International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 19s, 2025
https://theaspd.com/index.php

Dimensionality Reduction For Data Visualization By T-Sne With Agar Dataset

[1]**CH. Jyothi,** [2]**Dr. M.Raja Sekar**
[1]Department of Microbiology, St. Ann's College for Women,Mehdipatnam, Hyderabad, Telangana.
[2]Department of Computer Science Engineering (Cyber Security, Data Science, Artificial Intelligence and Data Science), VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana.

Abstract:
t-SNE (t-distributed Stochastic Neighbor Embedding) is an unsupervised, non-linear dimensionality reduction technique primarily used for visualizing and exploring high-dimensional data. Unlike PCA (Principal Component Analysis), which is a linear technique that focuses on preserving global structure and maximizing variance, t-SNE is designed to preserve local structure – that is, the small pairwise distances or similarities between nearby points in the high-dimensional space.While PCA provides a clear picture of variance and structure in the data, t-SNE gives you a feel or intuition for how data points relate locally. It has become a widely used tool in machine learning and bioinformatics for exploring hidden patterns, such as clusters of similar samples in high-dimensional biological data (e.g., gene expression or microbial morphology).

Keywords: *PCA, MNIST dataset, Dimensionality reduction, Geometrical interpretation, Mathematical Interpretation, t-SNE, embedding, stochastic, perplexity*

INTRODUCTION

t-distributed Stochastic Neighbor Embedding (t-SNE) is a powerful **machine learning algorithm** widely used for **visualizing high-dimensional data**. Introduced by **Laurens van der Maaten and Geoffrey Hinton in 2008**, t-SNE builds upon the earlier Stochastic Neighbor Embedding (SNE) framework to produce clear and intuitive **2D or 3D visualizations** of complex datasets [1, 2, 3]. In recent years, it has become a **state-of-the-art tool** for **exploratory data analysis**, particularly in applications like computer vision, bioinformatics, and microbiology. One such application is the **AGAR dataset—**a large-scale annotated image dataset of microbial colonies designed for computer vision tasks. It contains over **18,000 high-resolution Petri dish images** with detailed bounding box annotations for **336,000+ microbial colonies**across **five species**. Each colony can be described by **high-dimensional morphological features[4,5,6]**, such as size, shape, texture, and color—making the AGAR dataset a perfect candidate for t-SNE-based analysis and visualization.

Ii. Why use t-sne for agar?

The AGAR dataset involves **complex variations in microbial colony morphology** that are difficult to interpret in raw form. To address this, **t-SNE is employed to reduce the dimensionality of extracted features** (either handcrafted or deep learning-based), allowing researchers to visualize and interpret **hidden structures or clusters** in the data[7,8,9].

t-SNE works by modeling**pairwise similarities** between data points in high-dimensional space and then attempting to preserve these **local relationships** in a lower-dimensional (typically 2D) space. It uses **probabilistic techniques** and minimizes the **Kullback–Leibler (KL) divergence** between high-dimensional and low-dimensional similarity distributions[10].

Applied to AGAR, t-SNE:

- Reveals **clusters of colonies with similar morphology**, often corresponding to the same species.
- Highlights **intra-species variability** and **inter-species similarities**.

- Provides an intuitive visualization that helps identify overlapping phenotypes, rare colony shapes, or imaging anomalies.

Iii. Pca vs. T-sne on agar data

While **Principal Component Analysis (PCA)** is a common baseline for dimensionality reduction, it is a **linear method** that focuses on **preserving global structure** by projecting data along the directions of **maximum variance**. However, PCA often fails to capture **nonlinear patterns** in data such as microbial colony shapes, especially when visualized in only two dimensions.

For instance, if we apply PCA to the AGAR dataset's morphological features, we may observe overlapping clusters and poorly separated species. In contrast, **t-SNE can uncover clear clusters and subtle structural differences** due to its **nonlinear nature** and emphasis on **local neighborhoods**.

Ii. Neighborhood of point embedding

So, Lets understand two concepts as we have two concepts in our t-SNE which are stochastic and neighborhood embedding. First we will try to understand the terms neighborhood and embedding. Imagine in my d dimensional space (high dimensional space) and also ensure that d is large enough [10, 11, and 12]. Assume that red points indicate zeros and blue points indicate ones. If you take our MNIST dataset, we have 784 dimensions and all our zeros are in one region which is close to each other and all of ones are in another region which is close to each other in 784 dimensional spaces. Let's try to define neighborhood of a red point shown in Figure2. So let's say we are taking this point. So the neighborhood of this point is typically those points which are close to it[13, 14, 15]. Of course we could have some points which are not in the neighborhood. Suppose I could have some points here. If the central point is Xi(red point) as shown in Fig.2 So the neighborhood of the red point in Figure2 is typically those points which are close to it.

Neighborhood N(Xi)={Xj such that Xi and Xj are geometrically close}

Geometrically close means, if the distance between these points if you is very less.

$||Xi-Xj||^2 = dist^2$

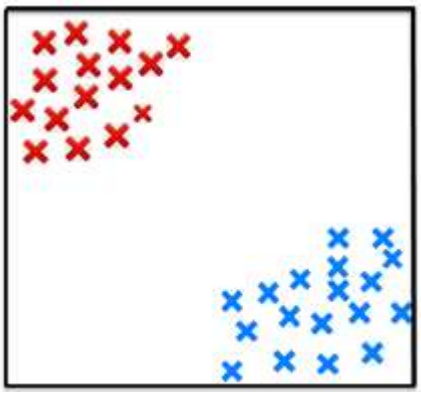

Figure1: Data in d dimensions

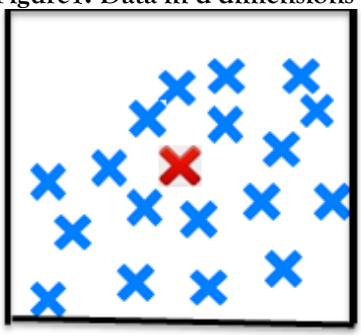

Figure2: Neighborhood of red point

So the neighborhood of the red point in Figure2 is typically those points which are close to it[16, 17, 18].

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 19s, 2025
https://theaspd.com/index.php

Neighborhood N(Xi)={Xj such that Xi and Xj are geometrically close}
Geometrically close means, if the distance between these points if you is very less.
$||Xi-Xj||^2 = dist^2$

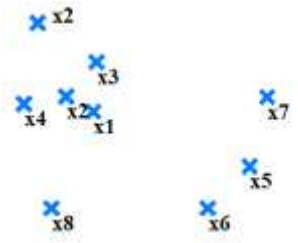

Figure3: Neighborhood of a point

N(X1)={X2, X3,X4} does not contain X8 and X9 as they are far away from X1. This is the meaning of neighborhood.

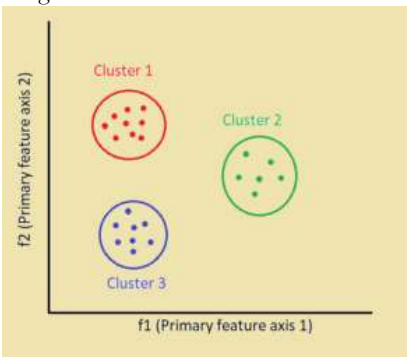

Figure4: Neighborhood with clusters

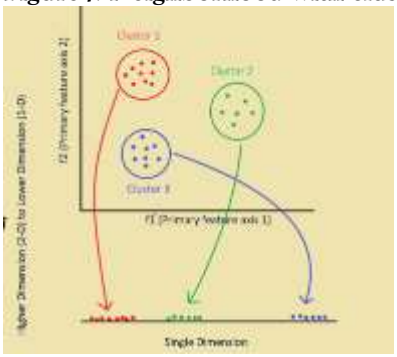

Figure5: Embedding

Now, let's understand what is embedding means and embedding is very interesting idea. So, imagine if I have d dimensional space and of course, what is the objective our dimensional reduction and we would like to get these higher dimensional data to two dimensional or three dimensional, so that I can visualize. Let's assume I have dataset like in figure4, what does embedding means, embedding basically means, for every point Xi in the higher dimensional space and we are finding Xi' in the lower dimensional space. If we find a corresponding point in the low dimensional space such concept is called as embedding[19, 20].

Iii. How t-sne work?

As it is mentioned that t-SNE preserves the local structures of the data while converting from higher dimensions to lower dimensions. That's the point where the term the terms neighborhood and embedding are useful. In the figure4, we can see that there are three clusters in the higher dimensional space (2D in this case). We are calling them clusters as the intra cluster distances are very small. What t-SNE does while reducing to lower (1 dimensional in this case) is that it embeds the points into lower dimension while

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 19s, 2025
https://theaspd.com/index.php

preserving the distances like the higher dimension points (2D points) are projected to the lower dimension(1-D) but the intra cluster distances are similar but t-SNE does not promise to preserve the inter-cluster distances. This process discussed has highly mathematical base to solve the optimization. In layman's term, t-SNE uses a symmetric probability density function in both higher and lower dimension to preserve the neighborhood. Although due to the curse of dimensionality, the points in the higher dimensional space tend to get crowded in lower dimensional space, causing the crowding problem. To solve the crowding problem t-SNE uses the student's t-distribution. Again how t-distribution solves the problem has rigorous mathematics behind it, but t-distribution promises that it will try best to preserve the neighborhood by means of probability (stochasticity). That's why it is t-distributed and stochastic[21].

Iv. Crowding problem

We said that t-SNE literally tries to preserve the distances in a neighborhood N. This could create problems. Let's assume I havetwo dimensional data and we have to project this into one dimensional using t-SNE.

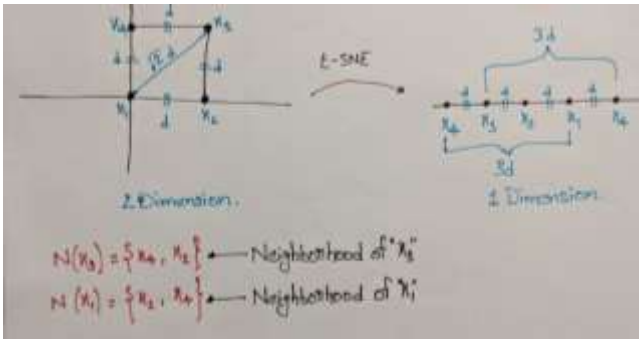

Figure6: Crowding Problem

Suppose our four points are x1, x2, x3 and x4. Let's assume that they are corners of a square. If we have four points like this in 2D space which are at equal distant. As they are at equal distance, they are forming basically a square. All these are vertices of the square and we want to embed them into 1D Space. Let's assume we want to embed our x1, we will place our x2 such that the distance between x1 and x2 is d as shown in Fig.6

Figure:6 Two points with equal distance in one dimensional space

While placing x2 in one dimensional space, we must make sure that the distance between them is d. We have embedded x1 and x2 in one dimensional space. Now consider x3, the good thing is x3 is of the distance of d from x2, so we can embed x3 as shown in figure 7.

Figure7: Three points in one dimensional space

Observe that x1 and x3 are not in a neighborhood.N(x1)={x2,x4} not x3 because this distance is more than d which is two times of square root of d. Let's assume we are defining neighborhood using distance d. We need to preserve the distance between x1 and x3, because the distance between x1 and x3 is two times of square root of d in one dimensional space. But in the two dimensional space it two times of square root of d, since x3 not in N(x1) at distance d and we need not preserve it. Placing x3 in one dimension space as shown in figure 8 is correct. The tricky part is that where can I place our x4? To place x4, we have multiple options here. Let's say we place x4 at distance d from x1 which means I am preserving the distance of two dimensionalspaces in one dimensional space. I

Figure 8: Four points in one dimensional space

The distance between x3 and x4 is 3d but it must
be d as they are neighborhoods in two dimensional spaces.
N(x3)={x4,x2} but where can we place x4, so that distance willbe preserved. Similarly, if we place x4 as shown in figure9 then the distances will be preserved. Whatever we do, we will make at least one mistake independent of the position of x4 which means at least one distance will get corrupted as shown in Figure9.

Figure9: Four points in one dimensional space with modified locations

In the case of hyper cubes, it is impossible to preserve distances in all the neighborhoods. This problem is called crowding problem. There was a lot of research on stochastic neighborhood embedding before t-SNE. Let's use t-distributed stochastic neighborhood embedding again this t-distribution is one of the distributions in statistics. It's called a student's t-distribution and it has a shape which is slightly different from Gaussian distribution. t-distribution was primarily used to resolve the crowding problem.

V. Applying t-sne

Since we have discussed about what t-SNE actually does. Let's try to run t-SNE on simple datasets to understand how t-SNE behaves. I have three groups of data points with colors blue, orange and green. So, our dataset is basically three groups of points in 2D space.

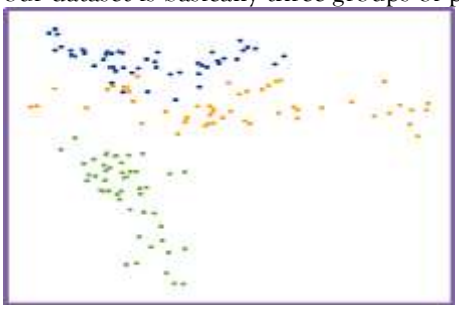

Figure10: Three clusters with equal number of Points before applying t-SNE

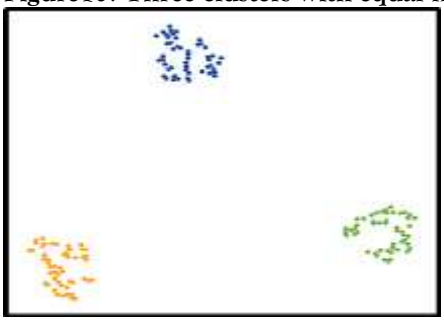

Figure11: Three clusters with equal number of Points after applying t-SNE

From Figure 11 and 12, we can understand that three clusters with equal numbers of points, but at different distances from each other. Here it is two dimensional data with perplexity 10 and epsilon 5. Points per cluster are 50. After 5000 iterations, points of figure 10 are converted to pixels in figure11. For example consider the point in Figure12, in which we have two clusters consisting equal number of points.

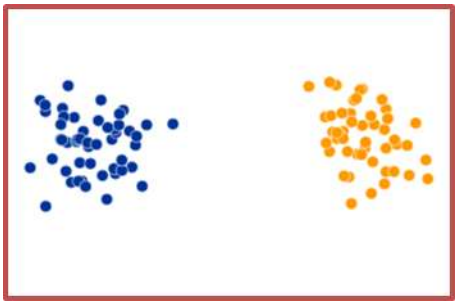

Figure12: Two clusters with equal number of points before applying t-SNE.

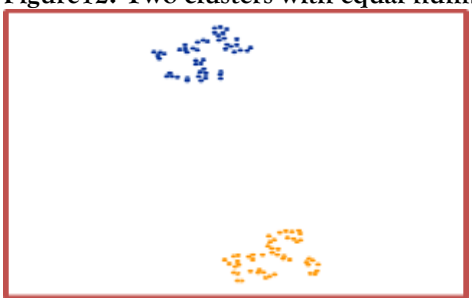

Figure12: Two clusters with equal number of points after applying t-SNE.

Imagine if my original dataset is like two parallel lines of points as shown in Figure13. t-SNE is an iterative algorithm which means t-SNE tries process all the data in iterations and eventually it will converges.

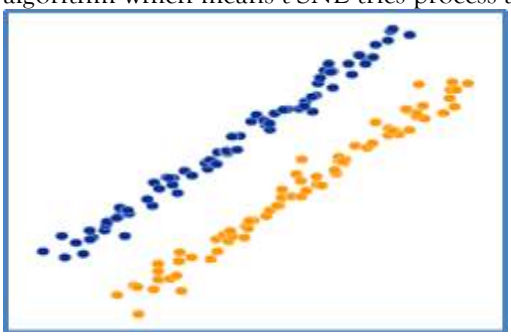

Figure13: Two parallel lines dataset

At every stage, it tries to move the points and also tries to find suitable embedding. In every iteration it will tries to improve the embedding and also it trying to preserve as many neighborhoods as possible.If you take these two parallel lines., there are 50 points per cluster. There are two most important parameters of your t-SNE. First parameter is called perplexity and the second parameter is called step. So step size is basically number of iterations, just remember that with every iterations, we are going to find a better solutions.

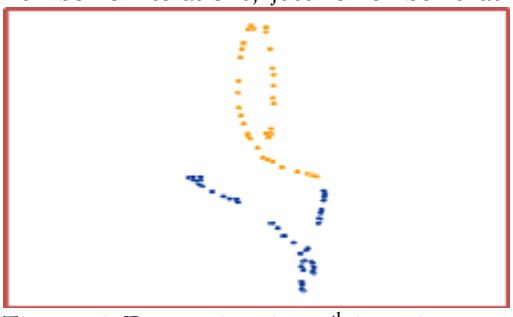

Figure14: Data points in 15th iteration.

Our original dataset is two parallel lines blue points and orange points, at 15th iteration; we are not getting proper results as shown in figure14. As the iteration number is increasing, the shape is stabilizing and shape

does not change much. Two sets of points, arranged in parallel lines that are close to each other. Note curvature of lines.

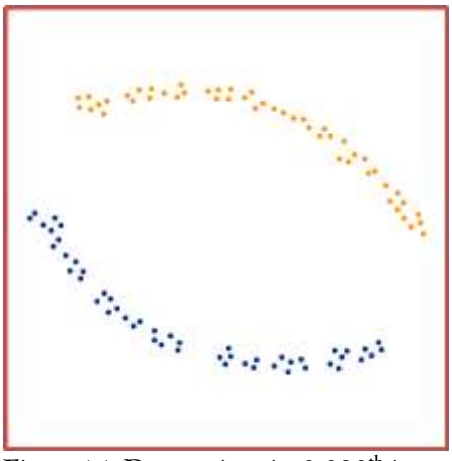

Figure15: Data points in 3,390th iteration.

It is very similar to the underlying structure of the data. So, similarly if you look at step is number of iterations. We should keep iterations till the time shape doesn't change much, after 2000 iterations, it doesn't change much. The second parameter and the most important parameter of t-SNE are called perplexity. Let's understand what perplexity means is, you can think of perplexity roughly or loosely as the number of neighbors. If I say I want to preserve distance of five of my nearest neighborhoods, say if we make our perplexity is equal to 5, what it tries to do is, for every point it will take five points which are closed to it and it will try to preserve the distance. Perplexity can be thought of as a number of points to preserve when I go from high dimension to low dimension. So, first thing is never fix one perplexity value, always run your t-SNE for multiple perplexity values to understand the actual shape. So never run your t-SNE only once which is the most important thing.

Figure16: Dataset for perplexity and steps

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 19s, 2025
https://theaspd.com/index.php

One interesting thing is when we make our perplexity equals to 100 and total numbers of points are 100, we get a mess literally as shown in figure 15. Why is that happening, observe when I am trying to preserve distances for every pair of points. Always try to keep perplexity less than number of data points.

VI. t-SNE on AGAR DATA SET

Let's understand how t-SNE works on AGAR. is a 16,384 dimensional dataset and we are trying to project into two dimensional.

Figure17: Visualizing AGAR with t-SNE.

How would we color them?. We colored them in such a way that all red points indicate zeros. For each data point, we have 16,384 dimensionsand also we have corresponding class label.

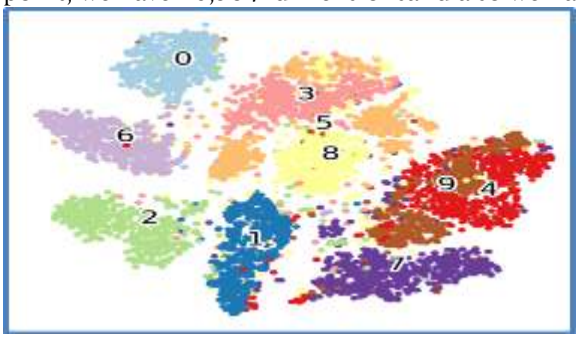

Figure18: Visualizing AGAR Dataset

We notice that all of our zeros are grouped together and all of our sixes are grouped together here. It is not only separating all of our ones from rest of our characters but also it is separating all of your slant ones are together which is important as shown in figure17.This means whatever visually looks similar are getting grouped based on visual similarity which is extremely important. t-SNE grouping the points based on visual similarity which is very interesting. Using t-SNE, we have visualized a 784 dimensional data set by embedding this dataset into two dimensional spacessuccessfully.

VII. CODE EXAMPLE OF t-SNE

```
tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
X_tsne = tsne.fit_transform(X_pca)
# Plotting
plt.figure(figsize=(10, 7))
scatter = plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_encoded, cmap='tab10', s=30)
plt.legend(handles=scatter.legend_elements()[0], labels=le.classes_, title="Classes")
plt.title("t-SNE Visualization of AGAR Dataset")
plt.xlabel("t-SNE Dimension 1")
plt.ylabel("t-SNE Dimension 2")
plt.grid(True)
```

plt.show()

Let us define our model and it can be defined. The variable is model and calling the t-SNE constructor and we are saying that number of components is two and the random state=0. t-SNE is a randomized algorithm which means if you run t-SNE two times, we will get slightly different results. By ensuring that the random state is always set to some number, we are going to get consistent or same results from one run of the program to the second run if you skip this, running t-SNE on the same data set two different times might results in slightly different outcomes because t-SNE is a randomized algorithm. Randomized algorithms will give slightly different results that if you run them two or more times. Now let's look at some default parameters, the default perplexity is set to 30 and learning rate is set to 200. Default maximum number of iterations is 1000. Perplexity is one of the most important parameter along with number of iterations. This basically takes 1000 data points and creates our t-SNE data, which is of course two dimensional because we are saying the number of components is two.Model.fit_transform( ), this basically takes thousand data points and create out t-SNE data, which is of course two dimensional. We are basically combining my t-SNE data with labels data. We are converting this t-SNE data into a data frame with first column named dimensional "Dim1" and second column named "Dim2" and third column being label.

```
# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.show()
```

We have a data frame and we can plot it using Fceit Grid with my hue as labels with my coloring as labels and now let's go and plot it as shown in Figure18.

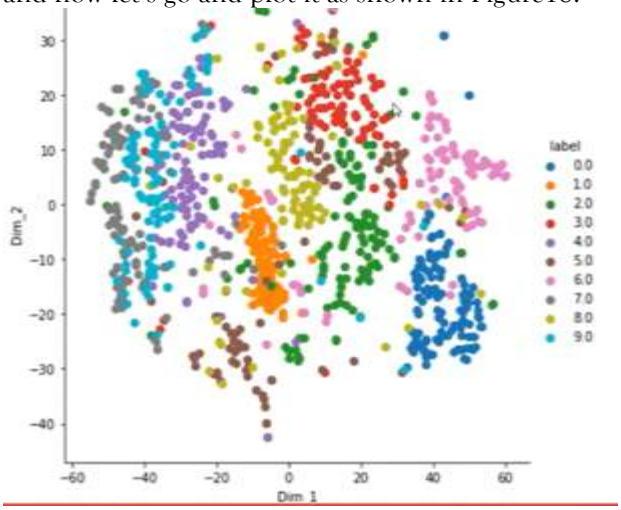

Figure19: AGAR DATASETVISUALIZATION

This is just for 1000 points and perplexity of 30 and just 1000 iterations. This makes very good sense and which all of our zeros are well clustered and all of twos are also well clustered; of course there are small overlaps here and there. If we just had 15000 points or all the 42000 points and if we run 5000 iterations or 10000 iterations and this output could be much better. Now what if we change the perplexity. Soin my code it is just literally one parameter to change. We are keeping the number of components as 2 and my random state as zero and we are changing the perplexity to 50.

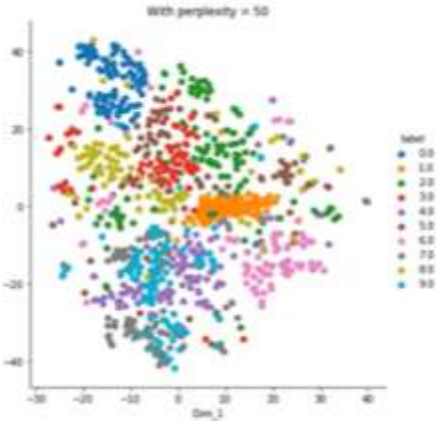

Figure20: AGAR DATASETVISUALIZATION

When we see the figure 18 and 19, it is observed that around 30 or 50 perplexities seem to make sense for 1000 points. Now the next question here is what if I increase the number of iterations. So perplexity 50 seems fairly stable. Around 30 to 50 seems reasonably stable and if I am just setting random state to zero, now instead of 1000 iterations, we want to make the number of iterations equal to 5000and rest of the code is same. When we go from 1000 iterations to 5000 iterations the shape is very stable. We need to try for various values of perplexity and so we tried it 30 and 50 perplexity and 1000 iterations. Perplexity 30, 50 both the shapes looked fairly stable and then increase number of iterations from 1000 to 5000 and the shape still remains fairly stable. We could take 1000 points and shape make pretty good sense, of course when we go from thousand points to 1500 points oreven 2000 points, the points will be separated because of the more information. If you take 42000 data points and you try for various perplexities around 30, 50. Find the good perplexity and run up to 5000 to 10000 iterations. Now just as a sanity check what happens, if we make our perplexity equals to 2. We are not touching the number of iterations which is 1000 by default, random state is zero.We have lost all of the information, so that perplexity=2, may not work very well.Itis advisable to run this code on 42000 points with various values of perplexity and number of iterations to understand. T-SNE is much better if 42000 points take too long on your laptop, you can even do a 15000 points.

VIII. REFERENCES

[1]Sri Siddhartha Reddy, Yen Ling Chao, Lakshmi Praneetha Kotikalapudi, Ebrima Ceesay, "Accident analysis and severity prediction of road accidents in United States using machine learning algorithms", 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp.1-7, 2022.

[2]Ghadah Alshabana, Thao Tran, Marjan Saadati, Michael Thompson George, Ashritha Chitimalla, "Machine learning Models to Predict COVID-19 Cases", 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp.1-8, 2022.

[3]. Nan Jia, Lahari Bagam, Patricia Fabijanczyk, Ebrima Ceesay, "Machine Learning models for Customer Relationship Analysis to Improve Satisfaction Rate in Banking", 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp.1-9, 2022.

[4] Reyhaneh Abdolazimi, Maryam Heidari, Armin Esmaeilzadeh, Hassan Naderi, "MapReduce Preprocess of Big Graphs for Rapid Connected Components Detection", 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp.0112-0118, 2022.

[5]Armin Esmaeilzadeh, Maryam Heidari, Reyhaneh Abdolazimi, Parisa Hajibabaee, Masoud Malekzadeh, "Efficient Large Scale NLP Feature Engineering with Apache Spark", 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp.0274-0280, 2022.

[6] Masoud Malekzadeh, Parisa Hajibabaee, Maryam Heidari, Brett Berlin, "Review of Deep Learning Methods for Automated Sleep Staging", 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp.0080-0086, 2022.

[7] Dr. M Raja Sekar. 2017. Optimization of the MixedModel Processor Scheduling.International Journal ofEngineering Technology and Computer Research(IJETCR). 5(5): 74-79.

[8] Dr. M Raja Sekar. 2017. Fuzzy Approach toProductivity Improvement. International Journal ofComputer & Mathematical Sciences. 6(9): 145-149.

[9] Dr. M Raja Sekar et al. 2016. An Effective Atlas-guided Brain image identification using X-rays.International Journal of Scientific &EngineeringResearch. 7(12): 249-258.
[10] Dr. M Raja Sekar. 2017. Fractional Programmingwith Joint Probability Constraints.InternationalJournal of Innovations & Advancement in ComputerScience. 6(9): 338-342.
[11] Dr. M Raja Sekar. Solving Mathematical Problems byParallel Processors. Current Trends in Technologyand Science. 6(4): 734-738.
[12] Victor Corja, Austin Ryan Crow, Nannan Liu, Ebrima Ceesay, "Bimodal Key Data and Semantic Analysis", 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp.0222-0228, 2022.
[13] Parisa Hajibabaee, Masoud Malekzadeh, Mohsen Ahmadi, Maryam Heidari, Armin Esmaeilzadeh, Reyhaneh Abdolazimi, James H Jr Jones, "Offensive Language Detection on Social Media Based on Text Classification", 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp.0092-0098, 2022.
[14] Johnstone IM, Lu AY. 2009 On consistency and sparsity for principal components analysisin high dimensions. J. Am. Stat. Assoc. 104, 682–693. (doi:10.1198/jasa.2009.0121)
[15] Rao CR. 1958 Some statistical methods for comparison of growth curves. Biometrics 14, 1–17.(doi:10.2307/2527726)
[16] Dr. M Raja Sekar. 2017. Interactive FuzzyMathematical Modeling in Design of Multi-ObjectiveCellular Manufacturing Systems. International Journalof Engineering Technology and Computer Research(IJETCR). 5(5): 74-79.
[17] Dr. M Raja Sekar. 2017. Breast Cancer Detectionusing Fuzzy SVMs. International Journal for Researchin Applied Science & Engineering Technology(IJRASET).5(8): 525-533.
[18] Dr. M Raja Sekaret al. 2017. Areas categorization byoperating support Vector machines. ARPN Journal ofEngineering and Applied Sciences. 12(15): 4639-4647.
[19] Dr. M Raja Sekar et al. 2017. Tongue Image AnalysisFor Hepatitis Detection Using GA-SVM. IndianJournal of computer science and Engineering. 8(4).
[20] Boente G, Silibian-Barrera M. 2015 S-estimators for functional principal components. J. Am.Stat. Assoc. 110, 1100–1111. doi:10.1080/01621459.2014.946991).
[21]. Vichi M, Saporta G. 2009 Clustering and disjoint principal component analysis. Comp. Stat.Data Anal. 53, 3194–3208. (doi:10.1016/j.csda.2008.05.028).