaces. In other words, instead of measuring distance in terms of the Pythagorean theorem-derived Euclidean-distance formula, Isomap optimizes distances along a discovered manifold [5].

The **LLE** or Locally Linear Embeddings employs multiple tangent linear patches to develop a manifold. In other words, it performs PCA on the neighbours locally, that generates a linear hyperplane. This hyperplane is later compared with the results globally to find the best nonlinear embedding. LLE primarily tends to ‘unroll’ or ‘unpack’ the provided dataset structure in distorted fashion, that creates a figure having high density in the center with extending rays [5].

Sammon mapping is used to fit data of dimensionality on nonlinear space along with preserving the inter-pattern distance. The classifiers sensitive responds well with such models where the pairwise distances given by this metric are well-preserved by the projection metric [6].

3.2 Datasets

The **MNIST dataset** contains 70,000 images of hand-written digits labelled with their digit from 0 to 9 and divided in a train set (60,000 instances) and test set (10,000 instances) [1]. As mentioned in the repository, the original black and white images were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field. For our experiments, we didn’t need a train and test dataset. The authors provided a 2,500 subset of the MNIST dataset alongside the implementation. However, the paper claimed that the figures were computed for 6,000 instances and we wanted to test the reproducibility of code on any random MNIST subset. The authors didn’t mention the preprocessing techniques for their experiments. Thus, we examined their file and found the shape was (2500,784) and intensity varied from 0.0 to 1.0. The data was imported, vectorized and normalized from pixel range [0,255] to [0,1].

The **Olivetti faces dataset** was imported directly from sklearn dataset repository. It consists of faces of 40 different people with small changes in frame of reference, large change in their expression, and random inclusion of spectacles, which leads to 400 images of size 92X112 (10,304) pixels [1].
Since the dataset is not as large as MNIST dataset, the entire 400 images were used in the visualization study.The labels here start with 0 to 39, accounting to the 40 different people.

3.3 Hyperparameters

For each hyperparameter of the model, we performed a manual search to find the best hyperparameters. The hyperparameters of the t-SNE model are:

1. Number of Principal Components: the value used in the paper was 30. First, we tried to remove the PCA from the model. Next, we searched over the following values 3, 10, 20, 40, 50, 60, 70, 90 and 100.
2. Perplexity: can be thought of as a smooth measure of the effective number of neighbours. The value used in the paper was 30 and is typically between 5 and 50.
3. Early exaggeration: the value used in the paper was 4 for the first 250 iterations. First, we tried to remove the early exaggeration from the model. Next, we searched over the following values 2, 4, 6, 10 and 30 for the following numbers of first iterations 25, 50, 100, 200 and 500.
4. Initialization of the gradient: The gradient initialization is introduced to exponential decaying sum of previous gradients to optimize the changes in coordinates in the graphical representation at their corresponding iterations of gradient search.
5. Momentum and Learning rate: Momentum assists the algorithm in avoiding poor local minima. The range of momentum in our hyperparameter modeling was 0.1 to 2. Similarly the learning rate is updated in adaptive manner after every iteration. The learning rate examined from 50 to 500.

3.4 Experimental setup and Computational requirements

Fisrtly, we tried to reproduce the figures of the paper using the code provided by the author with the MNIST and Olivetti faces data sets. Then, we tried to remove components of the model (PCA, early exaggeration) and performed a manual search over different values for the hyperparameters (like perplexity, momentum, learning rate, etc) mentioned in the Section above.

All the experiments for MNIST and Olivetti datasets were performed on Google Colab using CPU. The subset of 6,000 images from MNIST dataset, used for reproducing the figures, required 48 minutes to compute t-SNE visualization. Whereas for 2,500 image subset the computational time was around 8 minutes. Therefore, we performed all the hyperparameter-tuning experiments on the 2,500 subset. The exponential increase in computational time, is very much evident which led the authors to report all the experiments for the 6,000 images subset for MNIST dataset. If we were to include the entire 70,000 images MNIST dataset, GPU could help in reducing the computational complexity, but the computational time would still be comparatively very high. Conversely, the Olivetti faces dataset, which is comparatively smaller and requires lesser RAM, was imported entirely and all the experiments were performed on them. These experiments required around 3 minutes for visualizing and executing the t-SNE experiments.

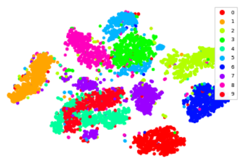
4 Results

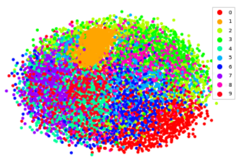
Overall, our results support the main claims of the original paper as we found that t-SNE visualization was more informative and interpretable than the other visualization techniques we tested. As stated by the authors, perplexity and early exaggeration had a great impact on the algorithm performance.

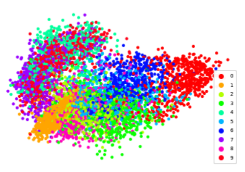
4.1 Results reproducing original paper

4.1.1 Experiment 1: MNIST data set

In Experiment 1, we used the code provided by the author to support the Claim 1 that t-SNE performs better than three other visualization techniques to obtain a separation of the 10 digits classes. As shown in Figure 1, we successfully reproduced the better result of t-SNE over the other methods obtained in the paper. However, our t-SNE plot for MNIST data set wasn't exactly similar to the one in the paper: the 7s were split in 3 different clusters, the 5s & 3s and the 4s & 9s were mixed together.


(a) Visualization by t-SNE


(b) Visualization by Sammon mapping


(c) Visualization by Isomap

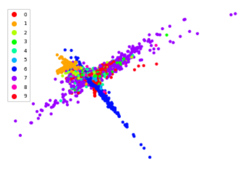
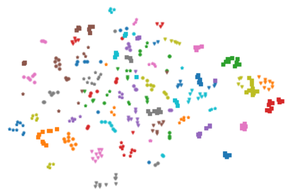
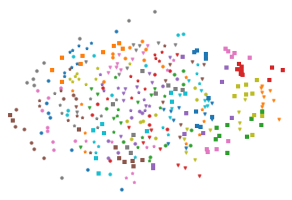

(d) Visualization by LLE

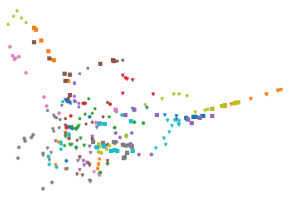
Figure 1: Visualizations of 6,000 handwritten digits from the MNIST data set

4.1.2 Experiment 2: Olivetti faces dataset

In Experiment 2, we used the code provided to test Claim 2, i.e. that t-SNE performs better than the other three visualization techniques for the Olivetti Faces dataset. As shown in Figure 2, our results support Claim 2 of the paper. However, our t-SNE plot was not as similar or clustered as distinctly in the plot shown in the original work. This may be due to non-convexity of the t-SNE algorithm's cost function. Additionally, our Isomap visualization looked quite differnt from what was provided in the paper even when selecting various hyperparameters to run the model with.


(a) Visualization by t-SNE


(b) Visualization by Sammon mapping


(c) Visualization by Isomap

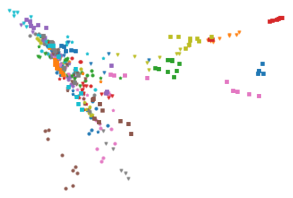
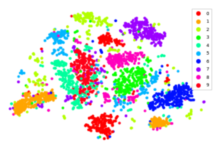

(d) Visualization by LLE

Figure 2: Visualizations of Olivetti faces data set

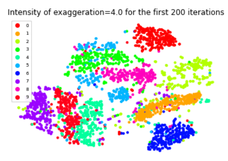
4.2 Results beyond original paper - ablation study

4.2.1 Experiment 3: Early exaggeration and tuning

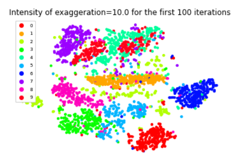
Without early exaggeration, no clusters were identified for 0s, 2s and 7s. Using a high exaggeration value for a high number of iterations, all instances were mixed. The best result was observed for an exaggeration of 4 for the first 200 iterations (9s & 4s were not mixed, 7s not split).



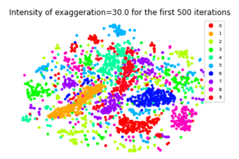
(a) t-SNE without early exaggeration



(b) t-SNE with exaggeration of 4 for the first 200 iterations



(c) t-SNE with exaggeration of 10 for the first 100 iterations

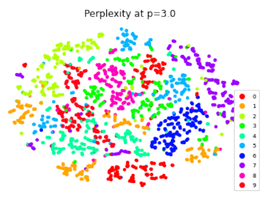


(d) t-SNE with exaggeration of 30 for the first 500 iterations

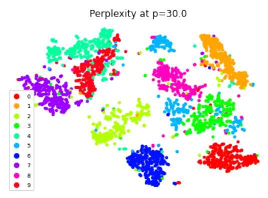
Figure 3: Visualizations of the MNIST data set with different settings of early exaggeration in t-SNE visualization model

4.2.2 Experiment 4: Perplexity

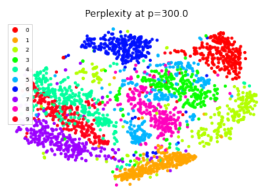
We ran t-SNE with perplexity values of 3.0, 30.0 (what was used in the paper) and 300.0. For the visualization with 3.0 perplexity the clusters are very loose. A perplexity value of 300.0 was also less clustered than a perplexity of 30.0. The paper mentioned values of 5-50 being the operational range and we saw that for those values the clusters looked a lot more distinct.



(a) t-SNE with perplexity of 3.0



(b) t-SNE with perplexity of 30.0

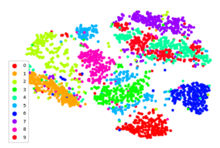


(c) t-SNE with perplexity of 300.0

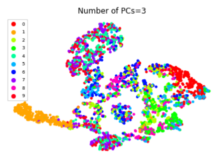
Figure 4: Different perplexity values on the MNIST dataset

4.2.3 Experiment 5: PCA ablation and tuning

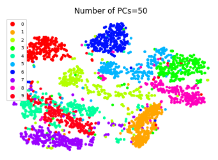
Processing time with or without PCA was similar (7 min 43 sec and 7 min 46 sec, respectively). Without PCA, the digits were less efficiently grouped within their labels, i.e. more digits were mixed and except for 0s and 6s could not be seen as independent clusters. When the number of PCs was set to a very low value, no clusters were identified. With a high number of PCs, the clustering was also less effective, i.e. 4s & 9s and 5s & 3s were mixed together and the 2s and the 7s were not grouped. The best result was observed with 50 PCs (7s were not split, 3s & 5s not mixed).



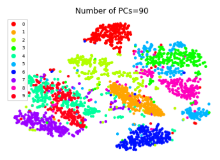
(a) Visualization by t-SNE with no PCA



(b) Visualization by t-SNE with 3 PCs



(c) Visualization by t-SNE with 50 PCs



(d) Visualization by t-SNE with 90 PCs

Figure 5: Visualizations of the MNIST data set with different number of PCs

4.2.4 Experiment 6: Effect of momentum and learning rate

The optimum momentum defined by the author for these experiments were 0.5 (for gradient iterations less than 250) and 0.8 (for iterations more than 250). If we increase the momentum, the visualization becomes less interpretable and we can observe overlapping as shown in figure 6b for MNIST dataset and in figure 7b for Olivetti faces. The learning rate also affects the algorithm. As we can notice in figure 8 for MNIST data and figure 9 for Olivetti faces data, that by increasing the learning rate the scattered-nature of visualization reduces. Although the author mentions learning rate to be 100 in the paper, the github script provided works on learning rate equal to 500.

4.2.5 Experiment 7: Initialization of gradient

One of the weaknesses of the t-SNE algorithm is that its cost function is non-convex, whereas most other state-of-the-art visualization techniques including Isomap and LLE have convex cost functions. This means that the constructed solutions will in part depend on the initial random configuration of map points. We changed the variance of the normal distribution the initialization points were chosen from as well as pulled them from a random uniform distribution with $max = 1$ and $min = -1$ in order to test if the clustering from the constructed solution was sensitive to the initialization. For variances 10 times higher and lower we did not see much differences in the visualization the model outputted as shown in Figure 10a and 10b. Once we started initializing our model with very large variances (Figure 10c), we noticed a difference in the model's visualization likely due to the lack of strong repulsion of near dissimilar points that is there when the points are initialized with a tighter distribution. For the uniform distribution, we still observed the model outputting distinct clusters however they appeared slightly less tight than when normally distributed.

5 Discussion

Overall, our experimental results support the main claims of the paper, that is the better performance of t-SNE over other visualization methods and the importance of perplexity and early exaggeration. We didn't find that PCA improved the processing time or dramatically affected the performance of the algorithm.

The main strength of our work is that we tested extensively the hyperparameters of the model. We also acknowledge the limits: the hyperparameters were manually tuned one at a time. A more sophisticated grid search using combinations of hyperparameters (for example, modify learning rate and perplexity at the same time) could yield more reliable outcome. However, this was difficult because the output of the cost function didn't directly correspond to the quality of the visualization when compared to other visualizations with a different set of hyperparameters. In the future, using some metric of visualization or clustering quality would improve our hyperparameter search.

5.1 What was easy

The code to run t-SNE was available on the author's github and was well documented. This made following the code very easy to work and experiment with. In the paper, the mathematical details were outlined quite clearly and the pseudocode included was easy to follow along.

5.2 What was difficult

We noticed in the code provided by the author that the values of the hyperparameters were not the same as those used in the paper (for example the number of PCs was set to 50 by default instead of 30 mentioned in the paper). However, the code was easy to read and it wasn't difficult to find the correct line where a hyperparameter needed to be modified.

The authors mentioned they used 6,000 images from the MNIST data set but didn't provide informations on the way they selected these from the 70,000 images of the data set and on the preprocessing they performed on these images before t-SNE. A .txt file with the data for 2,500 MNIST images was enclosed in the folder containing their code. We used this file to understand the preprocessing using the shape (2500, 784) and minimum and maximum values (0.0 and 1.0). Therefore, we concluded that the data had been vectorized from 28*28 to 1*784 and normalize from 0.0-255.0 to 0.0-1.0. We applied these transformations to the MNIST data set and randomly selected 6,000 images. However, we didn't obtain a similar separation of labels as in the paper as 4/10 classes were mixed together, despite using their code and hyperparameters values. This could come from the selection of the images for the paper: authors may have selected the more representative images from the data set while we randomly selected 6,000 images.

For Sammon mapping, LLE and Isomap, the authors do not mention which code they used. Fortunately, LLE and Isomap are implemented in the scikit learn package. For Sammon mapping, we found an implementation online. For LLE and Isomap, the authors mention that they only visualize datapoints that correspond to vertices in the largest connected component of the neighborhood graph. However, they didn't expand on the threshold they use to define this largest connected component.

5.3 Communication with original authors

We had no communication with the original authors. We used their Github page which has an extensive FAQ page.

References

[1] Maten and Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008). ISSN: 2579-2605. URL: `https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf`.

[2] Walid M Abdelmoula et al. “Data-driven identification of prognostic tumor subpopulations using spatially mapped t-SNE of mass spectrometry imaging data”. In: *Proceedings of the National Academy of Sciences* 113.43 (2016), pp. 12244–12249.

[3] Philippe Hamel and Douglas Eck. “Learning features from music audio with deep belief networks.” In: *ISMIR*. Vol. 10. Citeseer. 2010, pp. 339–344.

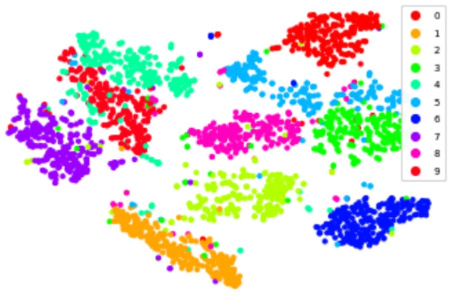
[4] Alexander Platzer. “Visualization of SNPs with t-SNE”. In: *PloS one* 8.2 (2013), e56883.

[5] Andre Ye. “Manifold Learning [T-SNE, LLE, Isomap,] Made Easy”. In: 2020. URL: `https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183`.

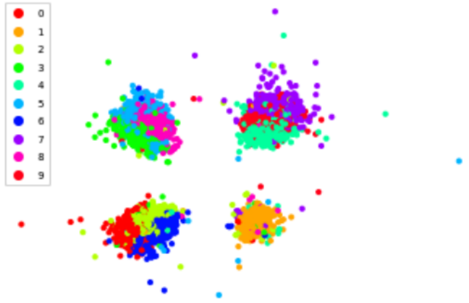
[6] Duin Ridder. “Sammon’s mapping using neural networks: A comparison”. In: *Pattern Recognition Letters* 18 (1997), pp. 1307–1316.

6 Appendix

6.1 Hyperparameter: Momentum



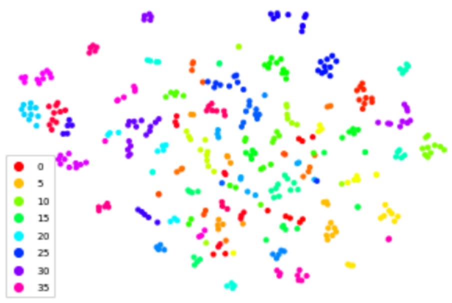
(a) Visualization by t-SNE with momentum 0.1 to 0.5



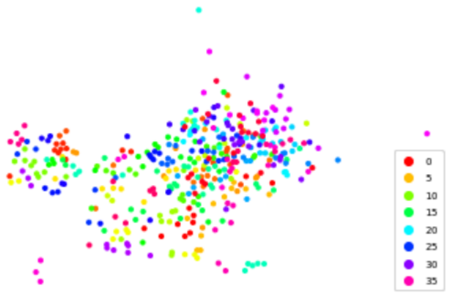
(b) Visualization by t-SNE with momentum 0.1 to 1

Figure 6: Different Momentum values on the MNIST dataset

6.2 Hyperparameter: Momentum



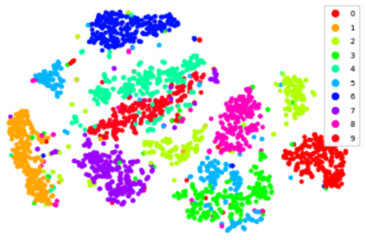
(a) Visualization by t-SNE with momentum 0.1 to 0.5



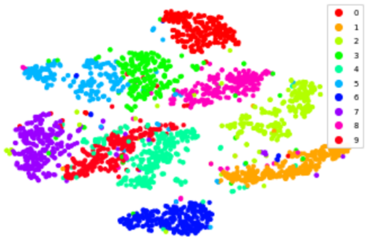
(b) Visualization by t-SNE with momentum 0.1 to 1

Figure 7: Different Momentum values on the Olivetti dataset

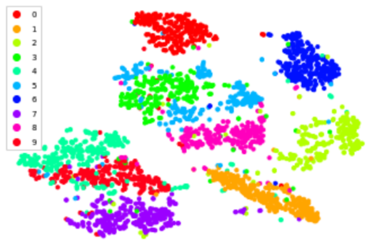
6.3 Hyperparameter: Learning Rate



(a) Visualization by t-SNE with learning rate of 50



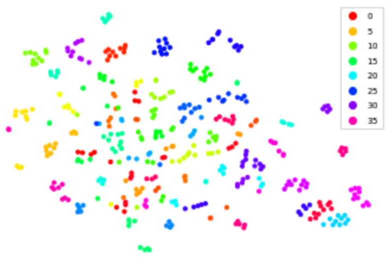
(b) Visualization by t-SNE with learning rate of 100



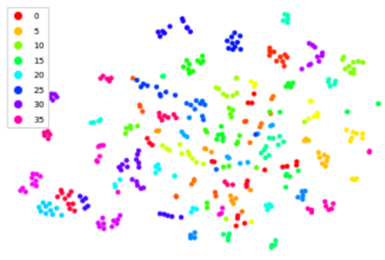
(c) Visualization by t-SNE with learning rate of 250

Figure 8: Different Learning rate values on the MNIST dataset

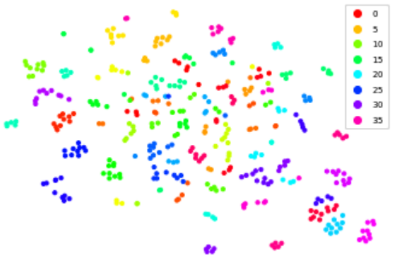
230 ## 6.4 Hyperparameter: Learning Rate



(a) Visualization by t-SNE with learning rate of 50



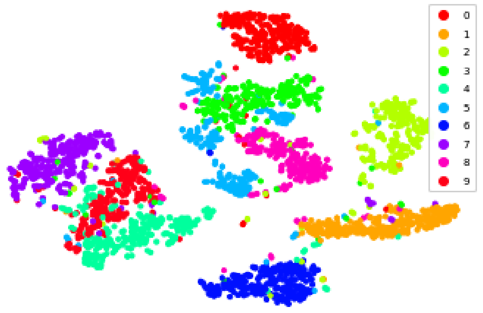
(b) Visualization by t-SNE with learning rate of 100



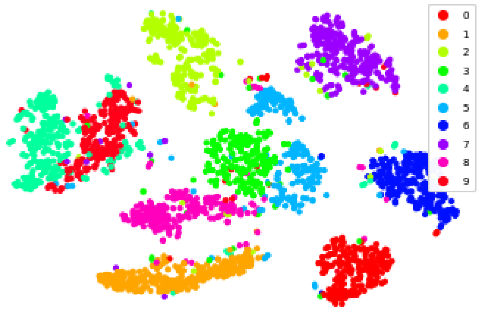
(c) Visualization by t-SNE with learning rate of 250

Figure 9: Different Learning rate values on the Olivetti face dataset

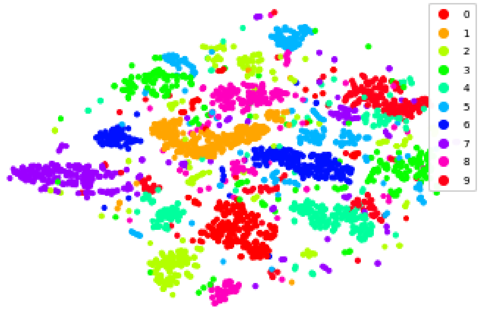
231 ## 6.5 Hyperparameter: Initialization of gradient



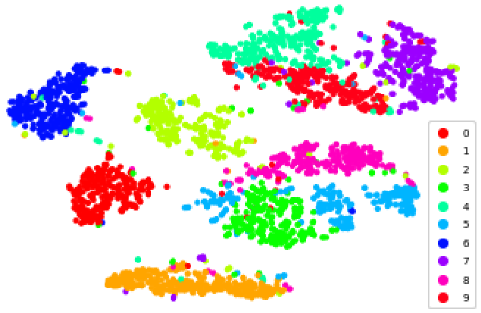
(a) Visualization by t-SNE with normal random initialization with $\mu = 0$ and $\sigma^2 = 0.1$



(b) Visualization by t-SNE with normal random initialization with $\mu = 0$ and $\sigma^2 = 10$



(c) Visualization by t-SNE with normal random initialization with $\mu = 0$ and $\sigma^2 = 30$



(d) Visualization by t-SNE with uniform random initialization from -1.0 to 1.0

Figure 10: Visualizations of the MNIST data set with different initialization