A Novel Deep Learning Framework to Identify Latent Neuroendophenotypes from Multimodal Brain Imaging Data

by

Yitao Wu

Bachelor of Science, South China University of Technology, 2019

Submitted to the Graduate Faculty of the

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Yitao Wu

It was defended on

April 8, 2021

and approved by

Liang Zhan, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Zhi-Hong Mao, Ph.D., Professor, Department of Electrical and Computer Engineering

Jingtong Hu, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Thesis Advisor: Liang Zhan, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

A Novel Deep Learning Framework to Identify Latent Neuroendophenotypes from Multimodal Brain Imaging Data

Yitao Wu, M.S.

University of Pittsburgh, 2021

The expertise required to ensure adequate treatment for patients with complex cases is significantly deficient, which leads to the high demand for subtyping or clustering analysis on different clinical situations. The identification and refinement of disease-related subtypes will support both medical treatments and pathological research. Clinically, clustering can narrow down the possible causes and provide effective treatment options. However, the clustering on non-invasive multimodal brain imaging data has not been well addressed.

In this thesis, we explore this clustering issue using a deep unsupervised embedded clustering (DEMC) method on multimodal brain imaging data. T1-weighted magnetic resonance imaging (MRI) features and resting-state functional MRI-derived brain networks are learned by a sparse autoencoder and a stacked autoencoder separately and then transformed into the embedding space. Then, the K-Means approach was adopted to set the initial center of the deeply embedded clustering structure (DEC) as the centroids, after which DEC clusters with the KL divergence. In the entire processing, the deep embedding and clustering are optimized simultaneously. This new framework was tested on 994 subjects from Human Connectome Project (HCP) and the results show that this new framework has better clustering performance in comparison with other benchmark algorithms.

Table of Contents

List of Tables

List of Figures

1.0 Introduction

Clustering, one of the most important and fundamental methods in data analysis and data visualization, has been thoroughly investigated today. Its primary goal is to classify samples into groups based on degrees of similarity. In contrast to other data reduction approaches, such as factor analysis, clustering produces groupings that rely on similarities in whole cases rather instead of individual variables that those cases consist of. Clustering is a powerful analytic technique in health sciences that can build profiles of both patients and customers, as well as their classification schemes and taxonomies.

Multimodal databases define objects across several dimensions that are called models. Each object in a modal has a set of properties that describes it. Multimodal clustering is a branch of clustering that divides multimodal data into disjointed classes without supervision. In recent years, multimodal clustering has become increasingly widespread. Matrix decomposition algorithms, correlation-based representation learning algorithms[1], and subspace multimodal algorithms[2] are some of the multimodal clustering algorithms that have been proposed.

Traditional clustering algorithms struggle with multimodal data. The methods critically depend on the input data. Different datasets require different correlation-based measurements and classification techniques and is significant for data clustering algorithms regarding the concept of distance or similarity. Distance is determined by the data in the feature space (e.g., K-means clustering[3]). The application-specific information of feature space is generally left to the end-user to decide, but the choice of feature space is clearly important. Clustering raw pixels with Euclidean distance is entirely ineffective for all but the simplest image datasets. These greatly influence on multimodal problems since the multimodal data can be well combined in the latent space but may not be well combined in the raw space. Therefore, dimension reduction and representation learning work in clustering applications simultaneously map the input data to feature space and separates them easily. For the sake of coordination goal, the Deep Neural Networks (DNN) method is raised. With DNN, it is possible to learn non-linear mappings that allow data to be converted into more cluster-friendly representations without having to manually extract and pick features. Further, if we want to build up an unsupervised framework, then autoencoder, which is a good choice, can map the raw data into feature space by reconstructing the data itself in an unsupervised way. For instance, [7] it uses representation learning. Reconstruction loss is first used to help the autoencoder learn the data representation and then implement the K-means. In addition, [4]leveraging a variational autoencoder can learn the better representations for clustering data.

We have drawn on recent literature on deep learning for computer vision[5], where learning better features resulted in significant improvements to benchmark tasks; however, these improvements were accomplished by supervised learning, while our work aims to cluster unsupervised results. To do so, we defined a parameterized non-linear mapping to optimize a clustering goal from the data space X to a feature space V which has a lower dimensionality. In comparison to previous work that operated on a shallow linear embedded space or a data space, we use stochastic gradient descent (SGD) via backpropagation on a clustering objective to learn the mapping ($X \rightarrow V$). A deep neural network can parameterize the mapping. Deep Embedded Multimodal Clustering, or DEMC, is the name assigned to this clustering algorithm framework.

It is difficult to optimize DEMC. The goal is to solve both the cluster assignment and underlying feature representation simultaneously. Unfortunately, we are not able to use labeled data to train the deep network. It is an unsupervised task that learning a cluster-specific representation without the use of ground truth cluster labels. Instead, we recommend iteratively refining clusters using an auxiliary target distribution arising from the most recent soft cluster assignment.

In this paper, we are interested in identifying latent neuroendophenotypes from multimodal brain imaging data. Small contributions from multiple genes are likely to be involved in many facets of brain circuitry and their relationship to behavior, rather than dominant contributions from just one or a few genes. As a result, identifying associations between brain circuit phenotypes and genotypes would necessitate a significant number of subjects.

The Human Connectome Project (HCP)[6] seeks to refine neuroimaging techniques and collect a dataset with outstanding levels and reliability for mapping long-distance relations between all regions of the brain in healthy individuals. Specifically, it went on to collect, interpret, and distribute high-resolution multimodal neuroimaging, and behavioral and genotype data from 1200 young adult twins and non-twin siblings in good health.

In our experiment, two types of data are further used to construct an unsupervised multimodal clustering mechanism. Here, we implemented a stacked autoencoder and sparse autoencoder to separately train two different types of data. An autoencoder is similar to spectral clustering. They all aim to find coding information in low-dimensional data and retain the maximum amount of original data information during reconstruction as much as possible. When processing a large amount of data, however, an autoencoder can be used to obtain sparse expressions, which can improve how efficiently data is processed.

The following are the major contributions:

1. We propose a deep embedded multimodal clustering algorithm with two types of autoencoders to combine the feature spaces from two types of data and effectively cluster.

2. We optimized the deep embedding and clustering. It performs better than simply using autoencoders and a traditional clustering algorithm (i.e., K-means).
3. A soft assignment is a novel iterative refining process.
4. Extensive experiments demonstrated the superiority of our multimodal clustering algorithm when compared with other baseline models.

In Section 2, we first present some traditional and popular clustering algorisms and deep clustering methods we implemented. We then discuss our multimodal clustering framework and its details in Section 3. Following that, in Section 4, we discuss the experiments and the results. We draw conclusions in Section 5.

2.0 Background and Related Work

2.1 K-means

K-means is a traditional and commonly used unsupervised clustering algorithm. The K-means algorithm is also quite fast and effective for high-dimensional clustering (in case the dimension does not reach tens of dimensions).

K-means algorithm is sensitive to the initial value, so different cluster partitioning rules may lead to a selection of different initial values. To prevent anomalies created by this sensitivity in the final results, multiple sets of initial nodes can be initialized to create various classification rules, and then the best construction rules can be chosen. As a result, several new algorisms have been proposed: binary K-means algorithm, K-means++ algorithm, Canopy algorithm, etc.[8] In this paper, to initialize the clustering centroid, the DNN initialized by the DEMC will be utilized to pass the data to obtain embedded data points, and then in the feature space Z, regular K-means clustering will obtain K initial centroid values, which is $\{\mu_j\}_{j=1}^{k}$.

The objective function of the K-means algorithm is

$$\overset{min}{(n_1,n_2,\ldots,n_k)}\sum_{i=1}^{n} c^2(X_i, n(X_i)) \tag{2-1}$$

Among them, $n_1, n_2, \ldots, n_k$ are the center points of K clusters, respectively. $n(X_i)$ represents the centroid of the cluster to which X_i belongs to. In general terms, the purpose of the K-means algorithm is to minimize the square sum of the distances between all non-central points in the sample space and the center points of their respective clusters.

The aforementioned issue is an NP-hard problem. Hence, the coordinate descent method, a non-gradient optimization method, is recommended. In each iteration, it searches along the course of a coordinate axis, seeking for the objective function's local minimum value by cycling through various coordinates.

2.2 Deep Learning

While deep learning is a fairly old branch of machine learning, it only took off in the first decade of the 21st century. In the years that followed, it made revolutionary progress, achieving remarkable results on perceptual problems, such as vision and hearing processing[7] that involve technologies that seem natural and intuitive to humans but have long been difficult for machines to solve.

Deep learning's basic strategy is to use this distance value as an input signal to fine-tune the weight values to lower loss values associated with the current case. This adjustment is accomplished by the optimizer, which implements the so-called Back Propagation algorithm, which is the core algorithm to deep learning. In the beginning, the neural network weights were randomly assigned, so the network simply implemented a series of random transformations. The output, of course, is far from ideal, and the losses are correspondingly high, but with more and more examples of network processing, losses are steadily minimized as the weights are gradually fine-tuned in the correct direction. This is the Training Loop, which is repeated enough times (often dozens of iterations over thousands of examples) to produce a weight value that minimizes the loss function. The qualified network is the one with the lowest losses and a performance value that is as similar to the goal value as possible.

2.3 Autoencoder

An autoencoder[9] is a data compression algorithm with data-dependent, lossy compression, and decompression features that are learned automatically from the sample. In cases where autoencoders are specified, the compression and decompression features are usually performed using neural networks.

1) The autoencoder is data-dependent or data-specific, meaning that it can only compress data that is similar to the training data. Since it studies features associated with a human face, an autoencoder trained on a human face is not well suit for compressing other images, such as wood.

2) As with compression algorithms like MP3 and JPEG, the autoencoder is lossy, which means that the decompressed output degrades in comparison to the initial signal. Lossy compression algorithms are not the same.

3) Autoencoders automatically learn from data samples, making it easy to train a particular encoder on the input of a given class without having to implement any additional work.

Setting up an autoencoder requires three things: setting up the encoder, setting up the decoder, and setting up a loss function that measures the information lost due to compression. Encoders and decoders, which are usually implemented when using neural networks, are parameterized equations that are differentiable in terms of the loss function. By minimizing a loss function, such as SGD, the parameters for encoders and decoders can be optimized.

In this paper, the autoencoders are implemented as a sparse autoencoder and stacked autoencoder.

The sparse autoencoder simply combines the sparse penalty $\mathrm{p(h)}$ of the coding layer and reconstruction errors during training, which are represented as

$$L\left(x, T\big(f(x)\big)\right) + \mathrm{p(h)} \qquad (2\text{-}2)$$

Where $T(H)$ is the decoder output, usually H is the encoder output, that is, $H = f(x)$. Sparse autoencoders are typically used to learn features for tasks, such as classification. The sparsely regularized autoencoder must represent the special statistical features from the training data collection rather than merely serving as an identification function. Trained in this way, replication tasks with sparse penalties can yield models that learn useful features.

Generally speaking, the number of nodes in the autoencoder hidden layer is less than the number of nodes in the input layer, that is, in the training process, the autoencoder tends to learn the internal rules of the data, such as correlation, then the learning results for the autoencoder are likely to be similar to PCA, and what it obtains is a dimensionality reduction representation of the input data[10].

Assuming that the hidden layer activation function adopts Sigmoid, then a hidden layer output of 1 means that the node is active, and a hidden layer output of 0 means that the node is inactive. Based on this, we introduced KL dispersion[14] to determine the degree of similarity between a hidden layer node's average activation performance and the set sparsity

$$KL(q\|\widehat{q_J}) = qlog\frac{q}{\widehat{q_J}} + (1-q)log\frac{1-q}{1-\widehat{q_J}} \qquad (2\text{-}3)$$

Where $\widehat{q_J} = \frac{1}{n}\sum_{i=1}^{n}[a_j^{(2)}x^{(i)}]$, n is the number of samples for training.

Hence, we can apply KL dispersion to the loss function as a regular term to constrain the sparse rows of the entire network coding:

$$J_{sparse}(V,b) = J(V,b) + \beta\sum_{j=1}^{s2} KL(q\|\widehat{q_J}) \qquad (2\text{-}4)$$

Stack autoencoders are multiple cascaded autoencoders to complete feature extraction layer by layer. The final features obtained are more representative and have small dimensions. The training process for the stacked autoencoder is: N AEs are trained in sequence. After the first AE is trained, the output of its encoder is used as the input for the second AE, and so on. The final feature is used as the input for the classifier to complete final classification training.

The t-SNE algorithm[11] is a dimensionality reduction machine learning algorithm. It is also a nonlinear dimensionality reduction technique, which is quite suitable for reducing dimensionality in high data to very low dimensions for visualization. Compared with PCA, it can be considered to be a more advanced and effective method.

3.0 The Proposed Methodology

3.1 The Framework of Deep Embedded Multimodal Clustering (DEMC)

The structure used in this paper consists of three parts, namely, two autoencoders (stacked autoencoder and sparse autoencoder), and deep embedded clustering (DEC).

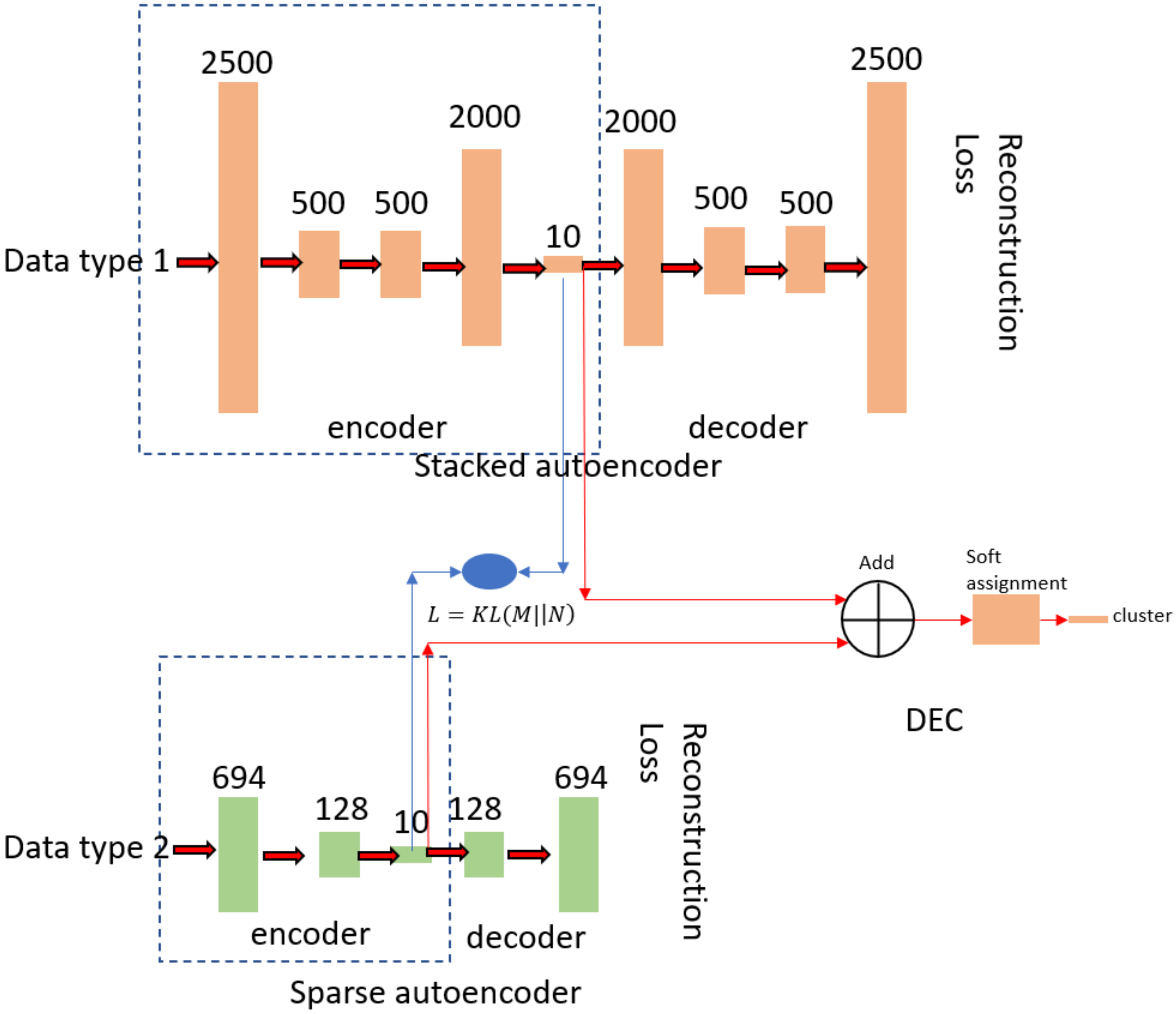

Figure 1 The basic structure of deep multimodal clustering

We formed autoencoders in the deep structure, which consisted of an encoder and decoder[12]. All layers are connected one after the other. The autoencoder is fine-tuned to reduce reconstruction losses, after greedy layer-by-layer training. Multiple layers of deep autoencoders are used at the end as well as a bottleneck coding layer in the center. The encoder layer is then used to replace the decoder layer and initialize the mapping of the space between the space and feature.

The two types of data are passed through autoencoders separately and form two latent feature spaces. The first job of DEC structure is to update the parameters in the encoder layers and optimize feature space by minimizing the losses in KL divergence. The M and N in figure 1 respectively represent feature spaces from data type 1 and type 2.

In the meantime, the second job of DEC is to merge two latent feature spaces by addition. Next, the new fusion feature space initializes the cluster centroids by K-means. Then, after having the greatest similarity between the two feature spaces, the probability of each sample in each cluster is obtained based on soft assignment, and the maximum value is taken as the cluster in which the sample is located.

Table 1 displays the algorithm for the entire process.

Table 1 Algorithm of deep embedded multimodal clustering

	Algorithm 1: Training process for deep embedded multimodal clustering
	Input: Input data type 1 and data type 2;
	Output: Clustering results in C;
1	Build the training datasets A and B based on data types 1 and 2 by data cleaning and normalization.
2	Construct the sparse autoencoder, stacked autoencoder, and deep embedded structure.
3	Put A and B into the sparse autoencoder and stacked autoencoder separately and pre-train them based on reconstruction loss. The max iteration is 200.
4	Discard the decoder parts from the autoencoders.
5	Update the parameters in the encoder layers and optimize feature space by minimizing the losses in KL divergence. The max iteration is 200.
6	Meanwhile, merge the feature spaces by addition and use K-means to initial the clustering centroids of DEC.
7	Obtain the probability of each sample in each based on soft assignment.
8	Select the highest possibility of cluster assignment for each embedded point to form C.

3.2 Stacked autoencoder

We used a stacked autoencoder (SAE) to initialize the DEMC for feature extraction on the first type of data, since studies on real-world datasets[12] consistently generate semantically correct and well-separated representations. As a result, understanding DEMC cluster representation is aided by an unsupervised representation of SAE learning.

Each layer of the SAE network is a denoised autoencoder that can be trained to recreate the performance of the previous layer after it has been randomly destroyed[13]. The noise reduction autoencoder is a two-layer neural network with the following definition:

$$\tilde{x} \sim Dropout(x) \qquad (3\text{-}1)$$

$$j = \ h_1(w_1\tilde{x} + b_1) \tag{3-2}$$

$$\tilde{j} \sim Dropout(j) \tag{3-3}$$

$$y = \ h_2(w_2\tilde{j} + b_2) \tag{3-4}$$

Where $Dropout(.)$ [5] is a random mapping that moves one input dimension component to 0 at random. h_1 is the activation function in encoder layers, while h_2 is in decoder layers. $\delta = \{w_1, w_2, b_1, b_2\}$ are model parameters. To prepare, the least square loss $\|x - y\|_2^2$ is reduced. We train the next layer using the output j of the previous layer as data. Except for h_2 in the first pair (which includes reconstructing the input data; zero-mean images, for example, have both positive and negative values.) and h_1 in the last pair (as a result, data embedding in its final state preserves all of the information), we use ReLUs in all encoder/decoder pairs[13].

3.3 Sparse autoencoder

First, we set the degree matrix $C = diag\{c_1, c_2, \dots, c_n\}$, where $c_i$is the degree of node i. We should treat S as the training set containing n instances if we have an n-node graph G and its similarity matrix S. It is important to note that $s_i = \{s_{ij}\}, j = 1,2, .., n$. The normalized training set $C^{-1}S$ is then fed into the deep neural network, with the output features in the deepest layer of the DNN serving as embedded points.

Optimization seeks to narrow the gap between the initial data x_i and the reconstructed data y_i from the current representation j_i,

$$\arg \min_{\delta \in \sigma} \sum_{i=1}^{n} \|y_i - x_i\|_2 \tag{3-5}$$

In the hidden layer, we also add sparsity restrictions to the activation. In other words, we take the reconstruction error and apply a regularization term to it[15],

$$Loss(\delta) = \sum_{i=1}^{n} \|y_i - x_i\|_2 + \beta KL(q||\hat{q}) \tag{3-6}$$

Standard back-propagation algorithms can be used to solve[12]. After the current layer has been trained, the hidden layer activations are used as inputs to prepare the next layer. The Stacked Sparse Autoencoder [12] is based on this greedy layer-wise training method. Since all of the layers have been learned in this manner, we use the output from the final layer to represent feature space.

In certain real-world large-scale graph clustering problems, it is advantageous to seek such sparse representations as graph embedding; This will dramatically increase system performance in terms of both storage and data transmission; however, it typically increases clustering precision by removing any noisy information that degrades clustering performance.

3.4 Clustering with KL divergence

The n-point $\{x_i \in X\}_{i=1}^{n}$clustering problem is considered to be separated into k groups, with a centroid $\emptyset_j, j = 1, \ldots, k$. Rather than clustering directly in the clustering space X, we first use nonlinear data mapping f_δ :X →V, where V is the new feature space and δ is the trained parameter. V's dimensionality is usually smaller than X's.

To improve clustering, we consider utilizing an unsupervised algorithm that switches between two stages. First, a soft assignment is computed between the points that are embedded through autoencoders and the cluster centroids initialized by K-means. Next, deep mapping f_δ is upgraded, and the cluster centroids are optimized by learning from current high confidence assignments based on a new target distribution. This loop is carried out before the convergence condition is fulfilled.

3.4.1 Soft Assignment

The resemblance between embedded point v_i and centroid $\emptyset_j$ is calculated by using a kernel, which is the Student's t-distribution, as seen in[17]:

$$s_{ij} = \frac{\left(1+\frac{\left\|v_i-\emptyset_j\right\|^2}{\sigma}\right)^{-\frac{\sigma+1}{2}}}{\sum_{j'}\left(1+\frac{\left\|v_i-\emptyset_{j'}\right\|^2}{\sigma}\right)^{-\frac{\sigma+1}{2}}} \tag{3-7}$$

Where σ are the degrees of freedom for the Student's t-distribution. $v_i = f_\delta(x_i) \in V$ corresponds to $x_i \in X$. s_{ij} represents the possibility that a sample i is assigned to cluster j. We set σ = 1 on all tests because we are not able to cross-validate the unsupervised situation and understand it is not necessary[17].

3.4.2 KL Divergence Minimization

With the aid of an auxiliary goal distribution, we recommend that the clusters be refined iteratively by learning from their high confidence assignments. Our proposed model is precisely trained by corresponding the soft assignment to the target distribution. As a result, the goal is described as a loss in KL divergence, which is the difference between the soft assignments m_i and the auxiliary distribution n_i:

$$L = KL(M||N) = \sum_i \sum_j m_{ij} log \frac{m_{ij}}{n_{ij}} \tag{3-8}$$

DEC's output is dependent on the target distributions M that are chosen.

In our experiments, we used this formula to calculate m_i :

$$m_{ij} = \frac{{n_{ij}}^2/f_j}{\sum_{j'} {n_{ij'}}^2/f_{j'}} \qquad (3\text{-}9)$$

Where $f_j = \sum_i n_{ij}$ are soft cluster frequencies.

We start with an unlabeled dataset and an initial classifier, then label the dataset with the classifier, so that it can practice on its high confidence estimates, as in self-training. Indeed, studies show that DEC builds on the initial estimates from every iteration by learning from high-confidence predictions, which enhances low-confidence predictions.

3.4.3 Optimization

The cluster centroids $\{\emptyset_j\}$ and $\delta,$ the deep neural network parameters, are optimized collectively. During this process, Stochastic Gradient Descent (SGD) with momentum is implemented. The gradients of L for embedded point v_i and centroid $\emptyset_j$ are computed:

$$\frac{\partial L}{\partial v_i} = \frac{\sigma+1}{\sigma} \sum_j (1 + \frac{\left\|v_i - \emptyset_j\right\|^2}{\sigma})^{-1} \times (m_{ij} - n_{ij})(v_i - \emptyset_j) \qquad (3\text{-}10)$$

$$\frac{\partial L}{\partial \emptyset_j} = -\frac{\sigma+1}{\sigma} \sum_i (1 + \frac{\left\|v_i - \emptyset_j\right\|^2}{\sigma})^{-1} \times (m_{ij} - n_{ij})(v_i - \emptyset_j) \qquad (3\text{-}11)$$

The $\frac{\partial L}{\partial v_i}$ gradients are passed through the deep neural network, where they are working in regular backpropagation (BP) to calculate the DNN gradient $\frac{\partial L}{\partial \emptyset_j}$. In two iterations, if fewer than 10% of points change the cluster assignment, we avoid our protocol to avoid covering cluster assignments.

4.0 Experiments

4.1 Dataset

In our experiment, two types of data are jointly used to build a multimodal clustering mechanism.

The first type is feature variables from the gray matter using T1-weighted MRI(T1w MRI). Centered on the Desikan-Killiany atlas[9], FreeSurfer was used to collect 136 measurements for 68 brain ROIs, including cortical volumes and widths. The number of subjects totaled 1104. After preprocessing, such as data cleaning, some subjects with missing feature variables as well as some repeated feature variables are discarded. Finally, the first type of T1 data is transformed into the size 1104*694.

The second type is some young men's resting-state functional magnetic resonance imaging (fMRI)[17] connectome from the HCP (released in December 2015, named HCP900 Parcellation+Timeseries+Netmats,https://db.humanconnectome.org/data/projects/HCP_900)[18]. The dimensions of the network are 50 × 50, derived with ICA. There are 1003 subjects in total, 502 men and 501 women. To fit the number of subjects, we chose 994 subjects that correspond to the first type of data. After preprocessing, the second type of fMRI data is transformed into the size 994*2500.

We use some specific features in HCP clinical measurements to evaluate the effects from the clustering algorithm. The feature names and descriptions are in Table 2. And the mean±std is calculated in Table 3.

Table 2 Features used to evaluate the multimodal clustering algorithm

Feature Name	Feature Abbreviation	Description
ASR Somatic Complaints Raw Score	ASR_Soma_Pct	ASR Somatic Complaints Raw Score
ASR Anxious/Depressed Gender and Age Adjusted Percentile	ASR_Anxd_Pct	ASR Anxious/Depressed Gender and Age Adjusted Percentile
ASR Sum of Thought, Attention, and Other Problems Raw Score	ASR_TAO_Sum	ASR Sum of Thought, Attention, and Other Raw Score
ASR Internalizing Gender and Age Adjusted T-score	ASR_Intn_T	ASR Internalizing Gender and Age Adjusted T-score
Blood Pressure - Systolic	BPSystolic	Systolic blood pressure of subject. A normal systolic blood pressure is below 120.
ASR Total Problems Raw Score	ASR_Totp_Raw	ASR Total Raw Score

Table 3 Feature information (mean±std)

Feature Abbreviation	mean±std
ages	28.71±3.72
ASR_Soma_Pct	53.87±5.83
ASR_Anxd_Pct	53.84±6.21
ASR_TAO_Sum	17.48±9.47
ASR_Intn_T	48.41±10.58
BPSystolic	123.39±13.83
ASR_Totp_Raw	36.66±22.25

4.2 Evaluation metric

Because the experimental object in this study is an unlabeled data set, the traditional accuracy or NMI methods become ineffective. Here we use unsupervised evaluation methods, such as the contour coefficient, Calinski Harabasz score, etc. The Silhouette Coefficient range is fixed, which is from -1 to 1. Higher means better, and value that is near 0 indicates that a cluster is overlapped. If it is smaller than 0, it shows that samples are in the wrong cluster, since different clusters have a higher degree of similarity. This indicator can be used to visually see the clustering effect. Although the Calinski Harabasz score is quicker to calculate, the final expression results have no upper limit and are not intuitive.

For each point I in a cluster, the Silhouette Coefficient can be formed as

$$S(i) = \frac{t(i)-k(i)}{max\{k(i),t(i)\}} \tag{4-1}$$

Where $k(i)$is the inner-cluster similarity, and $t(i)$ is the outer-cluster similarity.

The dissimilarity is calculated by Euclidean distance. The final step is to calculate the mean of $S(i)$ to obtain the Silhouette Coefficient for the final clustering effect. When the contour coefficient is higher, the clustered instances are more compact.

The obtained predictive clustering labels are utilized to evaluate the clustering effect of samples in clinical measurements.

4.3 Benchmark Algorithms

To highlight the advantages in our approach over existing approaches, as Benchmark Algorithms, we have chosen the following clustering techniques.

1) PCA+K-means. We first merge the data to get the input data of 3194*994, then use PCA to reduce the dimensions, retain 99% of the principal components and reduce to 640 dimensions, and ultimately conduct K-means.

2) Autoencoder+K-means. Different from the experiment we proposed, this method only uses the autoencoder to extract the features and directly uses K-means clustering after extracting the features of the two data sets and dimensioning.

4.4 Implement and Results

First, we calculate the Hopkins Statistic.

The Hopkins Statistic is a spatial statistic that examines the spatial randomness of variables spatially distributed to determine whether or not a data set includes a nonrandom structure. When clustering approaches are blindly applied to datasets, they can produce misleading clusters.

Clustering pattern evaluation determines if a dataset has a nonrandom structure that allows for meaningful clustering. Although the clustering algorithm will return clusters for a data set with no nonrandom structure, such as randomly distributed points in the data space, these clusters are pointless and spontaneous. Clustering necessitates a non-uniform data distribution.

The formula for Hopkins Statistic is

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i} \tag{4-2}$$

y_i is calculated by generating another uniform distribution, sampling data from this distribution, and calculating the nearest distance to the data point in the data set D. So, if D also has uniform distribution, H is going to be close to 0.5. If the data in D is not uniformly distributed, H will be greater than 0.5, so clusters may exist.

Second, we need to find the number for K, which is the number of clusters. The BIC test and Elbow Method are going to be employed here[19][21].

The formula for BIC is:

$$B = -2 \times \ln(likehood) + \ln{(N)} \times k \tag{4-3}$$

Where N is the number of samples in the data set, and k is the number of features. BIC is a metric that determines how well a model fits and how complicated it is to fit. The section $-2 \times \ln(likehood)$ in the calculation formula is a metric for how well a model fits the data. A higher value indicates a worse suitability, and model complexity is measured by $\ln{(N)} \times k$.

The likelihood function is generally calculated by probability, which is a small value between 0 and 1. Combined with the image of the ln function, it can be seen that ln(likelihood) is a negative number, and a smaller likelihood function indicates a corresponding negative number has a larger absolute value. Therefore, a greater $-2 \times \ln(likehood)$ means the model fitting degree is worse.

For the elbow method[20], we all know that the objective function of K-means is to keep the square error between the sample and the particle as low as possible, and distortions are errors in distance between the particles and sample points in the cluster. As a result, a lower distortion degree indicates the cluster members are that much closer, and a higher distortion degree means that a cluster structure is softer. The degree of distortion declines as the number of categories increases, but for data with a high degree of distinction, the degree of distortion improves dramatically before it reaches a critical point, after which it gradually decreases. This critical point can be called a strong clustering point.

Third, the parameters of the stacked autoencoder are set up.

Due to the unsupervised clustering algorithm that we performed, we could not determine the hyperparameters by cross-validating with the validation set.

We used standard DNN parameters in the stacked autoencoder and avoided tailoring the dataset as much as possible. More precisely, we set the network size for the type of fMRI data to 2500-500-500-2000-6, which will differ between datasets. All of the layers are tightly (completely) connected.

We randomly initialized the weights derived from a Gaussian distribution with a mean of 0 and a standard deviation of 0.01 during the greedy layer-by-layer pre-training. Each tier was pre-trained in 50,000 iterations and had a 20% dropout rate. Without falling out, the depth autoencoder was run for another 100,000 iterations. The minimum batch size is set to 128, the beginning learning rate is set to 0.1, every 20,000 iterations is separated by 10, and the weight attenuation is set to 0 for layer-by-layer pre-training and end-to-end detection of the autoencoder. To obtain a rational reconstruction loss, set one of the aforementioned criteria and maintain it for all data sets. While dataset-specific parameter adjustments can improve the output for each dataset, we avoid such impractical parameter adjustments to improve the model's overall reliability.

Fourth, the parameters of the sparse autoencoder are set up.

The data type of T1w MRI is passed in a network with a scale of 694–128–6 in our experiments.

Two autoencoders parameters were tweaked: the sparsity penalty (0.01), which balances the weights of the two errors during optimization, and the sparsity target (0.5), which is the highest value that the hidden layer activations may reach. The neural networks feature three layers. As activations of each layer, the neural network uses element-wise Sigmoid functions. The learning rate is 0.1 and is iterated 200 times.

Finally, when it comes to the DEC, to initialize the centroid, we ran K-means 20 times for pre-training and picked the best clustering core after combining the outputs from the two AE. We practiced at a steady rate of learning of 0.01 in the KL divergence minimization step. TOL = 0.1% is the convergence threshold.

The following are the presentation of the results.

1. Hopkins Statistic

H = 0.71, which is greater than 0.5. It proves that the original dataset has a rather proper suitability for clustering.

2. BIC

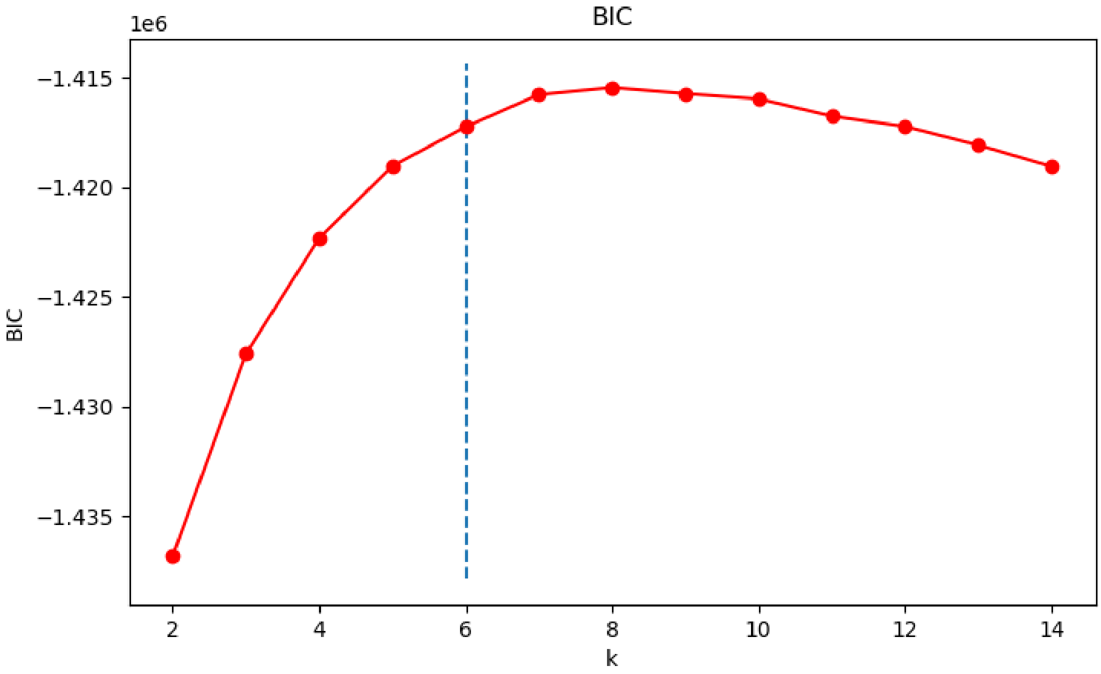

Figure 2 Relationship between the number of clusters and BIC

As shown in Figure 2, the k value with the fastest changing curvature first appears at 6, so for clustering this data set, the best clustering number should be 6.

3. Elbow method

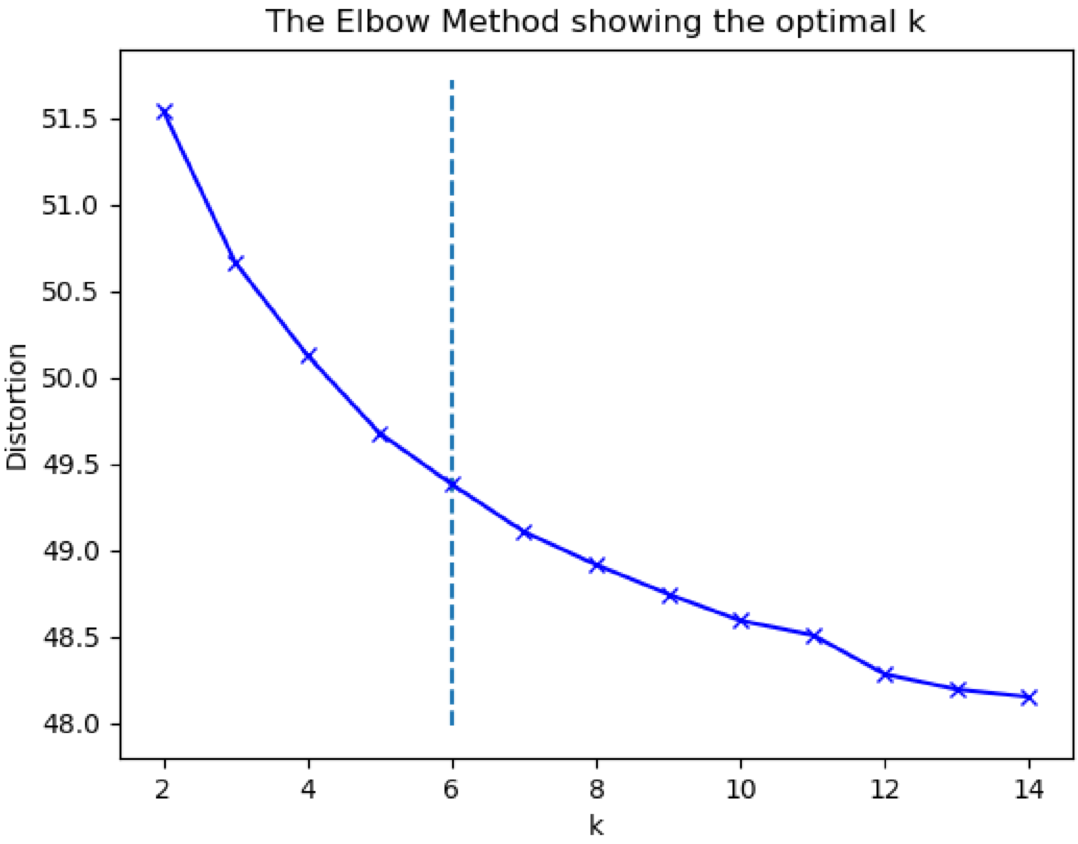

Figure 3 The relationship between the number of clusters and distortion

As shown in Figure 3, obviously, when k=6, the distortion value has the highest curvature. Therefore, the subjects in the two types of data can be properly assigned to 6 clusters.

The k value is the same in the elbow method and BIC test. We will use k = 6 to implement clustering.

4.Silhouette Coefficient

Through the experiment, we obtained the changes in the Silhouette Coefficient after 200 training times.

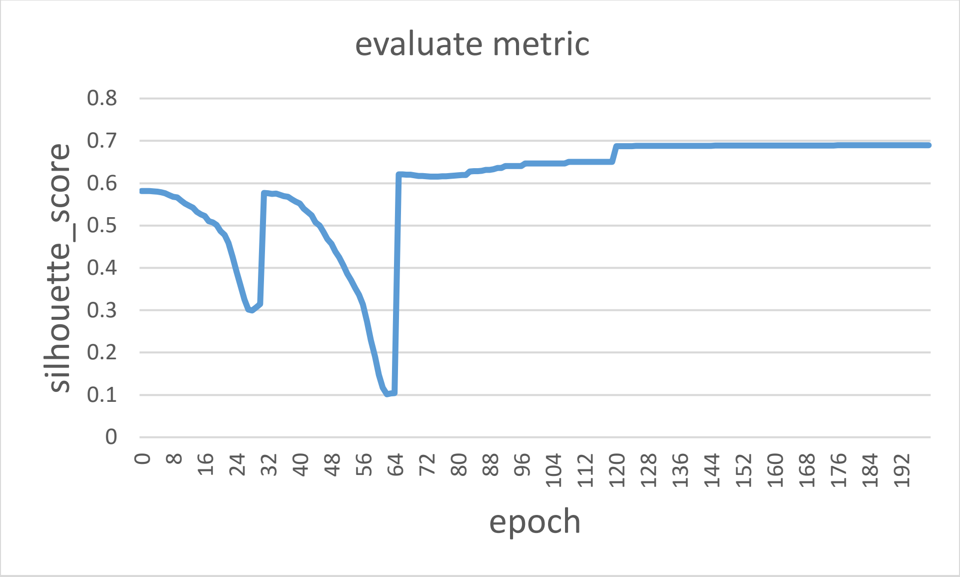

Figure 4 The values of Silhouette Coefficient while training

As shown in Figure 4, after the initial fluctuations, the Silhouette Coefficient gradually stabilized at 0.69.

In Figure 5, what we can learn is that the 994 subjects in the two types of data can be roughly separated into 6 clusters. The figure is generated by t-SNE.

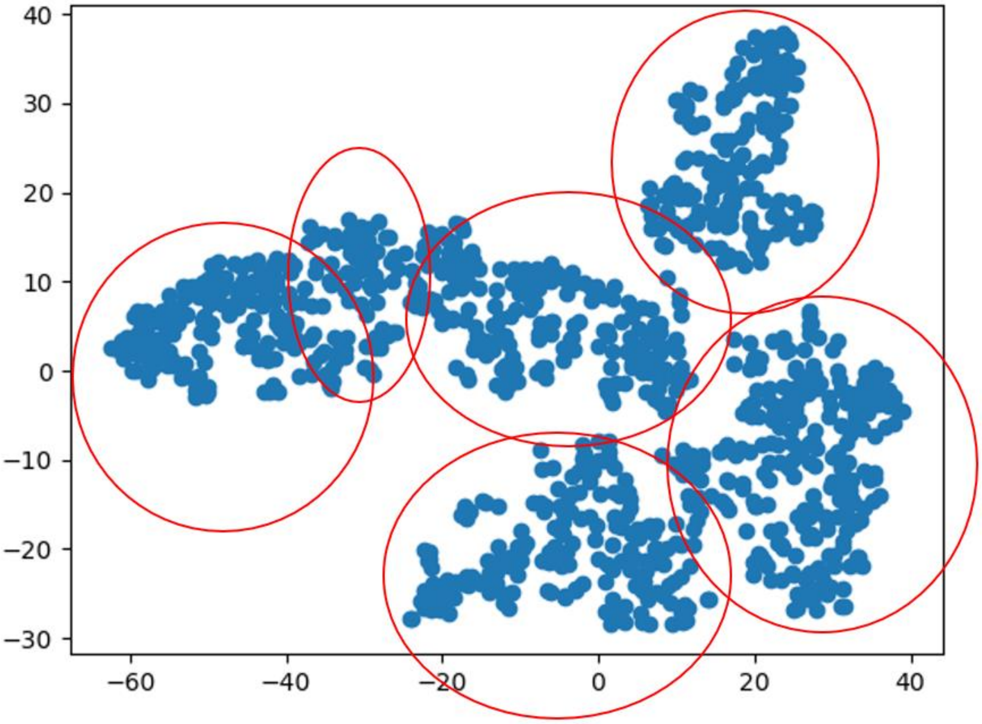

Figure 5 The effect of clustering via t-SNE

Finally, the obtained predictive clustering labels are utilized to evaluate the clustering effects of samples in the clinical measurement from Table 4. We selected these features according to the variances from low to high. Based on the Silhouette Coefficient of the specific feature, we discovered that our multimodal clustering algorithm performs better on features with greater variance.

Table 4 Clustering evaluation of some typical features in clinical measurement

Feature Name	Variance	Silhouette Coefficient
ASR_Soma_Pct	34.48	0.37
ASR_Anxd_Pct	38.58	0.56
ASR_TAO_Sum	89.84	0.74
ASR_Intn_T	111.99	0.79
BPSystolic	191.29	0.81
ASR_Totp_Raw	495.18	0.94

5. Benchmark comparison

Table 5 The value of Silhouette Coefficient in typical features among models

Method	PCA +K-means	Autoencoder +K-means	Our model
ASR_Soma_Pct	0.09	0.31	**0.37**
ASR_Anxd_Pct	0.11	0.53	**0.56**
ASR_TAO_Sum	0.17	0.69	**0.74**
ASR_Intn_T	0.17	0.72	**0.79**
BPSystolic	0.22	0.78	**0.81**
ASR_Totp_Raw	0.27	0.82	**0.94**

From Fable 5, the suggested approach outperforms many state-of-the-art baselines based on experimental findings from our dataset.

5.0 Conclusions

In this paper, we proposed a novel unsupervised Deep Embedded Multimodal Clustering (DEMC) framework to enable learning for a cluster-specific representation of multimodal brain image data without using ground truth cluster labels. Data points in two types of data form the multimodal inputs. They can be embedded from the sparse autoencoder and stacked autoencoder and then jointly merge the feature space. DEMC works by iteratively optimizing clustering targets based on KL divergence using self-training target distributions.

Experimental results show the advantages in this framework. In the presence of unlabeled datasets with different characteristic dimensions, better clustering performance was observed when compared to other benchmark algorithms in specific features from clinical measurements. By using this multimodal clustering method, we can learn that the 994 subjects in two types of multimodal brain image data have a rather strong pattern for forming clusters when the cluster number is 6.

However, some limitations to the proposed method exist. First, there might be a more efficient way to merge the two feature spaces after the autoencoders. Second, we can use more advanced cluster methods rather than K-means to set up the centroids. Third, we should compare the performance of our framework to some other multimodal framework and get more convincing results.

Bibliography

[1] R. Zhou and Y. D. Shen, “End-to-end adversarial-attention network for multi-modal clustering,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 14607–14616, 2020, doi: 10.1109/CVPR42600.2020.01463.

[2] D. B. Neill and H. J. Heinz, “USE SARIA2015 -- this is a copy -- Subtyping: What It Is and Its Role in Precision Medicine,” 2015.

[3] T. Y. Liang L, Zhao JY, Kathryn T, Bekker A, “HHS Public Access,” *Physiol. Behav.*, vol. 176, no. 3, pp. 139–148, 2019, doi: 10.1002/cne.24274.

[4] M. F. Glasser *et al.*, “HHS Public Access,” vol. 19, no. 9, pp. 1175–1187, 2018, doi: 10.1038/nn.4361.

[5] Y. Cai, Z. Zhang, Z. Cai, X. Liu, X. Jiang, and Q. Yan, “Graph convolutional subspace clustering: A robust subspace clustering framework for hyperspectral image,” *arXiv*, pp. 1–11, 2020, doi: 10.1109/tgrs.2020.3018135.

[6] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, “Structural Deep Clustering Network,” *Web Conf. 2020 - Proc. World Wide Web Conf. WWW 2020*, pp. 1400–1410, 2020, doi: 10.1145/3366423.3380214.

[7] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, “Clustering with deep learning: Taxonomy and new methods,” *arXiv*, pp. 1–12, 2018.

[8] A. F. Agarap and A. P. Azcarraga, “Improving k-Means Clustering Performance with Disentangled Internal Representations,” *arXiv*, 2020.

[9] D. M. Y. Adam Moser, Kevin Range, “NIH Public Access,” *Bone*, vol. 23, no. 1, pp. 1–7, 2008, doi: 10.1016/j.neuroimage.2013.05.041.

[10]S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, “ClusterGAN: Latent space clustering in generative adversarial networks,” *arXiv*, 2018.

[11]M. Pal *et al.*, “Speaker Diarization Using Latent Space Clustering in Generative Adversarial Network,” *ICASSP, IEEE Int. Conf. Acoust.*

[12]G. Shiran and D. Weinshall, “Multi-Modal Deep Clustering: Unsupervised Partitioning of Images,” *arXiv*, 2019.

[13]M. Abavisani and V. M. Patel, “Deep Multimodal Subspace Clustering Networks,” IEEE J. Sel. Top. Signal Process., vol. 12, no. 6, pp. 1601–1614, 2018, doi:10.1109/JSTSP.2018.2875385.

[14]D. Hu, F. Nie, and X. Li, “Deep multimodal clustering for unsupervised audiovisual learning,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 9240–9249, 2019, doi: 10.1109/CVPR.2019.00947.

[15]X. Zhang, J. Mu, L. Zong, and X. Yang, “END-TO-END DEEP MULTIMODAL CLUSTERING Dalian University of Technology , Dalian , China , xczhang@dlut.edu.cn Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province , Dalian , China Dalian University of Technology , Dalian , China ,” no. 61632019, pp. 1–6.

[16]Y. Yu and W. J. Zhou, “Mixture of GANs for clustering,” *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2018-July, pp. 3047–3053, 2018, doi: 10.24963/ijcai.2018/423.

[17]J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 1, pp. 740–749, 2016.

[18]Q. Wang *et al.*, “Cohort Validation,” vol. 64, no. 1, pp. 149–169, 2018, doi: 10.3233/JAD-171048.

[19]R. McConville, R. Santos-Rodríguez, R. J. Piechocki, and I. Craddock, “N2D: (not too) deep clustering via clustering the local manifold of an autoencoded embedding,” *arXiv*, 2019.

[20]M. Liu *et al.*, “DIG: A Turnkey Library for Diving into Graph Deep Learning Research,” pp. 1–9, 2021.

[21]X. Guo, X. Liu, E. Zhu, and J. Yin, “Deep Clustering with Convolutional Autoencoders,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10635 LNCS, pp. 373–382, 2017, doi: 10.1007/978-3-319-70096-0_39.