Investigating Dimensionality Reduction and Clustering Techniques for Pancreatic Cell Data

Spencer Briguglio
School of Computer Science
University of Windsor
Windsor, Canada

Nour ElKott
School of Computer Science
University of Windsor
Windsor, Canada

Zeeman Memon
School of Computer Science
University of Windsor
Windsor, Canada

Karan Vishavjit
School of Computer Science
University of Windsor
Windsor, Canada

***Abstract*—High-dimensional data poses significant challenges to model training and construction due to the curse of dimensionality, resulting in reduced model performance and increased computational overhead. This research paper investigates various dimensionality reduction techniques for deep learning applications to address these challenges. The study focuses on the Muraro dataset, which contains single-cell transcriptome data of the human pancreas. The proposed approach utilizes PCA, kPCA, Autoencoders, UMAP, and t-SNE for dimensionality reduction, followed by k-means clustering to evaluate the performance of each algorithm. Silhouette scores are used to assess cluster quality. The results indicate that nonlinear techniques, particularly UMAP, outperform linear methods in preserving global and local structures and achieving improved clustering results. The optimal number of clusters for kPCA-transformed data is consistently identified as four, while UMAP demonstrates the highest silhouette score among the tested methods.**

I. Introduction

Deep learning (DL) models are increasing in ubiquity and complexity as they are used to address more and more complex problems. At the same time, data collection has seemingly become commonplace and is very easy to perform in many applications. This is good because at the complexity of DL models increases, so too does their need for more data with which to train.[1] It is easy to assume that using more complex and large data is a surefire way to create more powerful models. After all, if we collect more features about training samples, then it only seems to follow that a DL neural network (DNN) will be able to learn more information about samples. Unfortunately, this is often not the case.

Increasing the number of features used in model training directly increases the complexity of the model itself, even without major architectural changes. At the bare minimum, the width of the input layer increases and therefore more connections to the first hidden layer in the DNN are required. This may lead to initial improvements in model accuracy, but this is often short-lived. In supervised tasks, a DNN is essentially a function on a vector, matrix or tensor of features. The purpose of this function is to learn a mapping of input features to a predetermined output features that approximates the true relationship of the input and the output, also called the predictors and the response. Suppose we are tasked with approximating the relationship between the weather today and the weather tomorrow with a DNN. We may collect important observations about temperature and wind but, without prior knowledge, it may not be possible to ascertain which of the two plays a more important role in predicting the weather of tomorrow. Perhaps we find that our model is only accurate about 55% of the time, slightly better than chance, we may then try to find new measurements like cloud cover, dew point or even the weather of neighbouring regions to improve our predictions. Eventually, we may end up collecting hundred or thousands of different measurements at any given time and develop a superior model. It may become apparent that some measurements are only important in specific circumstance (i.e. the amount of snow cover in the summer is almost always 0), and tend to obfuscate the true relationship between today's weather and tomorrow's. Increasing input dimensionality clearly has great initial benefit, but eventually becomes an issue of diminishing returns and eventually a detriment to the model's performance itself. [2] The effects of increasing dimensionality are not limited to the performance of the model in terms of accuracy, precision and other metrics. Increasing input width can lead to prohibitively expensive computation, data processing and data storage costs.

Rapid and large-scale expansion of dimensions gives rise to the "curse of dimensionality".[3] Important data is drowned out by complex or redundant data and interferes with the discovery of fundamental characteristics of the data. Manual feature selection could help to avoid this but may require domain specific knowledge or it may simply be infeasible to manually sift through hundreds or thousands of features. Perhaps it is wholly unknown which features are more important than the others. Fortunately, many methods exist to automatically identify important features and compress the dimensionality of the featurespace of high-dimensional data as this has been an area of intense research for some time. Reducing the dimensionality of data allows for the use of less data, less complex models and for improved model performance in terms of accuracy and training time.

Dimensionality reduction technique use existing features to create a lower dimensional featurespace. This can be accomplished chiefly by feature selection and feature extraction methods. Feature selection is the process selecting a subset of features from the current featurespace to guide model creation. This method provides more interpretability, reduced runtime, and improved generalization to the model. Feature extraction, on the other hand, seeks to isolate effective information from

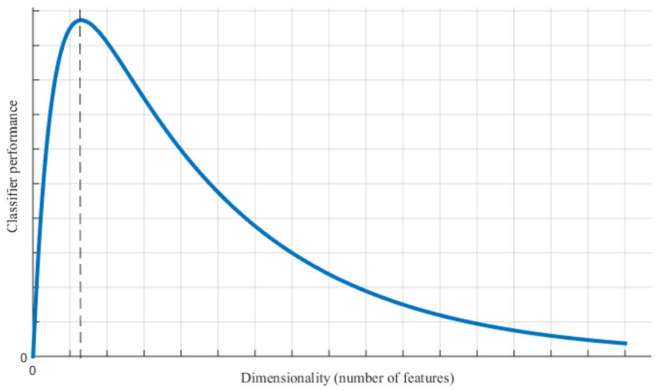

Fig. 1. A plot taken from [CITE] demonstrating the reduction in classifier performance as dimensionality of the input data increases

redundant data by generating new features from linear combinations of those of the original dataset; collecting the important information about them.[4] These techniques are exceedingly import in a variety of high-dimensional data applications such as those in bioinformatics.

Bioinformatics is a subdiscipline that involves the use of computer technology to collect, store, analyze and disseminate biological data and information. Data commonly includes amino acid or genetic sequences, chemical and biochemical measurements and more.[5] This data is leveraged to a wide variety of ends such as in the drug discovery [6], investigating the causes of disease [7], and even elucidating the mechanisms that make our cells and bodies work [8]. One issue commonly faced in this subdiscipline is the very high-dimensionality data associated with biomolecular data such as SNP genotyping and RNA sequencing. There are just over 3 billion basepairs in the human genome encoding approximately 20000 protein coding genes. Many of these genes are implicit in the physiology and function of disease but it not often known a priori which genes are important for which diseases. This makes understand our own molecular biology very difficult as the downstream effects of various genes are extremely complex and nebulous.[8] What's needed is a way to reduce the dimensionality of high-dimensional biological data to something that is permissive of high-performance model construction while avoiding the the curse of dimensionality altogether.

II. Problem Statement

A. *Problem Definition*

Increasingly high-dimensional data is a high-risk to model training and construction. The curse of dimensionality not only results in reduced model performance from an analytical perspective, it also leads to increased training, computation and data storage overhead. By selecting and extracting the most effective information, we can reduce the dimensionality of a dataset to a much more manageable size that mitigates the aforementioned risks. Unfortunately, important features are not always known *a priori* leaving manual feature engineering to require a considerable amount a domain specific knowledge. What's needed is a way to accurately select and extract important features from data so that the effective characteristics of the data are maintained without being lost in the sea of complex data. To this end, we investigate various dimensionlity reduction techniques for use in deep learning applications.

B. *Motivations*

Dealing with lots of features or dimensions is difficult, it can be hard to visualize, and analyze effectively. By reducing the dimensions, we can make the dataset manageable and easier to work with. This simplification of data also helps us understand patterns and relationships in the data more clearly, making visualization better 2 3. Reducing dimensions also saves time and computational resources because it makes analysis tasks less complex and there are less number of attributes or features to deal with. This means we can process the data more quickly and efficiently.

Dimentionality reduction also prevents overfitting. Having too many features, models can become over complicated and may not work well with test data. Dimensionality reduction solves this problem by simplifying the model and reducing the number of features . This makes the model better at understanding the overall trends and making accurate predictions.

Reduction techniques help in filtering out noise or irrelevant features 4. By focusing on the most important features, we can improve the quality of the analysis. This enhances the ability to interpret the data and understand the underlying factors that affect it.

Dimensionality reduction also involves feature selection and compression. Compression removes redundant features while retaining most prominent ones. Compression refers to representing the data in a more compact way, which saves storage space and makes computations faster. Feature selection means choosing the most informative features that are influencing the data and leaving out the less useful ones. Feature selection is very crucial in biology and finance.

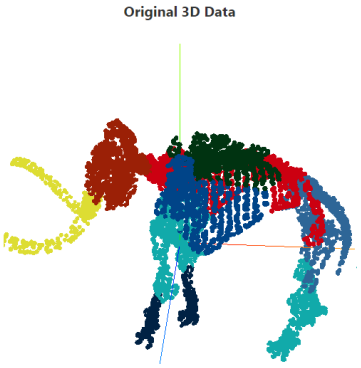

Fig. 2. Original High Dimension 3D Data [24]

C. *Justifications*

The application of dimensionality reduction techniques, such as PCA, kPCA, Autoencoders, UMAP, and t-SNE, in

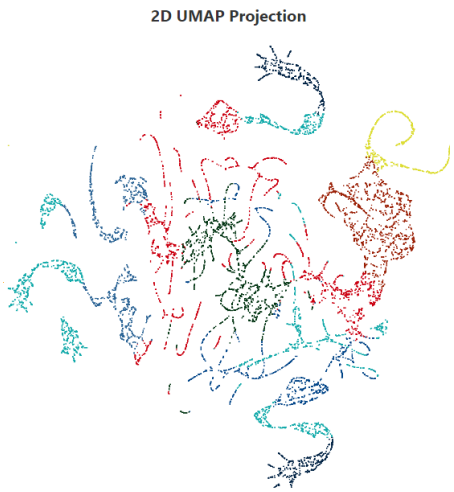

Fig. 3. 2D UMAP Projection [24]

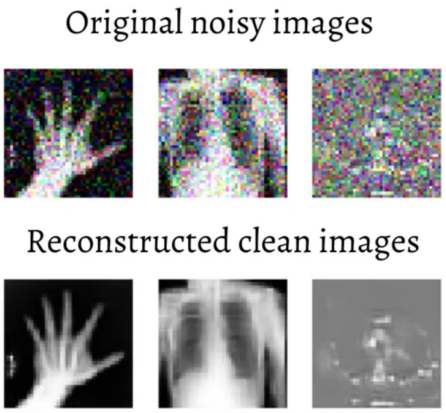

Fig. 4. Filtering Noise

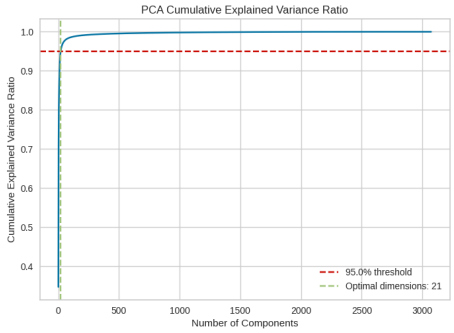

Fig. 5. Optimal Dimensions Computation

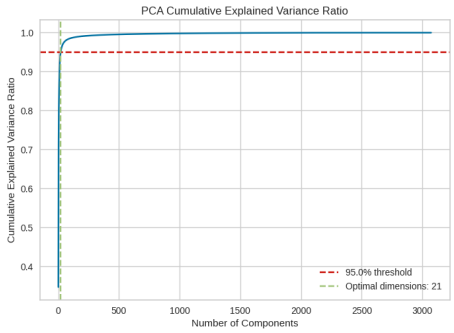

Fig. 6. Optimal Dimensions Computation

single-cell sequencing is crucial for overcoming the challenges presented by high-dimensional and sparse scRNA-seq datasets [9]. With the emergence of single-cell sequencing as a dynamic technology, researchers are now able to capture intricate cell-level information at the single-nucleotide level, enabling the analysis of individual cells independently and deepening our understanding of cellular processes. However, the resulting scRNA-seq datasets are notoriously high-dimensional and sparse, which poses significant obstacles for analysis and interpretation [10].

The use of dimensionality reduction methods becomes essential in scRNA-seq data processing as they transform the high-dimensional and noisy expression matrix into a lower-dimensional subspace [11]. By reducing the dimensionality of the data, these techniques facilitate effective noise removal, data visualization, and downstream analysis. Moreover, dimensionality reduction enables researchers to study the transcriptome of individual cells in a more manageable form, enhancing their ability to decipher the underlying biological processes [12].

The continuous refinement and development of dimensionality reduction techniques hold great promise for advancing our understanding of cellular heterogeneity and its role in biological processes. By providing concise and interpretable representations of the underlying biological processes, these techniques contribute to the effective management and interpretation of large and complex scRNA-seq datasets.

III. Related Works

There are several research papers on dimentionality reduction, below the original research and papers extending/applying the original work.

PCA [13] was first proposed by Karl Pearson in 1901. The paper introduced the concept of PCA, which is a statistical technique used for dimensionality reduction and data visualization. A 2021 paper “Machine Learning: Algorithms, Real-World Applications and Research Directions” highlights Applicability of different machine learning techniques on various real-world applications such as healthcare, e-commerce, cyber security and many more. Dimensionality Reduction and Feature Learning.

A 2020 research paper [14] ”Principal component analysis: a review and recent developments” outlines Preserving most of the information of data while drastically reducing their dimensions.

Original research on kPCA [15]“Nonlinear Component Analysis as a Kernel Eigenvalue Problem” Extending traditional linear approach to handle non linear data patterns. Kernel functions are introduced to map input data into high dimensional space where PCA can be applied.

A 2016 research [16]"A Unifying Review of Deep and Shallow Anomaly Detection" explains Reconstruction Models that are optimised to reconstruct normal data, thereby aim to detect anomalies by failing to reconstruct under learned model. Most common methods PCA for AD considered as baseline for default reconstruction. Auto Encoders: that use Neural Networks for encoding decoding.

Original research on Auto Encoders [17] "Learning Internal Representations by Error Propagation" Research describes as NN architecture capable of learning representations of input data by reconstructing input at output layer term "AutoEncoder" was not used. A 2021 research [18]"Autoencoders" Talks about various types of autoencoders and describe various application and use-case of autoencoders.

In 2020 research paper [16]"A Unifying Review of Deep and Shallow Anomaly Detection" explains Reconstruction models on the Big Moon, Small Moon PCA with linear subspace with the lowest reconstruction error kPCA enables an optimal reconstruction from (kernel-induced) nonlinear components in input space AE with 1-D latent code learns a 1-D, nonlinear manifold in input space having minimal reconstruction error

Original Research on UMAP [19] "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction" constructs a low-dimensional representation of the data by optimizing a cost function that preserve of both local and global structure. Can handle large datasets efficiently and is effective in preserving complex relation between data points. Understand high-dimensional data by providing low-dimensional representation while retaining important information.

A 2020 research [20] "A review of UMAP in population genetics" Applications of UMAP in population genetics Finds the nearest genetic neighbours for each individual and creates low-dimensional representations that group more closely-related individuals together, preserves longer-range relatedness through intermediary individuals.

IV. Methodology

A. Material and Data

The dataset used is the one compiled by Muraro et. al. (2016) for their paper titled "A single-cell transcriptome atlas of the human pancreas," which provides different cell types in the human pancreas at single-cell resolution, allowing for comparison of gene expression patterns among cell types and the identification of subpopulations within them. The dataset comprises over 5000 unlabeled data containing information about cell types and gene expression in the collected samples from the human pancreas. The purpose of using this dataset is to assess the effectiveness of the mentioned clustering and dimensionality reduction techniques in identifying significant cell types and patterns of gene expression. It is to be noted that in the original dataset, the columns represent samples rather than features of the cell samples. For the purposes of this project, the dataset has been modified so that the features are now the first element in each row.

B. Proposed Methods

This paper proposes a comprehensive approach for classifying the data from the Muraro dataset by employing the aforementioned dimensionality techniques. The methods considered for analysis include PCA, kPCA, Autoencoders, UMAP, and tt-SNE. To assess the performance of each algorithm, k-means clustering will be applied to the reduced-dimensional data obtained from each method. The resulting clusters will be evaluated using silhouette scores, a widely used metric for cluster quality assessment. The silhouette scores provide insights into the cohesion and separation of the clusters, allowing for the comparison and ranking of the different algorithms' performance in clustering the Muraro dataset.

C. Formal Complexity

Each dimensionality reduction technique has its own computational complexity, which refers to the amount of computational resources required to execute the algorithm. In simple terms, it indicates how much time and memory are needed to perform the calculations.

For PCA Computational Complexityis $O(n\hat{3})$, where n is the number of features or attributes in the dataset. PCA involves performing eigenvalue decomposition or singular value decomposition, which has cubic complexity.

kPCA (Kernel Principal Component Analysis) has a Complexity of $O(n\hat{3})$, where n is the number of samples in the dataset. Similar to PCA, kPCA requires eigenvalue decomposition or singular value decomposition, resulting in cubic complexity.

Autoencoders complexity Varies based on the architecture and complexity of the neural network. Training autoencoders typically involves multiple forward and backward passes through the neural network layers, and the complexity depends on the sizes of network size and number of training iterations. UMAP has logrithmic complexity of O(n log n), where n is the number of samples in the dataset. UMAP utilizes nearest neighbor searches and constructing a graph, which has a logarithmic complexity.

For t-SNE Complexity is $O(n\hat{2})$, where n is the number of samples in the dataset. t-SNE iteratively computes pairwise similarities between samples, which results in quadratic complexity.

After applying each dimensionality reduction technique, k-means clustering is performed on the reduced-dimensional data. The complexity of k-means clustering is typically O(I * n * k * d), where I is the number of iterations, n is the number of samples, k is the number of clusters, and d is the dimensionality of the data.

The silhouette score calculation, used to evaluate the quality of the resulting clusters, has a complexity of O(n), where n is the number of samples.

The proposed methodology complexity would mainly depend on the dimensionality reduction techniques used, the size of the dataset, and the number of clusters. The most computationally expensive steps are often the reduction tech-

niques involving eigenvalue decomposition or singular value decomposition, which have cubic complexity.

V. Computational Experiments

A. *Results and Discussions*

*1) **PCA and kPCA**:* The results obtained from applying PCA and K-PCA revealed interesting findings regarding the performance of PCA and kPCA on the Murano dataset. Before applying the dimensionality reduction, the optimal number of dimensionality was computed, which we found to be 21 as illustrated in Fig7.

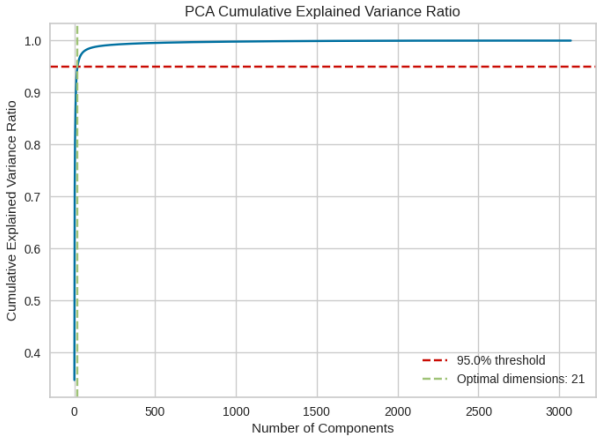

Fig. 7. Optimal Dimensions

PCA, despite its simplicity and ability to capture the most significant variance in the data, resulted in relatively lower silhouette scores compared to kPCA.

The discrepancy between the silhouette score and the elbow method results for the optimal number of clusters in the PCA-transformed space suggests that the underlying structure of the data might not align well with the assumptions made by K-Means clustering.

On the other hand, kPCA with the RBF kernel, particularly with a gamma value of 1.000, provided more consistent and higher silhouette scores. This implies that kPCA was able to capture the non-linear relationships in the data more effectively, resulting in better-defined clusters.

Moreover, the optimal number of clusters consistently identified by both the silhouette score and the elbow method for the kPCA-transformed data was four. This suggests that the kPCA approach with four clusters and a gamma value of 1.000 is the most suitable for analyzing the Murano dataset.

Diving deeper into PCA; we computed a range of cluster numbers (from 2 to 7) when applying KMeans clustering to the PCA-transformed dataset, from which we used to compute the average silhouette score. The silhouette score is a measure of cluster quality, with higher values indicating better-defined and well-separated clusters.

8

We found the the Silhouette Score Average to be 0.4963913357527921. Furthermore, the scatter plot in Fig9.

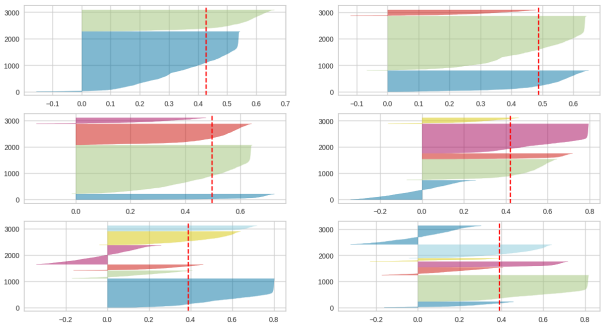

Fig. 8. Silhouette Average Score - PCA

provides insights into the distribution and separation of the clusters formed by applying the K-Means algorithm to the PCA-transformed data, followed by computation of the optimal number of clusters specifically for the PCA-transformed data as demonstrated in Fig10.

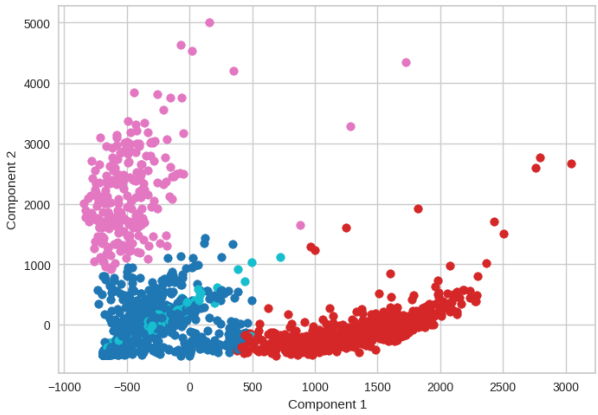

Fig. 9. Scatter Plot of PCA Transformed Data

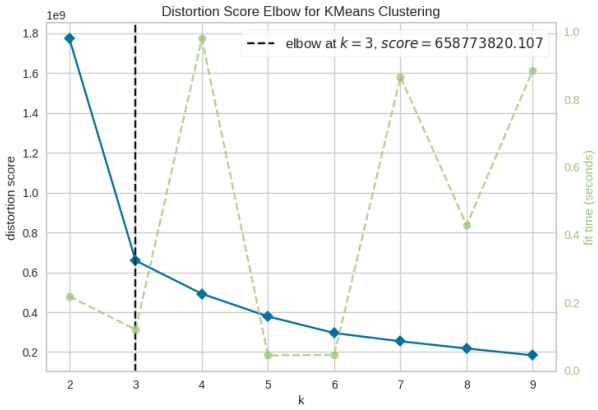

Fig. 10. Optimal Number of Clusters

To evaluate K-PCA results, we implemented K-Means clustering on the kPCA-transformed data for different numbers of clusters (2, 3, 4, 5, 6, 7). The silhouette score is calculated for each clustering result and displayed for evaluation in Fig11.

To analyze the optimal number of clusters for the kPCA transformed data using Elbow method, we concluded with 4 that yielded a score of 0.138.

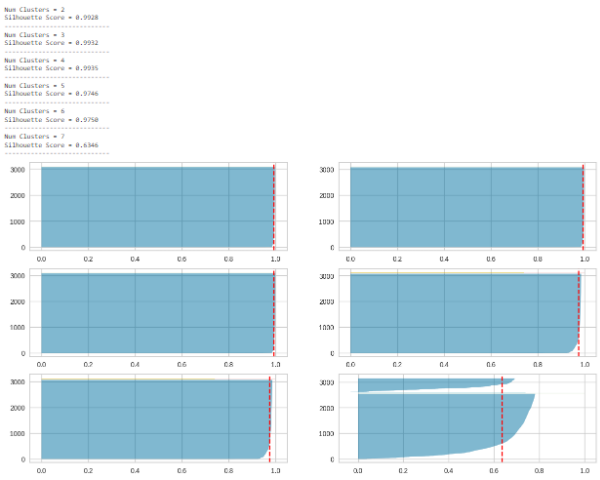

Fig. 11. K-Means Clustering Silhouette Scores

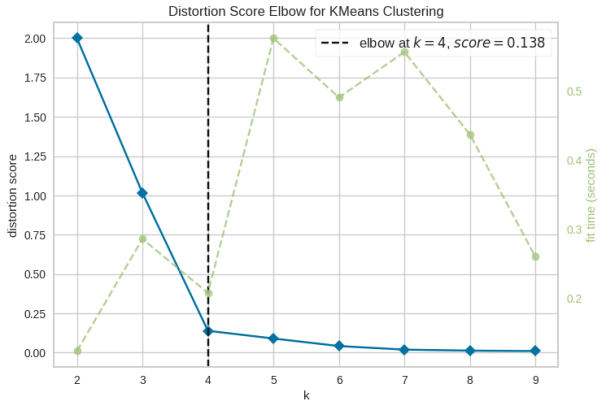

Fig. 12. Elbow Method for Optimal Cluster Number

In conclusion, after evaluating both PCA and KPCA along with their variations and tuning different parameters, the analysis indicates that kPCA with an RBF kernel, a gamma value of 1, and four clusters yields superior clustering results compared to PCA for the Murano dataset. This finding highlights the importance of considering non-linear relationships when analyzing complex datasets and emphasizes the potential of kPCA, particularly when appropriately tuned, for such tasks.

*2) **Autoencoders**:* PCA identified the optimal dimensionality of 21. It is a common assertion that autoencoders (AE) can find the same representation as PCA. To assess this an autoencoder with 5 hidden dense layers was constructed. The input and output layers of the AE each are the same dimension of the pre-processed dataset with a width of 19140. The first hidden layer h_1 had width 8192, the second layer h_2 had width 4096. The code bottleneck layer, the third hidden layer h_3, had a width that was a hyperparameter which we tune. The follow layers, h_4 and h_5 both had widths 4092 and 8192 respectively. It is important to note that in the AE architecture the initial two layers leading to the "bottleneck" are know as the encoder while the layers responsible for reconstructing the original input from this encoding is the decoder. The activation functions for all layers was a leaky ReLU except in the final hidden layer which utilized a basic ReLU activation function. The reason for this is because of the data being trained on is very sparse and leads to exploding and/or disappearing gradients. To combat this, the ReLU is used so that the majority of output are 0 as would be expected for a sparse output. Leaky ReLU is a bad choice here because it results in negative outputs which are not compatible with the data that has been normalized to a range between 0 and 1. To further increase the model's robustness to the sparse data, we also randomly initialized all weight and variables of the networks at runtime. All training was done using a 80-20 train-test split and 5-fold cross-validation. Early stopping and checkpointing were also implemented.

The AE here is used as a means of feature extraction. The goal is to train the AE to reconstruct the input layer. As a result loss is calculated using the input data as the desired output of the network. Models were trained with various bottleneck sizes ranging from 2 to 4096, increasing in code size by an exponent of 2 (i.e. 2, 4, 8, ...). In general the MSE on the testset was similar across the models at approximately 0.0034-0.0035. Following model training, the encoder was used to generate an encoding of the test data. 4 different models' encoders were selected to be used for test set feature extraction with output sizes 16, 21, 512, and 4096. The test data was passed separately through each of these encoders to generate the compressed data. A correlation matrix heatmap of the uncompressed data (not shown) demonstrates the redundancy of the original data. To illustrate the effectiveness of the compression, the correlation matrix heatmap for the 21 width encoded data is also shown. Clearly, the AE was able to extract effective data from this complex and largely redundant and sparse dataset.

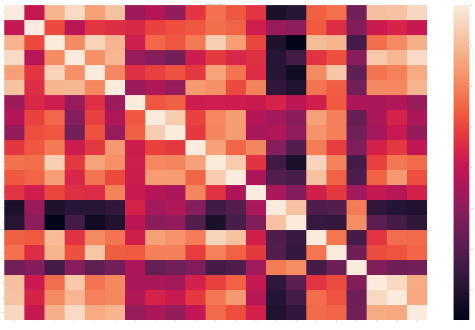

Fig. 13. The correlation matrix heatmap illustrating the features extraction of the AE. The bright the cells, the more highly correleated the row features are with the respective column features.

Following this data compression, the resulting output vectors were clustered so that they may be compared with other methods. The silhouette score was used to evaluate the quality of the k-means clustering of this encoded data for each of the encoding sizes. Additionally, the Elbow Method for Optimal Cluster Number identification was later used. In the smallest encoding which contains 16 features, we can see that the silhouette score is the highest when k=2 for a score of 0.5771. This result is echoed by the other AE observations which also exhibit the highest silhouette score when the number of clusters is 2 for a score of 0.5928, 0.5403, and 0.5660 for the size 21, 512 and 4096 codes respectively. A visualization of the silhouette scores is also shown where for each respective code size and k value tested, we plot the samples present in each cluster against their respective silhouette score. Across all code sizes, we can see that when k=2, the silhouette scores are maximized indicating that clusters are well-clustered into distinct and well-defined clusters. It is important to note that many sample exhibited a negative silhouette score compared to other methods. This means that this compression method resulted in many outlier samples which were not well capture by the various clusters.

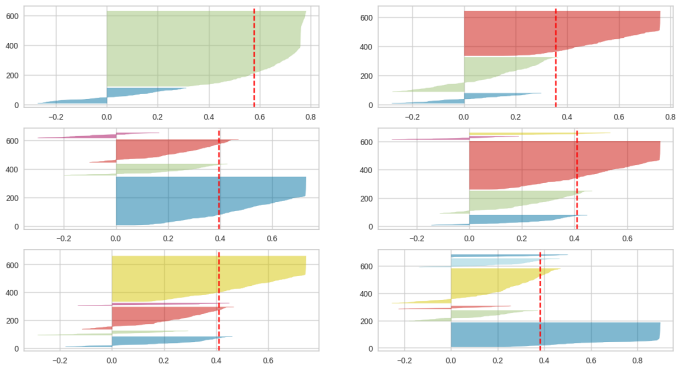

Fig. 14. A visualization of the silhouette scores showing the number of clusters against the silhouette scores for the 16 wide code vectors.

```
Num Clusters = 2
Silhouette Score = 0.5771
----------------------------
Num Clusters = 3
Silhouette Score = 0.3557
----------------------------
Num Clusters = 4
Silhouette Score = 0.3972
----------------------------
Num Clusters = 5
Silhouette Score = 0.4092
----------------------------
Num Clusters = 6
Silhouette Score = 0.4118
----------------------------
Num Clusters = 7
Silhouette Score = 0.3829
----------------------------
```

Fig. 15. The silhouette scores of K-Means clustering of the 16 wide code vectors using various number of clusters.

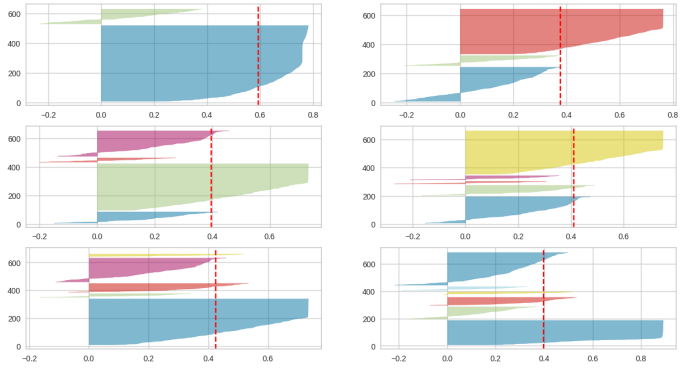

Fig. 16. A visualization of the silhouette scores showing the number of clusters against the silhouette scores for the 21 wide code vectors.

```
Num Clusters = 2
Silhouette Score = 0.5928
----------------------------
Num Clusters = 3
Silhouette Score = 0.3767
----------------------------
Num Clusters = 4
Silhouette Score = 0.3975
----------------------------
Num Clusters = 5
Silhouette Score = 0.4099
----------------------------
Num Clusters = 6
Silhouette Score = 0.4242
----------------------------
Num Clusters = 7
Silhouette Score = 0.3980
----------------------------
```

Fig. 17. The silhouette scores of K-Means clustering of the 21 wide code vectors using various number of clusters.

To further investigate the the optimal k-value for k-means clustering, elbow method was used. Here, the objective was to minimize the distortion score by iteratively adjusting the centroids of the various clusters. Here, the results conflict. with the optimal k and suggest a k-value of 4 or 5 is optimal. When compared to other methods this is, however, in agreement with results obtain using other feature extraction methods.

*3) **T-SNE and UMAP**:* In the initial analysis, the t-SNE algorithm was applied to the dataset to reduce its dimensionality and visualize the data in a lower-dimensional space. Before performing k-means clustering to identify distinct clusters, the average silhouette score was calculated, yielding a value of 0.4417124. The silhouette score measures the quality of clustering results, where higher values indicate better-defined clusters and better separation between clusters. In a similar vein, the UMAP algorithm was employed, and the average silhouette score was computed, resulting in a higher score of 0.62660885. This score indicates improved clustering quality and suggests better-defined and well-separated clusters in the UMAP-transformed space. 26 28 Figures 26 and 28 exhibit the resultant clustering outcomes following the application of t-SNE (Figure 26) and UMAP (Figure 28).

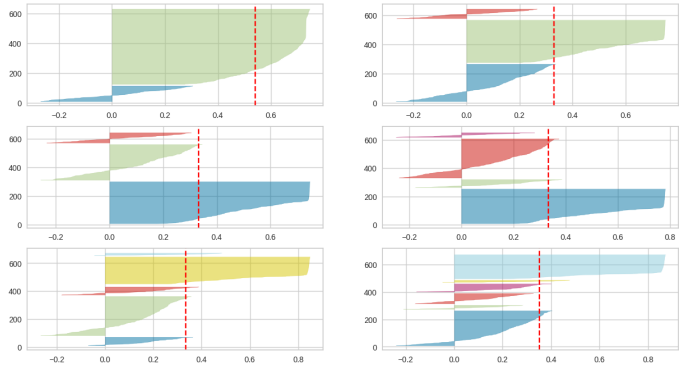

Fig. 18. A visualization of the silhouette scores showing the number of clusters against the silhouette scores for the 512 wide code vectors.

```
Num Clusters = 2
Silhouette Score = 0.5403
----------------------------
Num Clusters = 3
Silhouette Score = 0.3290
----------------------------
Num Clusters = 4
Silhouette Score = 0.3330
----------------------------
Num Clusters = 5
Silhouette Score = 0.3333
----------------------------
Num Clusters = 6
Silhouette Score = 0.3352
----------------------------
Num Clusters = 7
Silhouette Score = 0.3515
----------------------------
```

Fig. 19. The silhouette scores of K-Means clustering of the 512 wide code vectors using various number of clusters.

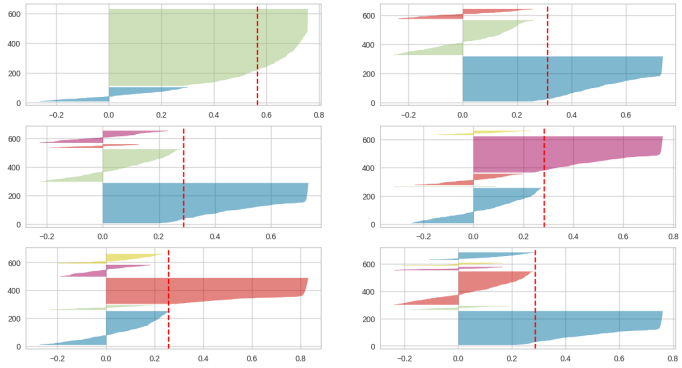

Fig. 20. A visualization of the silhouette scores showing the number of clusters against the silhouette scores for the 4096 wide code vectors.

```
Num Clusters = 2
Silhouette Score = 0.5660
----------------------------
Num Clusters = 3
Silhouette Score = 0.3126
----------------------------
Num Clusters = 4
Silhouette Score = 0.2898
----------------------------
Num Clusters = 5
Silhouette Score = 0.2830
----------------------------
Num Clusters = 6
Silhouette Score = 0.2572
----------------------------
Num Clusters = 7
Silhouette Score = 0.2862
----------------------------
```

Fig. 21. The silhouette scores of K-Means clustering of the 4096 wide code vectors using various number of clusters.

The disparity in silhouette scores between t-SNE and UMAP highlights the strength of UMAP in capturing the underlying structure and patterns of the dataset. 29 27 Figures 27 and 29 show the plots of the number of clusters (x-axis) against the corresponding silhouette scores (y-axis), providing a visual representation of the relationship between the number of clusters and the quality of clustering for t-SNE and UMAP, respectively. The higher silhouette score for UMAP implies that the resulting clusters are more distinct and well-separated compared to t-SNE. This demonstrates the superior ability of UMAP in preserving global and local structures, effectively representing complex relationships, and enabling more accurate and reliable clustering results. The higher silhouette score obtained with UMAP reinforces its effectiveness and suitability for tasks that require robust cluster identification and analysis in high-dimensional datasets.

The choice and application of the k-means clustering algorithm can have an impact on the accuracy of the final clustering results obtained with both t-SNE and UMAP. K-means is a popular clustering algorithm that assigns data points to clusters based on their proximity to cluster centroids. In the context of t-SNE and UMAP, the quality of the clustering results can be influenced by how well the clusters in the reduced-dimensional space align with the original high-dimensional data. If the clusters identified by k-means do not accurately reflect the inherent structure of the data, it can impact the accuracy of the final clustering.

In future research, addressing these challenges will require meticulous selection of the cluster count (k) and conducting multiple iterations of k-means with diverse initializations. The evaluation of clustering outcomes using metrics such as the silhouette score or domain-specific validation measures will be instrumental in gauging the precision and consistency of clustering across various runs. Additionally, exploring alternative clustering algorithms such as Gaussian Mixture Model (GMM), spectral clustering, and mean shift, or employing ensemble methods may enhance the dependability and accuracy of the final clustering results.

VI. Conclusion

A. *Summary*

In this study, we explored various dimensionality reduction techniques and clustering algorithms to analyze the Muraro dataset, which contains high-dimensional single-cell RNA sequencing data. The primary objective was to identify meaningful clusters of cells, representing distinct cell types, while

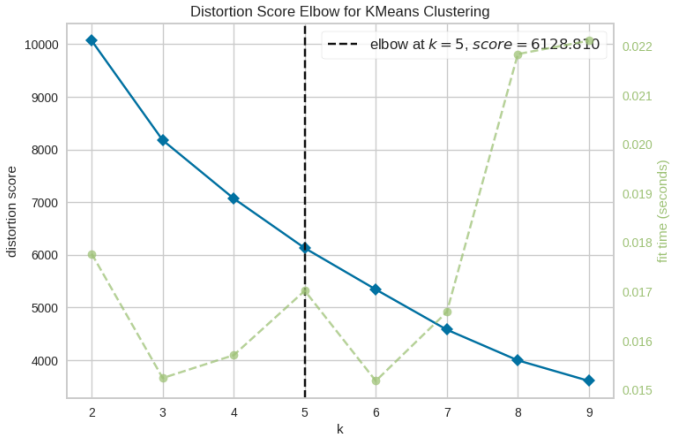

Fig. 22. An illustration of the elbow method used to determine the optimal k-value for the 16 dimensional code.

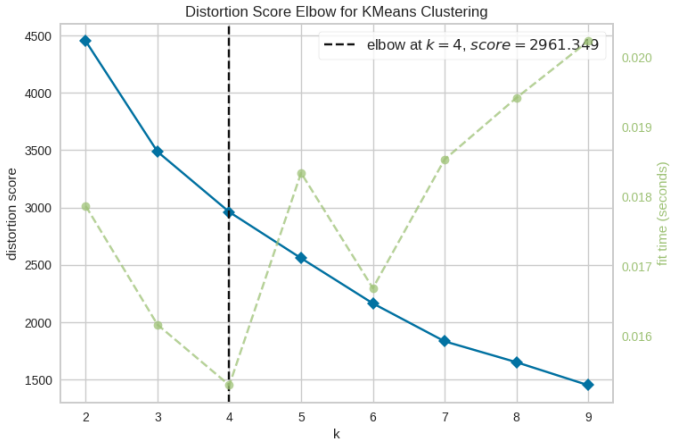

Fig. 23. An illustration of the elbow method used to determine the optimal k-value for the 21 dimensional code.

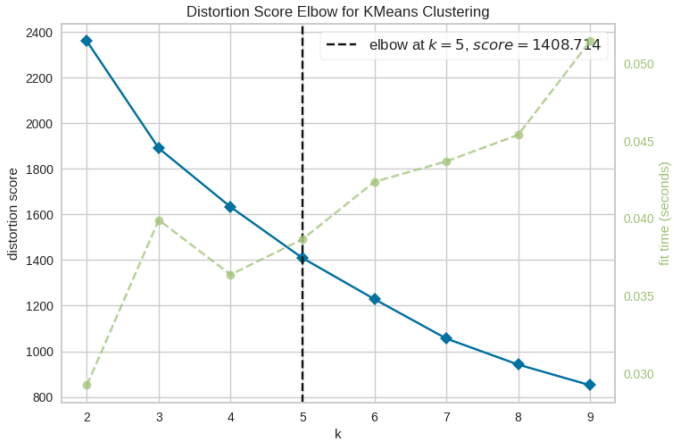

Fig. 24. An illustration of the elbow method used to determine the optimal k-value for the 512 dimensional code.

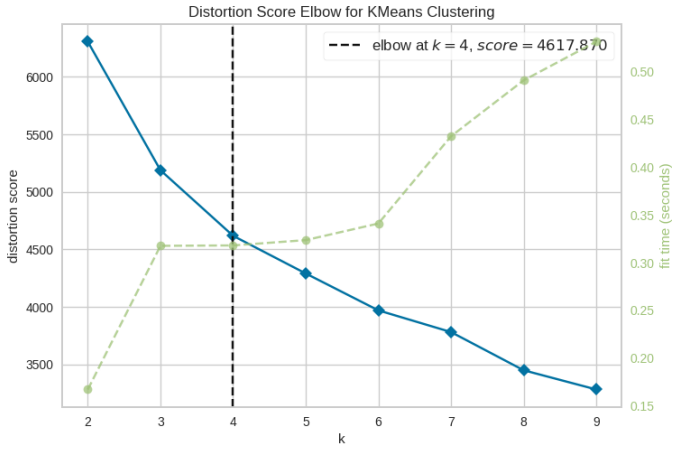

Fig. 25. An illustration of the elbow method used to determine the optimal k-value for the 4096 dimensional code.

preserving the inherent structure of the data. We investigated the performance of PCA, kPCA, autoencoders, t-SNE, and UMAP for dimensionality reduction, followed by k-means clustering to group the cells.

Our findings demonstrated that the non-linear techniques, particularly UMAP, outperformed the linear methods in terms of preserving global and local structures and yielding improved clustering results. The optimal number of clusters for kPCA-transformed data was consistently identified as four, and UMAP achieved the highest silhouette score among the methods tested.

Despite the challenges associated with generalization, interpretability, and preserving structure in reduced-dimensional representations, our study provides valuable insights into the application of different dimensionality reduction techniques and clustering algorithms for single-cell RNA sequencing data analysis. The results pave the way for the development of more advanced and efficient methodologies to decipher complex biological data and facilitate a deeper understanding of cell types and their functions in various biological contexts.

B. Future Research

Hybrid Approach: Instead of implementing single dimensionality reduction technique, future research can explore more adavanced and hybrid approaches that combine multiple techniques such as integrating PCA with Autoencoders or combining t-SNE with UMAP may potentially overcome limitaion of one technique and provide better representation and clustering results.

Deep Learning Architectures (GAN): Future research can explore more advanced deep learning architectures, such as Variational Autoencoders (VAEs) or Generative Adversarial Networks (GANs). These architectures are very efficient in capturing complex data and can potentially improve clustering performance.

Hyperparameter tuning: Instead of transforming data into two-dimensional space, researchers can explore data in high dimension and find the most suitable number of dimensions for the reduction and further performing optimization on optimal values.

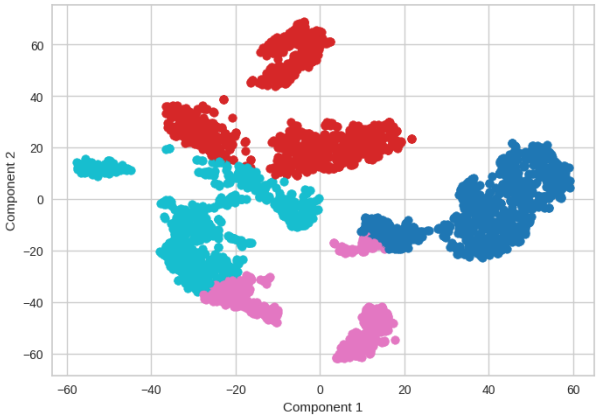

Fig. 26. TSNE clustering

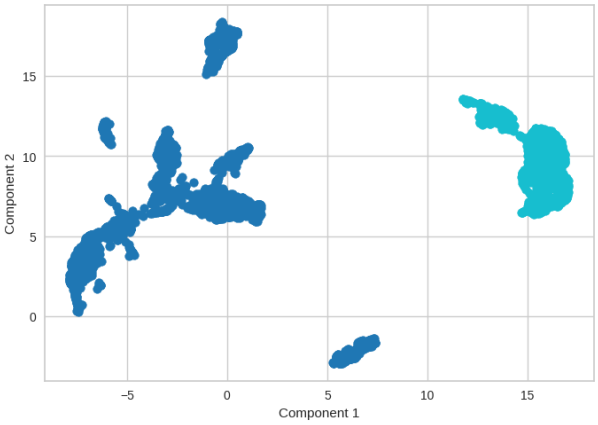

Fig. 28. UMAP clustering

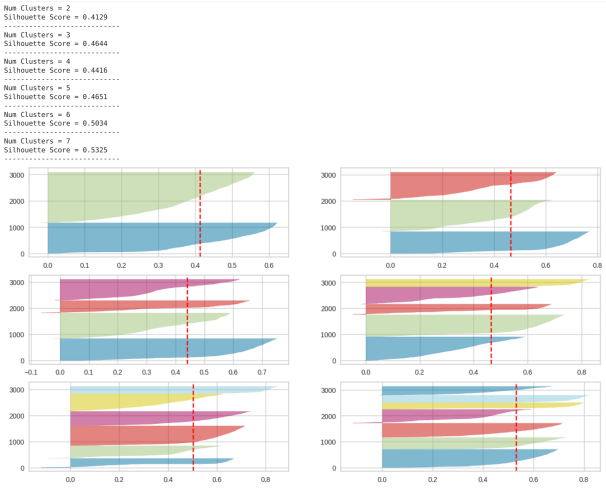

Fig. 27. TSNE Kmeans

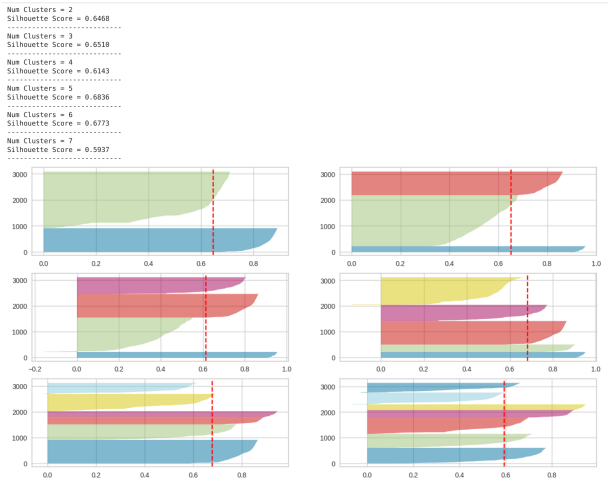

Fig. 29. UMAP Kmeans

Dataset Expansion: As the Muraro dataset has high dimensions but relatively small compared to current problems, future research can consider integrating larger and diverse datasets of brain cells to validate the proposed methodology. This can help assess the scalability of the dimensionality reduction techniques and clustering algorithms.

C. *Open Problems*

- Generalization : Generalization ability of dimensionality reduction algorithms data that is unseen is big challenge. The reduced-dimensional representations learned from a particular dataset may not be generic to new data points that come from a similar but varied distribution. Developing techniques that can improve the generalization performance of dimensionality reduction algorithms is an active research area [21].
- Interpretability: dimensionality reduction algorithms can effectively reduce the dimensions of dataset, however the challenge is to interpret the resulting reduced-dimensional representations . Understanding the meaning and significance of the reduced dimensions is important for interpreting and analyzing the data accurately. Developing techniques to provide meaningful interpretations of the reduced dimensions is an open problem [22].
- Preserving Structure: Dimensionality reduction algorithms have different approach when it comes to preserving structure of the data . For instance, primary focus of algorithms t-SNE and UMAP is to preserver the local relationships between data points, while PCA aims to capture the overall structure of the data. Finding a good balance between preserving both the local and global structures in the reduced-dimensional representations is a challenging task [23].

D. *Contribution*

Spencer Briguglio - Implemented Autoencoder as a method of reducing dimensionality of the input data, from Muraro *et al*, via encoding to a smaller dimension latent-space. Muraro dataset pre-processing. Completed the Abstract, Introduction, Problem Definition, and Summary sections with contributions to Results & Discussion.

Nour ElKott - Implemented t-SNE and UMAP dimensionality algorithms on the Muraro dataset. Completed the sections of the report covering the abstract, justifications, materials and data, proposed methods, as well as the discussion of t-SNE and UMAP results.

Zeeman Memon - Implemented PCA and K-PCA on the Muraro dataset, along with documentation and discussion of the PCA and K-PCA results Completed Summary and assisted in the completion of formal complexity, motivation, future research, and open problems sections of the report.

Karan Vishavjit - Researched and finalised the dataset that is best suited for the dimentionality reduction, Assisted in implementation of PCA, kPCA, Literature review of all related works. Completed section Motivaiton, Related Works, Formal Complexity, Future research, Open Problems.

REFERENCES

[1] M. Ekman, Learning Deep Learning Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using Tensorflow. Boston: Addison-Wesley, 2022.

[2] W. Jia, M. Sun, J. Lian, and S. Hou, 'Feature dimensionality reduction: a review', Complex & Intelligent Systems, vol. 8, no. 3, pp. 2663–2693, 2022.

[3] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of Dimensionality: A Review," International Journal of Automation and Computing, vol. 14, no. 5, pp. 503–519, 2017. doi:10.1007/s11633-017-1054-2

[4] W. Jia, M. Sun, J. Lian, and S. Hou, 'Feature dimensionality reduction: a review', Complex & Intelligent Systems, vol. 8, no. 3, pp. 2663–2693, 2022.

[5] A. M. Lesk, Introduction to Bioinformatics. Oxford: Oxford University Press, 2019.

[6] N. Fleming, 'How artificial intelligence is changing drug discovery', Nature, vol. 557, no. 7706, pp. S55–S55, 2018.

[7] J. D. Wu, K. Haugk, L. Woodke, P. Nelson, I. Coleman, and S. R. Plymate, 'Interaction of IGF signaling and the androgen receptor in prostate cancer progression', Journal of cellular biochemistry, vol. 99, no. 2, pp. 392–401, 2006.

[8] B. Wen et al., 'Deep learning in proteomics', Proteomics, vol. 20, no. 21–22, p. 1900335, 2020.

[9] Chen, X., Teichmann, S. A. & Meyer, K. B. From tissues to cell types and back: single-cell gene expression analysis of tissue architecture. Annu. Rev. Biomed. Data Sci. 1, 29–51 (2018).

[10] Kiselev, V. Y., Andrews, T. S. & Hemberg, M. Challenges in unsupervised clustering of single-cell rna-seq data. Nat. Rev. Genet. 20, 273–282 (2019).

[11] Dong, C. et al. Comprehensive review of the identification of essential genes using computational methods: focusing on feature implementation and assessment. Briefings bioinformatics 21, 171–181 (2020).

[12] Duò, A., Robinson, M. D. & Soneson, C. A systematic performance evaluation of clustering methods for single-cell rna-seq data. F1000Research 7, 1141 (2020).

[13] Pearson, K. (1901). 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. https://doi.org/10.1080/14786440109462720

[14] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

[15] Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation, 10(5), 1299–1319. https://doi.org/10.1162/089976698300017467

[16] L. Ruff et al., "A Unifying Review of Deep and Shallow Anomaly Detection," in Proceedings of the IEEE, vol. 109, no. 5, pp. 756-795, May 2021, doi: 10.1109/JPROC.2021.3052449.

[17] Ruineihart, D., & Williams, R. (1985). LEARNING INTERNAL REPRESENTATIONS BERROR PROPAGATION two. https://apps.dtic.mil/sti/pdfs/ADA164453.pdf

[18] Bank, D., Koenigstein, N., & Giryes, R. (2021). Autoencoders. ArXiv:2003.05991 [Cs, Stat], abs/2003.05991. https://arxiv.org/abs/2003.05991

[19] McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv.org. https://arxiv.org/abs/1802.03426

[20] Diaz-Papkovich, A., Anderson-Trocmé, L., & Gravel, S. (2020, October 14). A review of UMAP in population genetics - Journal of Human Genetics. Nature. https://doi.org/10.1038/s10038-020-00851-4

[21] Belkin, M., & Niyogi, P. (n.d.). Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. https://proceedings.neurips.cc/paper_files/paper/2001/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf

[22] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2018). Feature Selection. ACM Computing Surveys, 50(6), 1–45. https://doi.org/10.1145/3136625

[23] Nl, L., & Hinton, G. (2008). Visualizing Data using t-SNE Laurens van der Maaten. Journal of Machine Learning Research, 9. https://www.cns.nyu.edu/events/spf/SPF_papers/JMLR_Final.pdf

[24] Understanding UMAP. (n.d.). Pair-Code.github.io. https://pair-code.github.io/understanding-umap/