Topological Data Analysis Report

Joshua Sheldon | Dr. Mitra | CSE 5290 | April 29, 2024

Learning Objective

Select analyze 100 equally sampled MNIST digits (10 for each digit) with a TDA software (Mapper in Scikit Learn). For example, some handwritten 9's may look like a 0, they should show up closer together in your TDA visualization.

Field Summary

Topological Data Analysis (TDA) is a branch of data science focused on exploring the underlying shape and features of data to inform analyses such as clustering, predicting trends, and identifying relationships within datasets. At its core, TDA seeks to extract and use the geometrical and topological information encapsulated within data. It employs various algorithms and methods to analyze data from this unique perspective, with the goal of understanding its complex, often high-dimensional structure in a more intuitive, visual way.

One such method is the Mapper algorithm, particularly useful for simplifying and clustering high-dimensional data. The Mapper algorithm involves three primary steps: reducing the dimensionality of the data, forming overlapping clusters (or covers) from this simplified data, and then constructing a graph where nodes represent these clusters and edges reflect overlaps, indicating connections or similarities between data clusters. This approach aids in visualizing and understanding the relationships within the data. Another crucial technique in TDA is persistent homology, which identifies features of the data that persist across multiple scales, thereby distinguishing significant topological features from noise. It does this by evaluating the homology of simplicial complexes—geometric structures connecting nearby data points—and tracking which features (like holes) persist as the scale of observation changes. Persistence helps determine the stability and significance of these features, and methods like calculating the Wasserstein distance can further help compare and quantify the similarity between different data sets or track changes over time.

TDA provides powerful tools like Mapper and persistent homology to dissect and interpret the complex, shape-driven structure of data, enabling more insightful data analysis that goes beyond traditional methods.

Our Goals

We intend to utilize TDA as such:

1. Use the Mapper algorithm to cluster samples of MNIST digits by topological similarity and display those findings in a graph.
2. Use persistence homology and other tools to identify what digit shape features and/or image sections are primarily used to determine topological similarity.

Mapper

Pre-Processing

We begin by looping through the MNIST training set. We collect 10 images for each digit, although the number of images gathered can be configured in the Jupyter Notebook. For each image we collect, we reshape it from the default 28x28 format into a 784-dimension vector. Additionally, we append an identifier to the image's label, identifying which sample it is of the digit it represents.

Dimensionality Reduction

With 784-dimension vectors, we must reduce the dimensionality for any sort of meaningful visualization (often referred to as projection). However, by doing this, we enforce a lens through which we view this data. Depending on the pipeline of operations employed to reduce the dimensionality of the data, the end result may look very different. For example, consider the following figures:

```
plt.show()
```

```
0 = tab:blue
1 = tab:orange
2 = tab:green
3 = tab:red
4 = tab:purple
5 = tab:brown
6 = tab:pink
7 = tab:gray
8 = tab:olive
9 = tab:cyan
```

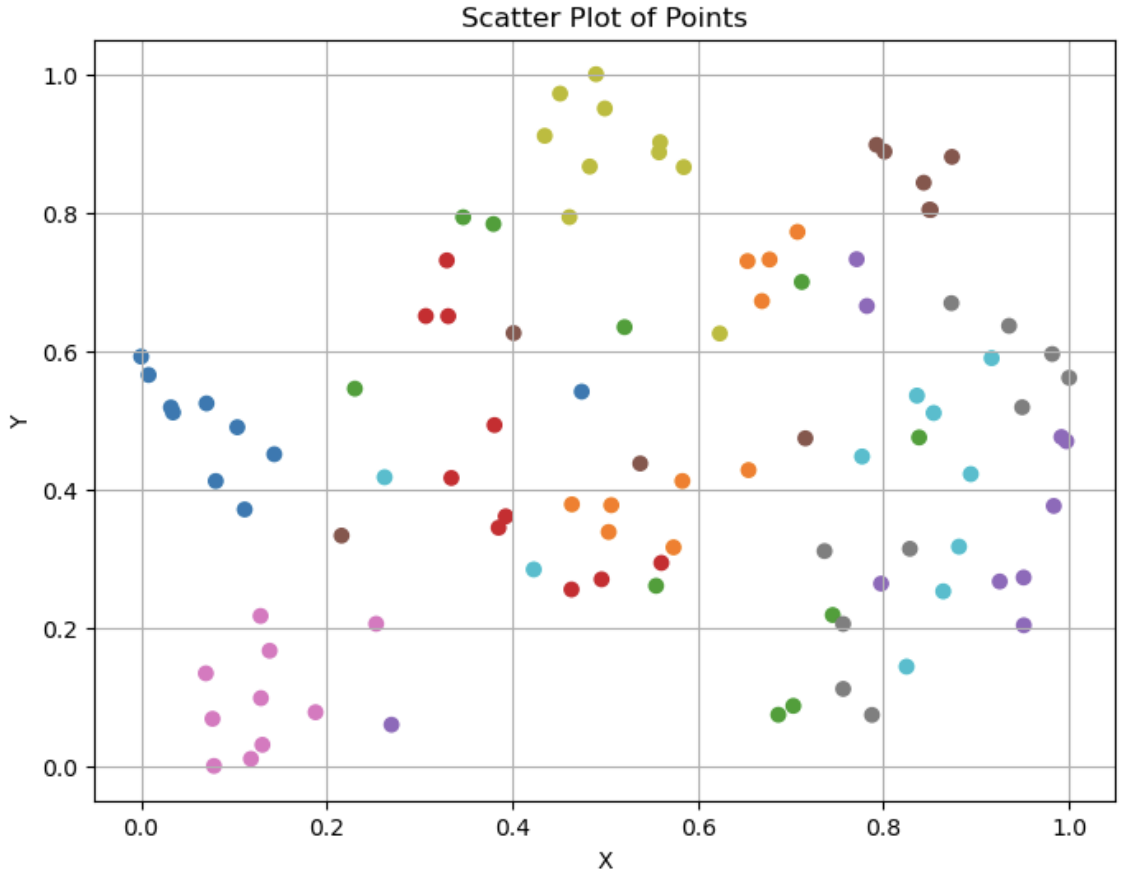

Figure 1 | First Projected Data Scatter Plot

```
0 = tab:blue
1 = tab:orange
2 = tab:green
3 = tab:red
4 = tab:purple
5 = tab:brown
6 = tab:pink
7 = tab:gray
8 = gold
9 = tab:cyan
```

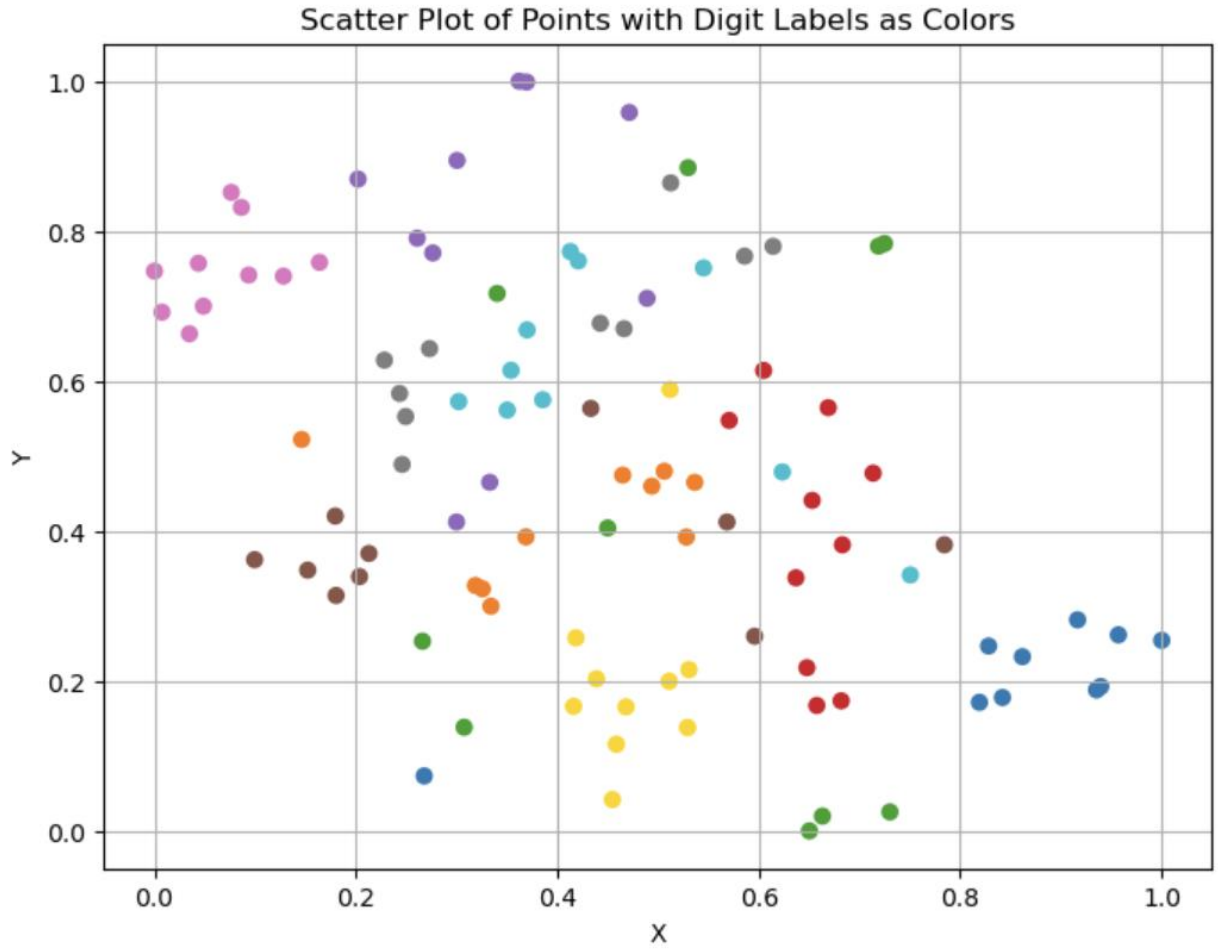

Figure 2 | Second Projected Data Scatter Plot

Both plots were created from the same dataset, and indeed both plots are very similar. If you rotate the first plot 180 degrees, you get an image that looks very similar to the second plot. However, there are subtle yet noticeable differences.

So, we have to be careful with what operations we use to reduce dimensionality. For the purposes of this project, we chose the following pipeline:

1. MinMaxScaler
2. t-SNE (2 components)

The MinMaxScaler scales each feature (each number that composes each image) on a range from (0, 1). The transformation is given by:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

Where X is the dataset, min = 0, and max = 1.

t-SNE stands for T-distributed Stochastic Neighbor Embedding. Though we could explore this tool in exhaustive detail, for the purposes of brevity: t-SNE is a tool that utilizes joint probabilities and the Kullback-Leibler divergences between them to reduce the dimensionality of data.

Optionally, another dimensionality reduction method like TruncatedSVD or PCA can be employed to reduce the dimensionality to a middle value (ex. 50) to improve speed and reduce noise. While we did experiment with this, we got best results by going directly from 784 to 2 dimensions with t-SNE.

Clustering

The next step is to cluster the data. We use two types of clustering: automatic and manual. The intent from the outset is that manual clustering would be intended as a "ground truth" obtained by creating clusters based on data point labels. From there, we figured we could attempt to adjust the methodology and parameters of our automatic clustering algorithms to replicate the results of manual clustering. However, this approach proved problematic for two major reasons.

The first is that with only 100 samples, our data points were very sparse, and there were not many shapes in the average data plot that could be picked up by a clustering algorithm.

The second is that data points of the same digit are not as tightly clustered as we originally expected. A few poorly placed outliers could cause manual clustering to make clusters that were far too inclusive, effectively negating the accuracy of the manual clustering. While we could've culled outliers by filtering by density or something akin to the k-neighbors algorithm to improve accuracy, manual clustering was already a violation of the principles of TDA, and due to the first reason we wouldn't accomplish our original intention for it anyways. Therefore, in interest of time, we decided to drop any further pursuit of it.

For the sake of documenting our learning, we will discuss our approaches to both manual and automatic clustering.

Manual Clustering
For manual clustering, we collect data points based on which digit they represent (i.e. gather all data points that represent "0", all that represent "1", etc.) Then, for each digit, we drew a concave hull around its corresponding data points. This concave hull effectively contained all data points belonging to that digit, while trying to minimize area as much as possible. Then, we created a cluster from every data point within that concave hull (regardless of what digit it represented), and that became the cluster for that digit. An edge would be formed between two clusters if a data point was present in both clusters. From these (clusters) and edges we formed graphs to indicate which digits were related to each other.

Automatic Clustering
For automatic clustering, we used a standard density-based clustering algorithm: DBSCAN. We chose this algorithm because we figured that data points from the same digit would be close together, and that by employing this algorithm we could create clusters that corresponded to each digit. Due to the nature of the data, this did not end up being the case, but we did find parameters that would generally land around 10 clusters.

Another promising option was Spectral Clustering, which works well with a small number of clusters, a medium number of samples, and even cluster size, which fits our target scenario very nicely (10 clusters, 100 samples, 10 samples per cluster). However, we could not get it working successfully.

Findings
We used the KeplerMapper implementation of the Mapper algorithm to visualize the relatedness of different digits, and came to some interesting findings:

- 0 and 6 were typically close on the scatter plot but did not have any overlap/edge between their clusters.
- 9 has a large cluster (implying a similar topology to many digits) that almost never overlaps with 0 or 6.
- 5 and 4 were typically close to 9 on the scatter plot.
- Both 1 and 3 were typically within 2's cluster.
- 9 and 7 were typically close on the scatter plot.
- **All of this could change with different/more data, but these were consistent across lenses/dimensionality reduction pipelines!**

However, even though we now have insights into how related each digit is, we wanted to attempt to get more of a grasp on the important topological features of each digit, and the areas of the digit images that were essential to their differentiation.

Persistence Homology

Our next TDA technique to apply was persistence homology. Simply, this tool works by creating a multidimensional sphere around each feature of the data, and increasing the diameter of the sphere, observing the homologies (shapes) formed by the overlap of the spheres, and their persistence (how long the shapes lasted).

The practical application of persistence homology in this case was generating a persistence diagram for each digit and observing the shapes that were created and their persistence. However, to our surprise, very few loops appeared in the diagrams for the digits, and the number of loops seemed to have little correlation with the actual number of loops in the shape of the digit (ex. 6 had four loops and 8 had one).

The only real utility these diagrams may have ended up serving was to tell the similarity between different digits by calculating the Wasserstein distance between the persistence diagrams of each pair of digits, but that utility was already well covered by our work with the Mapper algorithm.

Non-Negative Matrix Factorization

Introduction

Although non-negative matrix factorization (NMF) is not a TDA tool, we were inspired by Vaishnavi Dixit's presentation on it as it pertains to capturing the essential features of our data. Non-negative matrix factorization is essentially the strategy of factorizing a large dataset (represented by a matrix) into two matrices: W and H, where H is the set of components or features of the dataset, and W is the set of weights, the coefficients for the linear combination of the components that make up each image in the original dataset. The idea is that $W * H$ = the original dataset/matrix. However, one can extract some interesting findings simply by adjusting the number of components and observing the contents of H.

4 Components

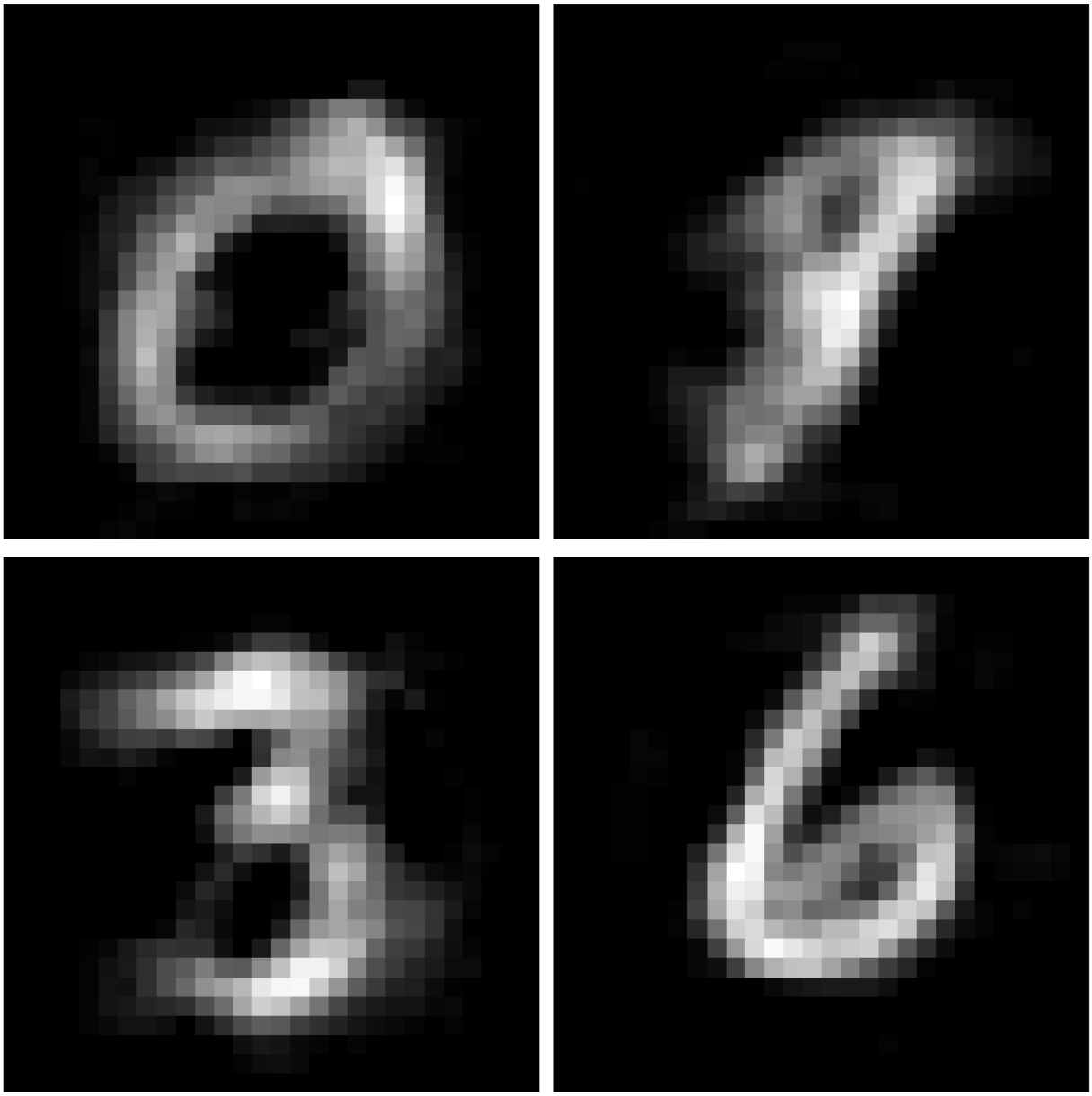

Figure 3 | NMF (4 Components), Visualized H

Interesting observations from these components:

- The four components bear a remarkable similarity to the numbers 0, 3, 6, and 9.
- Overlapping all of the components appears to cover all of the space necessary to represent each image/digit.
- One could imagine the construction of the digit 1 by isolating the overlap between the 9-like component and the 6-like component.

16 Components

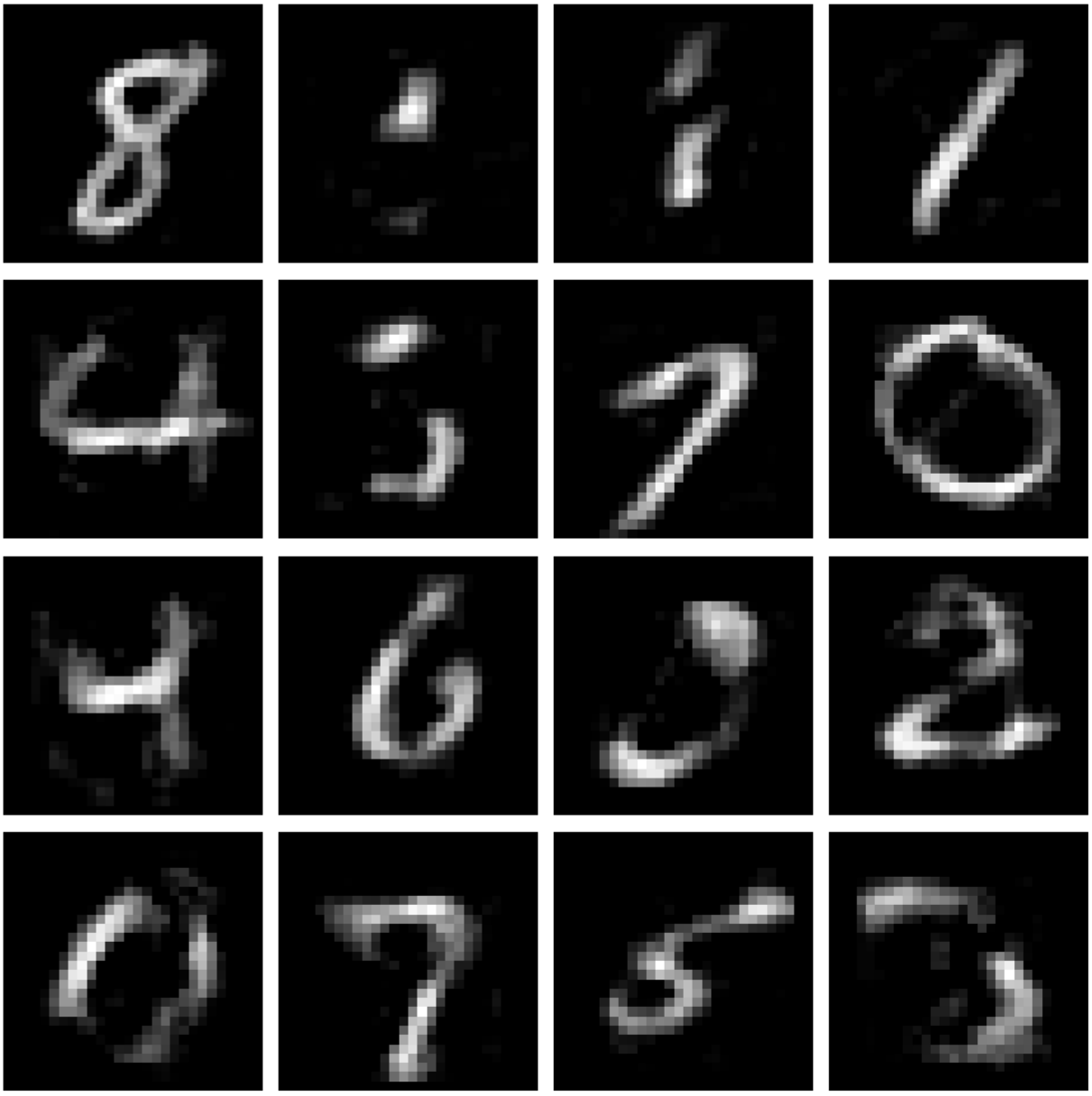

Figure 4 | NMF (16 Components), Visualized H

Interesting observations from these components:

- Many of these components appear to be full-fledged digits: 8, 1, 4, 7, 0, 6, 2, and 5 are clearly represented (in that order, reading left-to-right, top-to-bottom).
- Some of these components appear to represent brush strokes or seemingly random clusters of area. While these may appear random to the human eye, a computer could easily apply weights to all these different images to obtain a precise image.
- One of these components appears similar to the letter "i" (of course, there is no image of an "i" in the dataset, simply interesting).

Higher Number of Components

Although we did test higher component numbers, the factorization becomes less and less insightful as the number of components gets closer to the number of samples in the dataset. Interesting future experiments may include observing the results of NMF with a larger number of images per digit, subsequently causing a larger dataset, while maintaining a small number of components.

Bibliography

[1] F. Chazal and B. Michel, "An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists," *arXiv*, Feb. 25, 2021. https://arxiv.org/abs/1710.04019

[2] A. Dwaraknath, "Topological Data Analysis and Deep Learning," *CS 230*, 2018. https://cs230.stanford.edu/projects_spring_2018/reports/8290918.pdf

[3] E. Escolar, Y. Hiraoka, M. Igami, and Y. Ozcan, "Mapping Firms' Locations in Technological Space: A Topological Analysis of Patent Statistics," *arXiv*, Mar. 31, 2022. https://arxiv.org/abs/1909.00257

[4] A. Garin and G. Tauzin, "A Topological 'Reading' Lesson: Classification of MNIST using TDA," *arXiv*, Oct. 22, 2019. https://arxiv.org/abs/1910.08345

[5] KeplerMapper, "Digits Dataset." https://kepler-mapper.scikit-tda.org/en/latest/generated/gallery/plot_digits.html

[6] B. Krishnamoorthy, "An Introduction to Mapper," *YouTube*, Feb. 11, 2020. https://www.youtube.com/watch?v=NjcLSviRP5E

[7] Y. LeCun, C. Cortes, and C. Burges, "The MNIST Database," 1998. http://yann.lecun.com/exdb/mnist/

[8] scikit-learn, "Clustering." https://scikit-learn.org/stable/modules/clustering.html

[9] scikit-learn, "MinMaxScaler." https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[10] scikit-learn, "Non-Negative Matrix Factorization." https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html

[11] scikit-learn, "t-SNE." https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[12] S. Talebi, "Topological Data Analysis (TDA)," *YouTube*, Jun. 16, 2022. https://www.youtube.com/playlist?list=PLz-ep5RbHosVi8Qoyqvz1MEiYrz35Zb7F

[13] A. Wagenknecht, "Topological Data Analysis of Weight Spaces in Convolutional Neural Networks," *University of Missouri-St. Louis Dissertations*, Apr. 2023, Available: https://irl.umsl.edu/dissertation/1312/

[14] M. Wright, "Introduction to Persistent Homology," *YouTube*, Jul. 03, 2015. https://www.youtube.com/watch?v=h0bnG1Wavag