A Systematic Review and Categorization of Loss Functions in Deep Clustering

Xiaobo Huang

College of Big Data Statistics, Guizhou University of Finance and Economics, Guiyang 550000, Guizhou Province, P.R. China

Received: 02 April 2025/ Revised: 11 April 2025/ Accepted: 18 April 2025/ Published: 30-04-2025

Abstract— *Clustering techniques perform the task of discovering underlying patterns and structures in data. They play a crucial role in fields such as big data analytics, recommendation systems, and medical diagnostics, driving intelligent decision-making and efficient data processing. Deep clustering, with its strong ability to extract features, effectively overcomes the shortcomings of traditional clustering techniques, making it a prominent area of current research. Among these methods, the loss function, as the core component of deep clustering, guides the model in optimizing data representation, ensuring the effectiveness and stability of feature extraction from high-dimensional and complex data. However, existing studies primarily focus on the deep learning architecture, with few offering a systematic analysis from the perspective of loss functions. This paper reviews the current state of deep clustering research from the loss function viewpoint and categorizes relevant algorithms based on the characteristics of their loss functions. By analyzing the strengths and weaknesses of various loss functions, four essential elements for an effective loss function are proposed: information preservation, balance, robustness, and scalability. Future research directions are explored with respect to these four aspects.*

***Keywords*— Deep Clustering, Loss Function, Network Loss, Deep Learning, Network Architecture, Clustering Loss.**

I. INTRODUCTION

Clustering is an unsupervised learning method aimed at partitioning a dataset into several groups or clusters such that samples within the same group exhibit high similarity, while samples from different groups show low similarity, following the principle of "birds of a feather flock together." Clustering algorithms do not rely on pre-labeled training data; instead, they uncover the intrinsic similarities within data by analyzing its structure.

As a significant area in machine learning, clustering plays an indispensable role in real-world applications. When the data labels are unknown or difficult to obtain, clustering helps in understanding the inherent structure of the data and uncovering patterns and trends within. It can also be applied in anomaly detection to identify outliers that deviate significantly from the rest of the samples. In image segmentation and object recognition, clustering techniques can group similar regions or objects in images, improving the accuracy of image analysis. Clustering is a versatile tool that simplifies complexity, reveals underlying relationships, and provides powerful support for decision-making and problem-solving. With continuous technological advancements, the application potential of clustering analysis in various fields will continue to be explored and expanded.

Traditional clustering refers to relatively simple and computationally efficient methods. These methods typically focus on surface-level features of the data rather than deeper, more complex patterns or structures. Major types of traditional clustering include: 1) centroid-based clustering, 2) connectivity-based clustering, 3) density-based clustering, 4) model-based clustering, and 5) grid-based clustering [1]. However, traditional clustering methods fail to effectively handle increasingly high-dimensional and unstructured data. Later, some researchers employed dimensionality reduction and sampling techniques to extract features from data before performing clustering [2][3], but these methods still struggle to capture deeper nonlinear relationships within the data and cannot effectively handle unstructured data such as text and images.

To address these challenges, deep clustering, which combines deep learning with traditional clustering methods, has emerged as a research hotspot. Deep clustering leverages the powerful feature extraction capabilities of deep neural networks (DNNs) [4] to obtain high-level representations of the data and perform clustering on these representations. By establishing mutual feedback between feature learning and clustering, deep clustering can better handle complex, nonlinear, high-dimensional data. Moreover, through joint optimization of feature representations and clustering processes, deep clustering adapts to various data distributions and structures, yielding more precise clustering results and stronger generalization capabilities.

In deep clustering, the design of the loss function is crucial. A well-designed loss function guides the deep neural network to learn distinctive and clustering features representations, thereby improving the clustering performance, accuracy, and stability. It also enhances the stability of the training process and the model's generalization ability. The loss function must effectively integrate feature representation learning and clustering processes to ensure mutual reinforcement between the two.

Although there is a growing body of literature on deep clustering, few reviews specifically focus on the loss functions used in deep clustering. This paper categorizes deep clustering loss functions based on their design goals and their role in enhancing the effectiveness of clustering models. These categories include reconstruction loss, generative adversarial loss, clustering loss, contrastive loss, and graph-based loss.

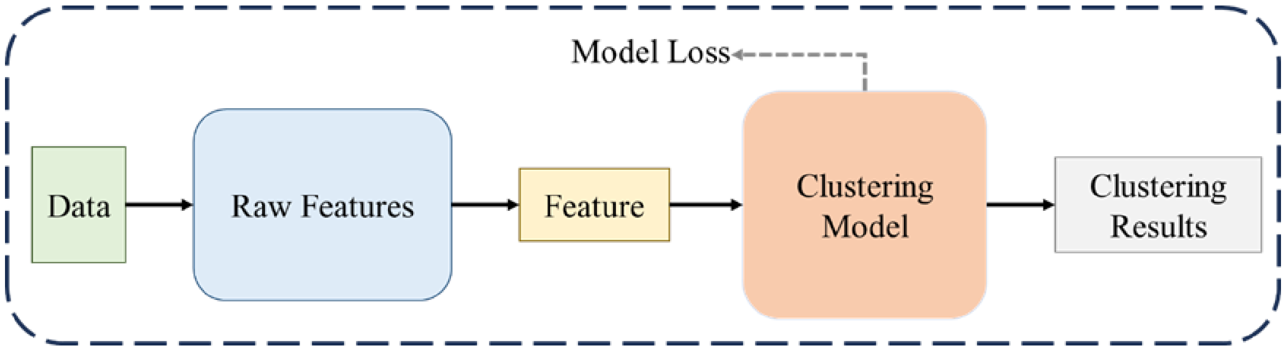

FIGURE 1. Traditional Clustering Process. The general process of traditional clustering. Data undergoes feature extraction, followed by loss calculation within the clustering model. The loss is then optimized to obtain the final clustering results

First, we briefly introduce traditional clustering methods and their associated loss functions. Next, we summarize the different forms of loss functions in deep clustering and discuss deep clustering algorithms from the perspective of lossfunctions. Additionally, some deep learning network models will be introduced to deepen the understanding of deep clustering algorithms. Finally, we conclude with an analysis of commonly used clustering metrics, datasets, and applications of deep clustering, highlighting four key elements that an ideal deep clustering loss function should possess.

II. LOSS FUNCTIONS IN TRADITIONAL CLUSTERING

In deep clustering methods, the design of loss functions often draws inspiration from the objectives of traditional clustering, integrating data representation learning with clustering structures. This integration ensures that deep feature learning and clustering performance reinforce each other. Understanding the loss functions in traditional clustering helps in grasping the design principles of deep clustering loss functions.

Traditional clustering algorithms are classical methods for handling data features. We will now introduce some common traditional clustering algorithms and their optimization objectives. The optimization goal of these algorithms is typically to minimize the loss of the clustering model, thereby achieving the best clustering performance. To control model complexity, prevent overfitting, and improve the model's generalization ability, the optimization objective often includes a regularization loss [5][6], as shown in Formula 1:

$$min\ L = L_{model} + \lambda L_{reg}, \lambda \geq 0 \tag{1}$$

The model's optimization goal is to minimize the loss L, where L_{model} represents the clustering model's loss, L_{reg} is the regularization term, whereλ is the regularization coefficient.

2.1 K-Means:

K-means is a widely used unsupervised clustering algorithm designed to partition a dataset into K clusters. The algorithm iteratively assigns data points to the nearest centroids and updates the centroid positions until convergence. With N data samples x and K initial cluster centroids μ, K-means aims to minimize the within-cluster variance, thereby ensuring an optimal division of the dataset into distinct clusters [7]. The optimization objective is as follows:

$$min\ L_{model} = \sum_{i=1}^{N} \sum_{k=1}^{K} I_{i,k} \|x_i - \mu_k\|_2^2 \quad (2)$$

$I_{i,k}$ indicates whether sample x_i belongs to cluster k. If x_i belongs to k, then $I_{i,k} = 1$; otherwise, $I_{i,k} = 0$. The within-cluster mean squared error in the optimization objective of K-Means can also be replaced with other metrics that measure data similarity or dissimilarity, such as the Pearson correlation coefficient, Mahalanobis distance, and so on. Table 1 presents some common similarity measures and their evaluations.

TABLE 1
METRICS USED FOR K-MEANS

Evaluation Metrics	Formula	Evaluation
Euclidean distance	$d(x,y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$	The most commonly used metric, easy to compute and understand. However, it is unsuitable for features with varying variances or non-spherical data.
Mahalanobis distance	$d(x,y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}$	Suitable for features with different variances or correlations, but has higher computational complexity.
Manhattan Distance	$d(x,y) = max_{i=1}^{N} \lvert x_i - y_i \rvert$	Focuses on the largest deviation and is highly sensitive to outliers, but may overlook smaller deviations in other dimensions.
Absolute value	$d(x,y) = \sum_{i=1}^{N} \lvert x_i - y_i \rvert$	Suitable for one-dimensional data.
Pearson Correlation Coefficient	$r(x,y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$	Computationally complex and measures only data correlation; not commonly used.

2.2 Density-Based Spatial Clustering of Applications with Noise:

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm that is particularly suited for discovering clusters of arbitrary shapes and handling datasets with noise. It defines clusters based on the concept of density and does not require the pre-specification of the number of clusters. DBSCAN only requires two parameters: the radius r and the minimum number of points ε within the radius to form a cluster [8].

The algorithm randomly selects an unvisited point. If the point is a core point (i.e., it has at least ε points within its radius r), the algorithm expands the cluster by including all points within the radius r and continues to grow the cluster. All core points within the radius, along with their neighboring points, are added to the cluster. This process is repeated until the cluster can no longer expand. The algorithm then proceeds to the next unvisited point, repeating the process until all points have been visited. Finally, points that do not belong to any cluster are labeled as noise. Figure 2 shows Process of DBSCAN Clustering.

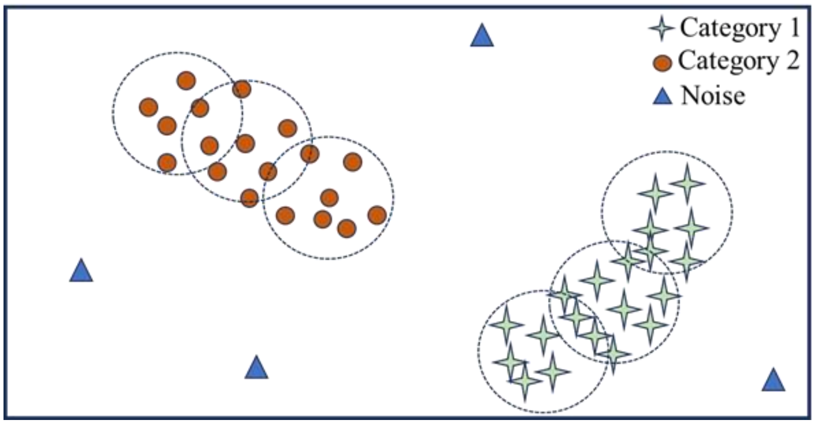

FIGURE 2: Magnetization as a function of applied field. Note that "Fig." is abbreviated. There is a period after the figure number, followed by two spaces. It is good practice to explain the significance of the figure in the caption

2.3 Spectral Clustering:

Spectral clustering is a graph-based clustering method that transforms the clustering problem into a graph partitioning problem. In spectral clustering, data points are treated as nodes in a graph, with the similarity between nodes represented as the weight of the edges. The fundamental idea is to use the eigenvectors of the graph's Laplacian matrix to find a low-dimensional representation of the data, and then perform traditional clustering in this low-dimensional space [9].

The similarity matrix A is constructed based on the distances between the samples, and the optimization goal of spectral clustering is to minimize the model's loss while solving for the spectral embedding features Z, as shown in Formula 3:

$$\min L_{model}(Z) = Tr(Z^T LZ) = \sum_{i,j} A_{i,j} \left\| z_i - z_j \right\|^2 \quad (3)$$
$$s.t. Z^T Z = I$$

Where z_i represents the i row of Z, corresponding to the spectral embedding features of the i sample x_i.

2.4 Subspace Clustering:

Subspace clustering [10] is a clustering method designed for high-dimensional data. It assumes that data from the same class are distributed in the same subspace, while data from different classes reside in different subspaces. Subspace clustering algorithms assume that each sample can be represented as a linear combination of other samples from the same class, a concept known as data self-expression. The optimization objective for subspace clustering is given by the following formula:

$$\min L_{model}(C) = \|C\|_p \quad (4)$$
$$s.t. X = CX, diag(C) = 0$$

Where X is the data matrix, where each row represents a sample, and C is the coefficient matrix that represents the combination of samples for self-expression.

2.5 Kullback-Leibler divergence:

KL divergence can be used to measure the difference between the distribution of data points within a cluster and the distribution of the cluster's center, or to assess the distributional differences between different clusters. KL divergence-based clustering utilizes KL divergence as a metric to evaluate the discrepancy between different probability distributions and groups data points into multiple clusters [11].

The probability that sample x_i belongs to the j class is computed using a Student's t-distribution, denoted as Q. The target distribution P is then defined based on Q.

$$q_{i,j} = \frac{\left(1 - \|z_i - \mu_j\|^2\right)^{-1}}{\sum_j \left(1 - \|z_i - \mu_j\|^2\right)^{-1}} \quad (5)$$

$$P_{i,j} = \frac{q_{i,j}^2 / \sum_i q_{i,j}}{\sum_j \left(q_{i,j}^2 / \sum_i q_{i,j}\right)}$$

The optimization objective for KL divergence is given by the following formula:

$$\boldsymbol{min\, L_{model} = KL(P\|Q)}$$
$$\boldsymbol{= \sum_i \sum_j P_{i,j} ln \frac{P_{i,j}}{q_{i,j}}} \quad (6)$$

2.6 Gaussian Mixture Model Clustering:

Gaussian Mixture Model (GMM) clustering is a density-based clustering method that assumes the data is generated from a mixture of several Gaussian distributions. Each Gaussian distribution represents a class in the data, and the entire dataset is a weighted sum of these Gaussian distributions. GMM clustering discovers the latent structure of the data by maximizing the likelihood function to estimate the parameters of the Gaussian distributions [12], as shown in Formula 7:

$$\max\ L_{model}(\pi, \mu, \Sigma) = -\sum_{i=1}^{N} ln\{\sum_{k=1}^{K} \pi_k N(z_i|\mu_k, \Sigma_k)\} \quad (7)$$

Where π_k represents the probability that a sample belongs to class K, while μ_k and Σ_k are the mean and covariance matrix of the kkk-th class, respectively.

2.7 Mutual Information Clustering:

Mutual Information (MI) is a method for measuring the amount of shared information between two random variables. If the two variables are independent, their mutual information is zero. Mutual Information Clustering is an information-theoretic clustering approach that uses the concept of mutual information to assess the interdependence between different data points or features, and performs clustering based on this measure [13].

The goal of the mutual information clustering algorithm is to optimize a conditional model $p(y|x;w)$, parameterized by w, that predicts the label distribution y_i as a sample x_i. The objective is achieved by maximizing the mutual information between the input variable X and the output variable Y. The mutual information between X and Y is given by the following formula:

$$\max\ L_{model}(\pi,\mu,\Sigma) = -\sum_{i=1}^{N} ln\{\sum_{k=1}^{K}\pi_k N(z_i|\mu_k,\Sigma_k)\} \tag{8}$$

After introducing the conditional model $\hat{p}(y,w) = \frac{1}{N}\sum_{i=1}^{N} p(y|x_i;w)$, the objective function for mutual information clustering is given by:

$$I_w(X,Y) = E_{\hat{p}(y,w)}(-\log\hat{p}(y,w))$$

$$-\frac{1}{N}\sum_{i=1}^{N} E_{p(y|x_i,w)}(-\log(y|x_i,w)) \tag{9}$$

Finally, the optimization objective for maximizing mutual information in clustering is given by the following formula:

$$\min L_{model} = -I_w(X,Y) + \lambda L_{reg} \tag{10}$$

H represents the entropy function, and $I(X,Y)$ denotes the mutual information between X and Y.

III. DEEP NEURAL NETWORKS IN DEEP CLUSTERING

Deep clustering involves learning the latent representations of data through neural networks, followed by clustering in the representation space. After defining the optimization objectives, it is crucial to select the appropriate neural network architecture to implement deep clustering [14][16]. In deep clustering tasks, neural networks are commonly used for feature extraction and data representation learning, allowing the loss function to more effectively reflect the relationships between samples. By optimizing the loss function, deep neural networks can efficiently assign samples to the correct clusters, thereby improving clustering performance. This section introduces different types of deep neural networks to provide readers with a comprehensive understanding of the deep clustering loss functions discussed later.

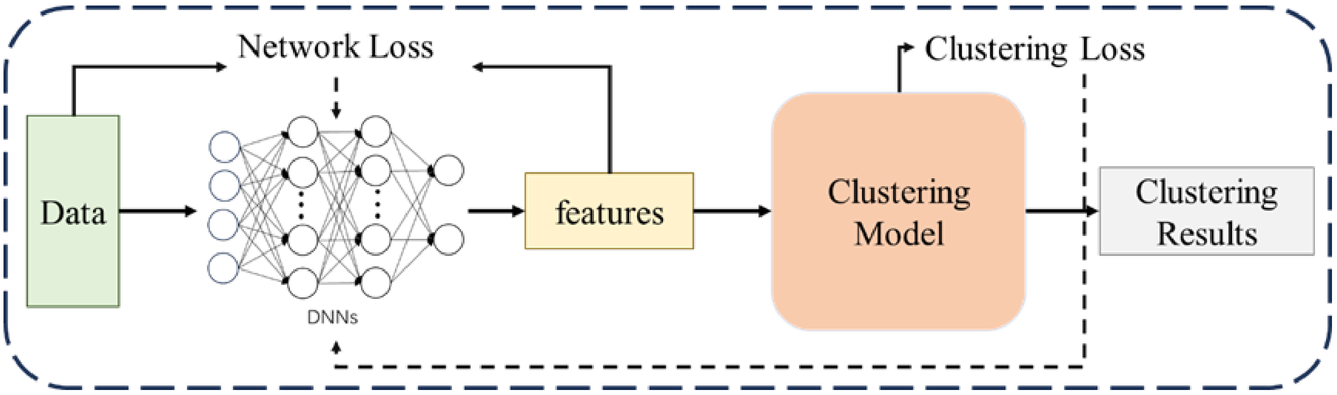

FIGURE 3. Deep Clustering Process. Unlike Figure 1, deep clustering extracts features through deep neural networks. In this process, the network loss and clustering model loss are fed back into the deep neural network to obtain improved data representations, thereby enhancing the clustering performance

3.1 Autoencoder:

An autoencoder typically consists of an encoder and a decoder. The encoder maps the input data to a lower-dimensional latent representation, while the decoder reconstructs the data from this latent representation, mapping it back to the original data space.

As shown in Figure 4, Z represents the embedding space. In deep clustering algorithms based on autoencoders, clustering is often performed in the embedding space, and hence Z is also referred to as the clustering layer.

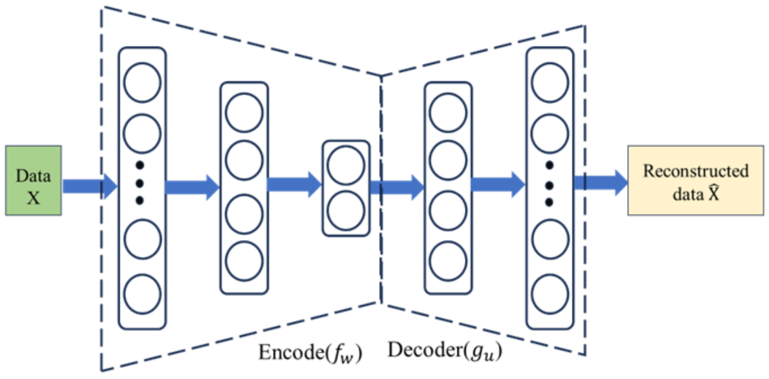

FIGURE 4: Autoencoder Structure

In 1986, Rumelhart et al. [17] proposed a structure similar to that of an autoencoder for unsupervised learning and feature extraction. While the model was not yet formally called an autoencoder, it laid the conceptual foundation for the later development of autoencoders. Subsequently, Hinton et al. [18] systematically introduced the concept of the autoencoder, which became a fundamental building block in deep learning and unsupervised learning. In 1986, Rumelhart et al. [17] proposed a structure similar to that of an autoencoder for unsupervised learning and feature extraction. While the model was not yet formally called an autoencoder, it laid the conceptual foundation for the later development of autoencoders. Subsequently, Hinton et al. [18] systematically introduced the concept of the autoencoder, which became a fundamental building block in deep learning and unsupervised learning.

3.2 Variational Autoencoder:

The Variational Autoencoder (VAE), shown in Figure 5, was proposed by Kingma [19] as an extension of the traditional autoencoder by integrating probabilistic models. In VAE, the latent representation is regarded as a probability distribution, enabling variational inference to learn the underlying data distribution, thereby enabling both data generation and reconstruction. Compared to traditional autoencoders, VAE excels in generative capabilities and can effectively capture implicit features of the data through its latent representation learning. Figure 5 depicts the structure of the variational autoencoder, where the data is mapped to the parameters of the latent space distribution, namely the mean μ and standard deviation σ.

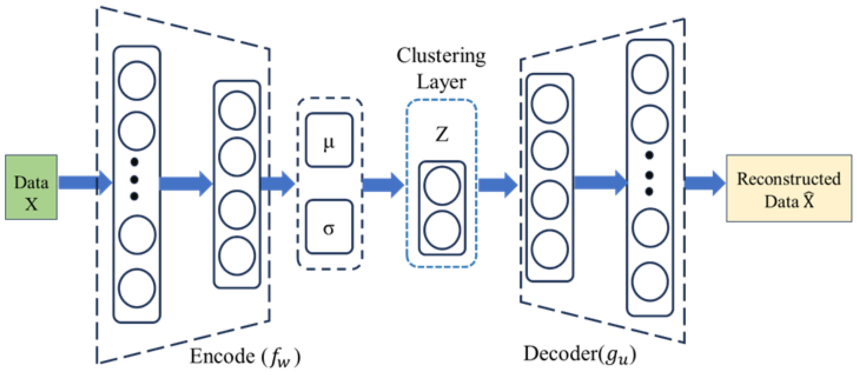

FIGURE 5: Variational Autoencoder

3.3 Generative Adversarial Networks:

Generative Adversarial Networks (GANs), introduced by Goodfellow et al. [20], consist of two components: a generator and a discriminator. The generator imitates real data by producing synthetic data, while the discriminator's task is to distinguish between real samples and the fake samples generated by the generator. It classifies the input data as either "real" or "generated," outputting a probability value of 0 or 1. Through this adversarial process, the generator and discriminator iteratively improve their respective capabilities, with the generator striving to create data that the discriminator can no longer effectively differentiate. This process continues until the generator produces data that the discriminator cannot distinguish, and the performance of the GAN is validated when the generated data becomes indistinguishable by the discriminator [39].

Figure 6 illustrates the structure of a GAN, where the generator G introduces noise and attempts to generate data that closely resembles the real data, while the discriminator D works to distinguish between the real and generated data.

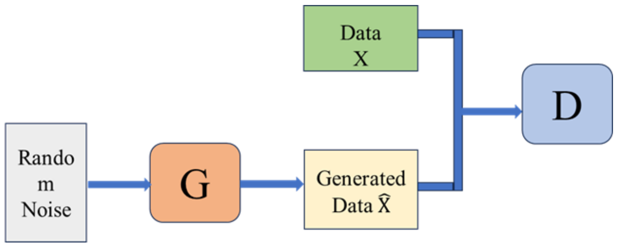

FIGURE 6: Generative Adversarial Network

3.4 Graph Neural Networks:

Graph Neural Networks (GNNs), first introduced by Scarselli et al. [21], as shown in Figure 7, are deep learning models designed to handle graph-structured data. In many practical scenarios, data is often represented in the form of a graph, such as in social networks, molecular structures, and transportation networks. GNNs are capable of processing these complex, non-Euclidean structures, enabling them to capture the relationships between nodes and their neighbors.

Figure 7 shows a Graph Neural Network (GNN), where the input consists of graph data, typically represented by a node feature matrix and an adjacency matrix that captures the relationships between nodes. The GNN aggregates neighborhood information through graph convolutions, learning representations for each node. Finally, the output is selected based on the task, such as node clustering or graph clustering.

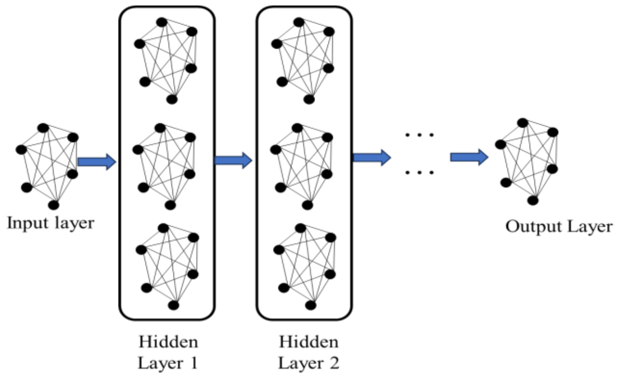

FIGURE 7. Graph Neural Network

3.5 Contrastive Learning Neural Networks:

Contrastive Learning Neural Networks, first introduced by Hadsell et al. [22], are a self-supervised learning method designed to learn data representations by comparing the similarities and differences between samples. In contrastive learning, the model does not rely on manually labeled data but instead trains the network by constructing positive and negative sample pairs, thereby learning discriminative and informative features.

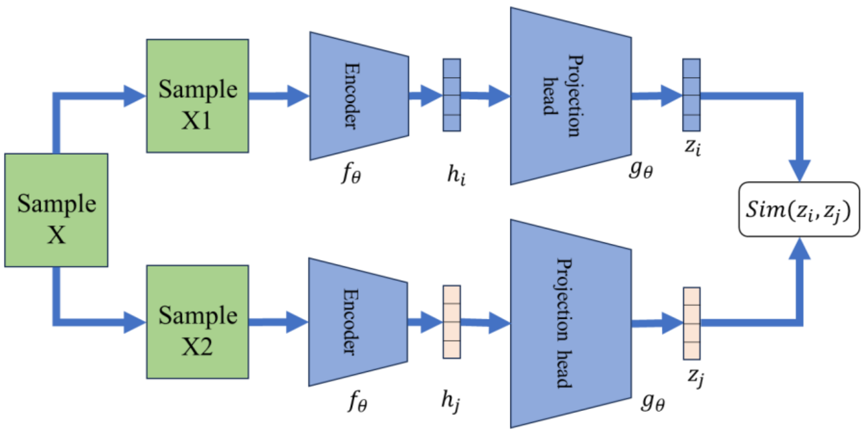

FIGURE 8. Contrastive Learning Neural Networks

3.6 Convolutional Neural Networks:

Convolutional Neural Networks (CNN), proposed by LeCun et al. [23], systematically demonstrate how convolutional layers effectively extract local features from images. A CNN typically consists of an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer. The input layer accepts a 3D tensor (H, W, C), where H is the height of the image, W is the width, and C represents the number of channels. The convolutional layers use filters that slide across the data to extract features, while the pooling layers perform down sampling on the feature maps to reduce computational complexity and the number of parameters, while preserving important feature information. After passing through convolutional and pooling layers, the network uses fully connected layers, similar to traditional neural networks, to output the data's features or class labels, with the output layer generating the final prediction.

In Figure 9, the CNN receives image data, where the convolutional layers extract local image features, the pooling layers reduce the dimensionality of the feature maps, and the fully connected layers integrate the features and map them to the output space.

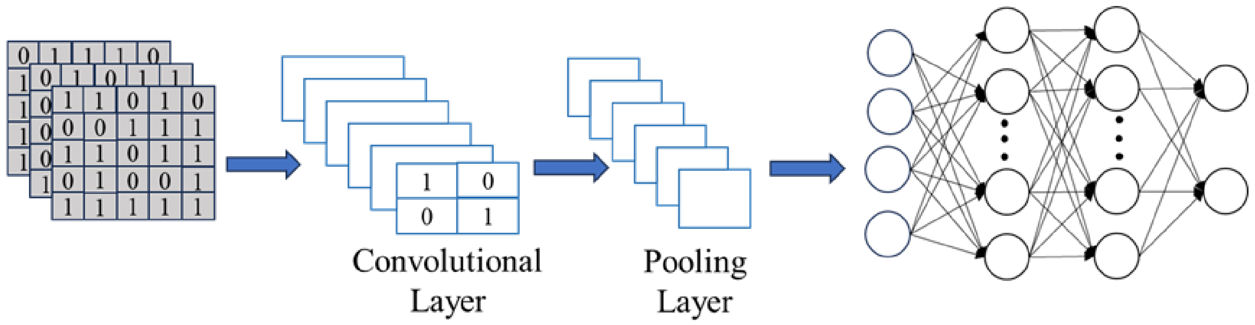

FIGURE 9: Convolutional Neural Network

The convolutional layer of CNN effectively extracts information from images, reducing computational complexity and the number of parameters, significantly improving the performance of computer vision tasks. It has also been successfully extended to various domains such as video analysis and natural language processing, driving the advancement of deep learning. When combined with other neural networks, such as autoencoders, CNN can effectively apply the strengths of different networks to image data clustering [44].

IV. LOSS FUNCTIONS

In deep clustering algorithms, the loss function is a core factor that determines model performance. It guides the model to learn high-quality feature representations by measuring the similarity and dissimilarity between samples in the embedding space. An effective loss function encourages the model to simultaneously minimize intra-cluster distances and maximize inter-cluster distances, resulting in clear clustering boundaries. Designing an appropriate loss function is crucial for improving the accuracy and stability of deep clustering. From the perspective of loss functions, deep clustering involves three components: network loss, clustering model loss, and regularization loss, as shown in Equation 11:

$$\min L = \alpha L_{net} + \beta L_{model} + \lambda L_{reg},$$
$$\alpha > 0, \beta, \lambda \geq 0 \tag{11}$$

Deep clustering algorithms can be classified into five categories based on the optimization objectives of their loss functions: (1) Reconstruction Loss focuses on reconstructing the data through autoencoders, ensuring that the embedding retains the original information. (2) Clustering Loss directly optimizes the clustering objective, ensuring that samples are grouped together in the embedding space. (3) Contrastive Loss strengthens the self-supervised representation by maximizing the similarity of similar samples. (4) Generative Adversarial Loss utilizes generative adversarial networks (GANs) to optimize the data distribution and enhance the clustering structure (5) Graph Structure Loss imposes graph Laplacian constraints on node relationships, ensuring that the embedding representation reflects the data's topological structure.

4.1 Reconstruction Loss:

Reconstruction Loss is primarily used in deep clustering algorithms based on autoencoders. It seeks to learn a compact, low-dimensional representation of the data, while simultaneously ensuring that the reconstructed output closely approximates the original input. This loss plays a crucial role in deep clustering algorithms, serving as one of the core components for optimizing data representations and clustering performance. It is typically combined with traditional clustering loss functions as introduced in Chapter 1. The expression for reconstruction loss is given by Equation 12:

$$L = \frac{1}{N}\sum_{i=1}^{N}\|x_i - \hat{x_i}\|^2 \tag{12}$$

Xie et al. [24] introduced Deep Embedded Clustering (DEC), which was the first to combine reconstruction loss with clustering loss, pioneering a deep learning-based embedded clustering framework. They employed an autoencoder to learn a low-dimensional representation of the data, while ensuring that the reconstruction preserved key features of the original input. In the embedding space, DEC progressively optimized the clustering loss, bringing the cluster centers closer to the data distribution. This approach laid the groundwork for subsequent research in deep clustering; however, it focused solely on global embedding learning, overlooking the retention of local similarities.

To address this limitation, Guo et al. [25] proposed Improved Deep Embedded Clustering with Local Structure Preservation (IDEC). Building upon DEC, IDEC incorporated a local structure-preserving mechanism by adding distance constraints between samples in the input space to the loss function. This modification ensured that neighboring samples remained similar in the low-dimensional representation, enabling the algorithm to capture both global features and better preserve the local structure of the data.

Further, Chen et al. [33] designed a deep clustering algorithm that incorporates manifold structure-preserving loss. Building upon reconstruction and clustering losses, they introduced manifold constraints that capture the complex, nonlinear structure of data, enabling the clustering process to better accommodate multi-manifold distributions. This method proves particularly effective in handling the manifold characteristics of high-dimensional data.

At the same time, Zhang et al. [34] proposed the Neural Collaborative Subspace Clustering (NCSC), an innovative approach that combines neural networks with subspace clustering. NCSC utilizes self-expression loss and reconstruction loss to ensure that the low-dimensional representations can be self-expressed through linear combinations of other samples, thus uncovering the underlying subspace structure. Additionally, the algorithm applies sparse regularization to mitigate noise interference and introduces a collaborative mechanism that enhances clustering robustness. In contrast, Zhou et al. [35] focused on preserving more of the original data information in the latent space. They proposed a latent distribution-preserving algorithm that strengthens the robustness of the latent representations, further improving clustering performance.

The application of reconstruction loss is not limited to spatial embedding and subspace learning. Fard et al. [36] combined reconstruction loss with K-means clustering loss to simultaneously optimize data representations and clustering results in a low-dimensional space. This approach overcomes the limitations of traditional K-means in the feature space, enabling better adaptation to complex datasets. Ren et al. [37] integrated autoencoders with density estimation, creating a new deep clustering model. By jointly optimizing reconstruction loss, density estimation loss, and clustering loss, they significantly enhanced clustering quality, particularly for datasets with high-density sample distributions.

Moreover, the clustering of multi-view data has also garnered attention. Yin et al. [38] proposed a novel multi-view clustering method that designs a mechanism for sharing generative latent representations. By minimizing the distance between representations from different views, this approach effectively integrates multi-view information. It demonstrates particularly strong performance in handling complex multimodal data.

4.2 Clustering Loss:

Deep clustering algorithms typically integrate the clustering objective directly into the loss function, constraining the embedding learning process of deep neural networks through specific clustering losses. These clustering losses include traditional K-means loss, KL divergence loss, and others, and are widely applied in deep clustering tasks.

The Variational Deep Embedding (VaDE) proposed by Jiang et al. [42] combines variational inference with deep learning, offering an innovative solution for deep clustering. VaDE not only optimizes the representation of the latent space using reconstruction loss but also ensures that the latent distribution of the data is close to a Gaussian distribution by optimizing the KL divergence loss. Subsequently, a Gaussian Mixture Model (GMM) is introduced in the latent space to achieve clustering, significantly enhancing the model’s adaptability to complex data.

Yang et al. [26] designed a deep clustering algorithm that generates representations suitable for K-means clustering. In addition to combining reconstruction loss and K-means clustering loss, this algorithm introduces a K-means clustering-friendly loss that aims to minimize the intra-class distances while maximizing the distances between different class centers, thus significantly improving the performance of K-means clustering.

Similarly, Yang et al. [15] proposed a method that improves K-means clustering by leveraging similarity loss. By reducing the distance between similar samples in the embedding space, they effectively capture the data structure while optimizing clustering performance.

Chang et al. [27] proposed a deep clustering method called Deep Adaptive Image Clustering (DAC) specifically for image data. Unlike traditional methods based on cluster centers, DAC uses convolutional neural networks to extract image features and generates pseudo-labels by calculating the cosine similarity between sample pairs. Then, DAC performs clustering using the binary cross-entropy loss of the pseudo-labels. Additionally, its adaptive mechanism provides new directions and insights for deep clustering algorithms.

4.3 Clustering Loss:

Contrastive loss is extensively utilized in deep clustering algorithms based on contrastive learning, where the objective is to maximize the similarity between samples of the same class while minimizing the similarity between samples of different classes. Typically, this loss function is integrated with self-supervised learning to enhance the discriminative capacity of the embedding space, as outlined in Equation 13. The fundamental principle of contrastive loss is to optimize the relative distances between samples, thereby fostering the aggregation of similar samples and increasing the separation between samples of distinct classes, ultimately improving clustering performance.

$$L = \frac{1}{2N}\sum_{i=1}^{N}(y_i \cdot d_i^2 + (1 - y_i) \cdot \max(0, m - d_i)^2) \tag{13}$$

In this context, y_i represents the label of the i sample pair, d_i denotes the Euclidean distance between the sample pair in the embedding space, and m is the threshold set to determine when samples are considered dissimilar.

For instance, Li et al. [28] proposed a method combining contrastive learning and traditional clustering objectives, aiming to enhance clustering quality by maximizing the similarity within the same class and the dissimilarity between different classes. Their algorithm simultaneously optimizes both contrastive loss and clustering loss, achieving more accurate clustering results. Similarly, Zhang et al. [47] introduced reconstruction loss and clustering loss to learn the data distribution in the latent space while incorporating contrastive loss and subspace constraint loss, further strengthening the model’s capability in extracting local features, thereby improving clustering performance.

Zhao et al. [48] proposed an image clustering algorithm by introducing category style loss to optimize the style similarity between samples. When combined with contrastive loss, this approach ensures that samples from the same class are more compact in the latent space. Unlike traditional methods, this method not only optimizes the distances between samples but also addresses the style differences between classes, significantly improving the image clustering performance.

Additionally, Li et al. [50] introduced a contrastive clustering method that enhances the discriminative power between samples using contrastive loss and combines it with clustering loss to achieve final clustering. This method is simple in structure, easy to extend, and applicable to a wide range of datasets. Yan et al. [51] proposed an image clustering method by combining autoencoders with probabilistic triplet loss. In addition to learning the low-dimensional representations of data through reconstruction loss, this approach incorporates probabilistic loss to optimize the latent space distance relationships between samples, further enhancing clustering accuracy.

Through these innovations, deep clustering algorithms based on contrastive loss not only improve clustering accuracy but also provide greater adaptability and extensibility for handling a wide variety of data types.

4.4 Generative Adversarial Loss:

Generative Adversarial loss (GAN loss) is widely used in deep clustering algorithms based on Generative Adversarial Networks (GANs). It consists of two components: the generator loss and the discriminator loss. The generator's objective is to generate samples, while the discriminator’s task is to distinguish whether the samples are real or generated. By combining generative adversarial loss with clustering loss and reconstruction loss, GANs play a crucial role in optimizing the generative and clustering structures within the embedding space. The specific formulation is presented in Equation (14).

$$\min_G \max_D L(D, G) = E_{X \sim P_X}[\log D(X)] + E_{\hat{X} \sim P_{\hat{X}}}[\log(1 - D(G(\hat{X})))] \tag{14}$$

In this context, $D(x)$ represents the discriminator, $G(\hat{X})$ the generator, P_X the true data distribution, and $P_{\hat{X}}$ the distribution in the latent space.

For instance, Dumoulin et al. [29] proposed an unsupervised learning approach that leverages the adversarial interaction between the generator and an inference model to learn the joint distribution of data and latent variables, thus inferring the latent representations of the data. While this method is not directly applied to clustering tasks, it offers a robust framework for latent space learning that benefits subsequent clustering applications.

Mukherjee et al. [30] introduced an innovative approach to enhance the performance of GANs in deep clustering by incorporating a clustering mechanism. This method combines adversarial loss, generator reconstruction loss, and latent space clustering loss. Through adversarial training, it enhances the diversity and representational capacity of the clustering structure, resulting in significant improvements in clustering quality.

Ghasedi et al. [40] proposed a novel deep clustering algorithm by combining self-paced learning with GANs. In this approach, in addition to the adversarial and clustering losses, a self-paced loss is introduced that dynamically adjusts the loss weights, allowing the model to better handle complex samples and improve clustering performance.

In multi-view clustering, Xu et al. [41] developed a GAN-based method that accounts for data incompleteness. Building upon traditional adversarial loss, reconstruction loss, and clustering loss, they introduced a view incompleteness loss, ensuring that even with missing data in some views, the algorithm can still perform clustering effectively, thereby enhancing the robustness of the model.

These innovations highlight that generative adversarial loss not only contributes to generating high-quality latent representations but, when integrated with clustering mechanisms, can significantly optimize clustering structures, thereby improving both the effectiveness and adaptability of deep clustering algorithms.

4.5 Graph Structure Loss:

Graph structure loss plays a crucial role in deep clustering algorithms such as Graph Convolutional Networks (GCNs) and Graph Autoencoders. By constructing the adjacency matrix or Laplacian matrix of a graph, graph structure loss ensures that nodes with similar structures are positioned closer together in the embedding space, thereby facilitating the successful completion of clustering tasks.

$$L = \frac{1}{2}\sum_{i,j} A_{ij}\left\|h_i - h_j\right\|^2 \tag{15}$$

Formula (15) represents Laplacian regularization, which constrains the embedding features of adjacent nodes to be as similar as possible through the Laplacian matrix. In this formula, A_{ij} is the edge weight between nodes i and j in the adjacency matrix A, while h_i and h_j are the embeddings of nodes i and j in the latent space.

The pioneering work of Kipf and Welling [32], namely Graph Convolutional Networks (GCNs), introduced a novel approach to feature learning on graph-structured data. While GCNs themselves were not directly applied to clustering tasks, they laid the groundwork for the widespread use of graph neural networks in deep clustering, profoundly influencing subsequent research directions.

Building on this foundation, Bo et al. [31] proposed the Structural Deep Clustering Network (SDCN), which cleverly integrates graph structure information with deep embedding techniques to enhance clustering performance. SDCN incorporates reconstruction loss from autoencoders, K-means clustering loss in low-dimensional space, and graph embedding loss, combining the strengths of both graph neural networks and deep learning. This significantly improved clustering outcomes and demonstrated the potential of graph structures in deep clustering.

Shaham et al. [45] took a different approach, combining spectral clustering with deep learning. They used isomorphic reconstruction loss and spectral loss to optimize the neural network and learn the spectral information of the data. The clustering loss further refined the clustering structure in the latent space. This method combines the theoretical strengths of spectral clustering with the powerful expressive capacity of deep learning, significantly improving clustering accuracy.

In a different direction, Yang et al. [46] optimized the distribution of the latent space using a variational autoencoder to approximate a Gaussian distribution. They then incorporated graph structure loss to ensure that similar samples in the graph remain similar in the latent space. This not only strengthened the representation of the data's underlying structure but also contributed to improved clustering results.

Deng et al. [49] introduced a deep clustering method that combines dual autoencoders with spectral clustering. In this approach, one autoencoder is optimized through reconstruction loss, while the other uses spectral loss to enhance clustering performance. This strategy effectively strengthened the similarity between samples and improved clustering accuracy.

Lastly, Huang et al. [52] proposed a multi-view deep clustering method that integrates the strengths of deep learning and spectral clustering. They introduced reconstruction loss and KL divergence loss to ensure that the data distribution in the latent space approximates a Gaussian distribution. Additionally, the method incorporates multi-view loss and inter-class loss to ensure consistency across different views and maintain sufficient separation between different classes, further improving clustering performance.

V. DATASETS AND EVALUATION METRICS

Evaluation metrics provide quantitative feedback for the loss functions in deep clustering algorithms, helping assess their actual contribution to clustering quality. By analyzing the results of evaluation metrics, the loss function design can be optimized to better align with task requirements and data distribution characteristics. Selecting appropriate metrics fosters Objective comparisons and improvements across algorithms, offering guidance for designing more efficient loss functions.

5.1 Graph Structure Loss:

Evaluation metrics are standards used to measure and assess model performance. In the field of deep clustering, various metrics are available to evaluate the effectiveness of deep clustering algorithms. This section categorizes these metrics based on their characteristics, providing a more comprehensive understanding of their use.

5.2 Accuracy:

Accuracy (ACC) is a commonly used metric in deep clustering algorithms. Unlike the traditional accuracy that simply matches the predicted labels with the true labels, deep clustering accuracy finds the optimal label mapping that maximizes the number of data points whose cluster labels align with the true labels. The expression for accuracy is shown in Formula (16):

$$ACC = max_{\pi} \frac{1}{N} \sum_{i=1}^{n} 1\{l_i = \pi(y_i)\}, \tag{16}$$

where l_i and y_i denote the true label and the clustering label of data point i, respectively, π is the function that maps the clustering labels to the true labels, and $1\{\cdot\}$ is the indicator function, which takes the value of 1 when the condition inside the parentheses is true, and 0 otherwise."

5.3 Normalized Mutual Information:

The Normalized Mutual Information (NMI) metric evaluates the quality of clustering results by measuring the amount of mutual information between the clustering results and the true labels, and normalizing it. The range of NMI is between 0 and 1, where a higher value indicates better clustering performance. The expression for NMI is shown in Equation (17):

$$NMI(y, l) = \frac{2 \cdot I(y, l)}{H(y) + H(l)} \tag{17}$$

Where y and l represent the clustering labels and true labels of the data.

5.4 Silhouette Coefficient:

The Silhouette Coefficient measures clustering quality by comparing the distance between a data point and other points within the same cluster, as well as the distance to the nearest point in a different cluster. The range of the Silhouette Coefficient is from -1 to 1, where higher values indicate better clustering performance. The expression for the Silhouette Coefficient is given by Formula (18):

$$SS(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{18}$$

where $a(i)$ is the average distance between sample i and other samples within the same cluster, and $b(i)$ is the average distance between sample i and the nearest sample from a different cluster.

5.5 Root Mean Square Error:

Root Mean Square Error (RMSE) measures the clustering performance by calculating the root mean square of the Euclidean distances from data points to the centroids of their assigned clusters. A smaller value indicates better clustering performance. The expression for RMSE is given by Formula (19):

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{n} (d(x_i, c_i))^2} \tag{19}$$

Where $d(x_i, c_i)$ denotes the Euclidean distance between data point x_i and the centroid c_i of the cluster to which x_i belongs.

5.6 Neighbor Consistency:

Neighbor Consistency (NC) is used to measure the degree of discrepancy between different domains or subgroups in clustering results. A lower value indicates that the clustering results are more consistent across different domains. The expression for Neighbor Consistency is shown in Formula (20):

$$NC = \frac{1}{N}\sum_{i=1}^{n}\frac{\left|\{j\lceil j \in N_k(i)\ and\ l_j = l_i\}\right|}{|N_k(i)\ |} \tag{20}$$

where $N_k(i)$ represents the k-nearest neighbors of sample i, and l_i is the true label of sample i.

5.7 Significance:

Significance (SIG) measures clustering quality by calculating the ratio of the minimum inter-cluster distance to the maximum intra-cluster radius. A larger ratio indicates that the samples within the same cluster are compactly grouped, while different clusters are well-separated. The expression for significance is shown in Formula (21):

$$SIG = \frac{\min(R_j)}{\max(r_i)}$$

$$j = 1,2,\dots,\frac{k(k-1)}{2} \tag{21}$$

Where R_j is the inter-cluster distance, and r_i is the intra-cluster distance.

The aforementioned common evaluation metrics for deep clustering algorithms are effective in assessing the performance of deep clustering models. It is important to note that when selecting evaluation metrics, the choice should be based on factors such as whether the data has labels, the specific use case, and other context-specific considerations, in order to objectively evaluate the algorithm's performance. Additionally, it may be beneficial to use multiple metrics to assess the deep clustering algorithm from different perspectives, enabling a more comprehensive evaluation. Table 2 presents some deep clustering evaluation metrics, their applicable scenarios, as well as their advantages and disadvantages.

TABLE 2
EVALUATION METRICS FOR DEEP CLUSTERING ALGORITHMS

Evaluation Indicators	Data Labels	Supervised Learning	Unsupervised Learning	Advantages	Disadvantages
ACC	√	√	√	ACC is easy to understand and calculate, and can directly reflect the accuracy of clustering.	ACC requires true labels for data and is sensitive to the number and distribution of categories.
NMI	√	√	√	NMI can compare the clustering results of different data sets and is insensitive to the number of categories.	NMI requires true labels and is sensitive to noise and outliers
ARI	√	√	√	ARI is not sensitive to changes in the number of categories	ARI requires data labels and is sensitive to noise
NC	√	√	√	NC measures the consistency between samples and neighborhood samples and is used to evaluate local structure.	NC requires true labels and is sensitive to noise
SS	-	-	√	NC requires true labels and is sensitive to noise	SS has high computational complexity and is insensitive to clusters of different shapes
RMSE	-	-	√	RMSE reflects the accuracy of the cluster center and is easy to understand and calculate.	RMSE is computationally expensive
SIG	-	-	√	SIG is easy to understand and calculate, and intuitively reflects the clustering effect of the algorithm.	SIG only considers extreme cases and cannot reflect the general effect of clustering.

"√" indicates that the evaluation metric requires the use of data labels or is suitable for the specific context, while "-" indicates that the evaluation metric does not require data labels or is not suitable for the context.

VI. DATASETS

The characteristics of the dataset directly influence the design of the loss function. For example, the distribution of the data, the number of categories, and the similarity between samples determine the patterns that the loss function needs to capture. Different datasets may require specific loss functions to adapt to their characteristics, ensuring high clustering performance. By studying the structure and attributes of the dataset, more generalizable and targeted loss functions can be designed to improve the effectiveness of deep clustering algorithms.

The datasets commonly used in deep clustering algorithms generally fall into the following eight categories: Image Datasets, Text Datasets, Time Series Datasets, Gene Expression Datasets, Social Network Datasets, Audio Datasets, Structured Datasets, Graph Datasets.

6.1 Image Datasets:

- **Modified National Institute of Standards and Technology database (MNIST):** The MNIST dataset consists of 60,000 training images and 10,000 testing images of handwritten digits, covering the range of d*igits from 0 to 9, with a total of 10 categories.*
- **CIFAR-10:** The CIFAR-10 dataset [57] contains 60,000 32×32 color images, categorized into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class has 6,000 images.
- **CIFAR-100:** The CIFAR-100 dataset [58] is an extension of CIFAR-10, containing 100 categories, with 600 images per category, totaling 60,000 32×32 color images.
- **Fashion-MNIST:** The Fashion-MNIST dataset [59] is a replacement for the MNIST dataset, with the same size, format, and training/testing split as MNIST. It contains 10 categories of fashion items, including t-shirts, pants, and jackets, presented in front-view images.

6.2 Text Datasets:

- **20 Newsgroups:** The 20 Newsgroups dataset [60] contains approximately 20,000 news group documents, which are divided into 20 categories representing different topics such as sports, music, and politics.
- **Reuters-21578:** The Reuters-21578 dataset [61] includes 21,578 news articles categorized into different topics, widely used for text classification and clustering tasks.

6.3 Time Series Datasets

- **UCR Time Series Classification Archive:** The UCR Time Series Classification Archive dataset [62] provides 128 time series datasets, each with sample data and labels. Covering various fields, it is an important open-source resource in time series mining.
- **ECG Data:** The ECG Data dataset [63] contains electrocardiogram (ECG) data, with labeled samples, used for clustering tasks related to heart activity in time series data.

6.4 Gene Expression Datasets

- **Single-cell RNA Sequencing (scRNA-seq):** The Single-cell RNA sequencing (scRNA-seq) dataset is an unlabeled dataset used for analyzing single-cell RNA sequencing data to identify different cell types.
- **Cancer Gene Expression Profiles:** The Cancer Gene Expression Profiles dataset [64] contains gene expression data from different types of cancer, used for clustering tasks in cancer research.

6.5 Social Network Datasets

- **Facebook:** The Facebook dataset is an unlabeled social network dataset that contains user relationship information, used for community detection and user grouping clustering tasks.

- **Enron Email Dataset:** The Enron Email Dataset [65] is an unlabeled network dataset that contains email communication data between employees of the Enron corporation.

6.6 Audio Datasets

- **Speech Commands Dataset:** The Speech Commands Dataset [66] is a labeled dataset that contains audio data for several different voice commands, used for speech recognition and clustering tasks.
- **TIMIT:** The TIMIT dataset [67] is a standard dataset in the field of speech recognition, containing audio recordings and corresponding textual labels.

6.7 Structured Datasets

- **Iris Dataset:** The Iris dataset [68] contains 150 samples, each with 4 features, categorized into 3 different classes.
- **Wine Dataset:** The Wine dataset [69] includes data on 3 different varieties of wine, with 13 chemical characteristics for each sample, totaling 178 samples.

6.8 Graph Datasets

- **CORA:** The CORA dataset is a labeled dataset that contains citation network data of scientific papers, suitable for classification and clustering tasks in graph data.
- **PubMed:** The PubMed dataset [70] is a labeled, large-scale biomedical literature database containing citation relationships between papers, used for graph clustering and community detection.

The selection of these datasets typically depends on the specific application scenarios and the requirements of the clustering tasks. Deep clustering algorithms often incorporate loss functions with specific functionalities to extract meaningful features from complex data, thereby achieving more accurate clustering results.

VII. SUMMARY AND ANALYSIS

Based on the characteristics of the loss functions used in most existing deep clustering algorithms, this paper categorizes deep clustering loss functions into five general types: reconstruction loss, clustering loss, contrastive loss, adversarial loss, and graph structure loss.

Among these, reconstruction loss is a commonly used loss function in deep clustering and plays an important role: 1) By reconstructing the input data back into its original form, the model can learn the latent structure of the data. 2) During the reconstruction process, all input features are considered, and reconstruction loss effectively handles high-dimensional data. 3) The reconstruction process helps smooth the data distribution, improving the model's robustness.

However, reconstruction loss often requires complex neural network architectures, and deep clustering algorithms using this loss function demand longer training times and higher computational costs. Additionally, it is prone to overfitting. Although reconstruction loss can restrict the latent representation within a certain range, this representation often lacks practical meaning. Moreover, after training, the algorithm still needs to reconstruct the data, leading to wasted computational resources.

Clustering loss typically refers to a loss function that directly incorporates the clustering objective in deep clustering algorithms. During the deep embedding learning process, the clustering loss constrains the deep neural network. The advantages of clustering loss are as follows: 1) It focuses more on improving clustering performance compared to other losses. 2) It reduces the decoupling between feature learning and clustering, leading to a latent space that is more aligned with clustering requirements. 3) The model design is simple and efficient, reducing training time and saving computational resources.

However, clustering loss also has some drawbacks: 1) The focus on clustering may lead the model to neglect learning meaningful data features. 2) For more complex data structures, simple clustering loss may fail to capture data features effectively, resulting in poor clustering performance. 3) In unsupervised learning, directly optimizing the clustering objective can sometimes lead to unstable convergence behavior.

Contrastive loss is simple to implement and, by pulling similar samples together and pushing dissimilar samples apart, can effectively learn useful feature representations, enhancing the model's understanding of the relationships between samples. One of the main advantages of contrastive loss is its strong generalization ability. Through optimization of similarity and dissimilarity, it generalizes well to new data. However, there are some issues in training the model: 1) The need to compute the similarity between every pair of samples leads to high computational costs. 2) Contrastive loss relies on the selection of negative samples, and if the negative samples are not representative, it results in poor model performance. 3) Focusing on local relationships between samples might cause the model to fail to capture global features effectively. These are key areas of focus when using contrastive loss.

Adversarial loss is becoming increasingly widespread in deep clustering algorithms. By continuously training with adversarial generation, it can enhance feature learning and effectively learn the true distribution of the data. Additionally, adversarial models have strong flexibility and can be combined with convolutional neural networks, recurrent neural networks, and other architectures to adapt to different data types. However, adversarial loss comes with some disadvantages: 1) Training instability, 2) Difficulty in tuning hyperparameters (such as learning rate, batch size, etc.), 3) High computational costs, 4) Lack of clear evaluation criteria, which require careful handling in practical applications. Properly designing the training process and optimization strategies can improve the performance and stability of adversarial networks.

Graph structure loss is primarily used in graph neural networks and other graph-related tasks, emphasizing the relationships and structural information between nodes. Graph structure loss effectively utilizes graph topological information, aiding in the understanding of node relationships. It can be applied to both directed and undirected graphs and is highly flexible and adaptable. Through graph structure loss, the model can better aggregate information between nodes, improving clustering quality. For sparse data scenarios such as user behavior data and social networks, graph structure loss can help the model extract useful features. However, graph structure loss also has notable drawbacks: 1) High computational complexity due to the need to consider numerous edges and nodes, 2) Unstable model performance.

Since graph neural networks rely on local neighbor information during training, the model may fall into local optima, resulting in suboptimal performance. Currently, more and more deep clustering algorithms take these issues into account, designing different loss functions in deep clustering to improve model performance. For example, reconstruction loss can be combined with K-means-friendly clustering loss, encouraging autoencoders to learn latent spaces that are more compatible with K-means clustering. Alternatively, multiple loss functions can be combined based on the specific requirements of the clustering task to enhance clustering performance.

VIII. KEY ELEMENTS OF AN EXCELLENT LOSS FUNCTION AND FUTURE DIRECTIONS

In this paper, we analyze and summarize existing deep clustering algorithms from the perspective of loss functions. We propose that a good deep clustering model should have a loss function that possesses four essential elements: information retention, balance, robustness, and scalability. Furthermore, we also identify two future research directions for deep clustering algorithms: deep clustering assumptions and deep representations based on KAN.

8.1 A. Key Elements of an Excellent Loss Function:

8.1.1 Information Retention:

In deep clustering, the loss function should have strong information retention capabilities to ensure that the model can effectively capture and express the key information in the data. This means that the loss function should not only quantify the discrepancy between the predicted and true labels but also retain as much of the input data's features and structure as possible. In high-dimensional spaces, the ability to retain information directly affects the accuracy of clustering results and the model's generalization ability. By optimizing the loss function to maximize information retention during training, the model can better adapt to complex data, thereby improving clustering performance and effectiveness.

8.1.2 Balance:

To enhance clustering performance, deep clustering algorithms often include multiple components in their loss functions, especially reconstruction loss and clustering loss. An excellent loss function needs to achieve a reasonable balance between these components, ensuring that the embedding space retains the original data's information while also improving clustering performance. How to balance the various parts of the loss function will be an ongoing research topic for deep clustering algorithms.

8.1.3 Robustness:

Data often contains outliers and noise. An excellent deep clustering loss function should exhibit a certain level of robustness, ensuring that clustering is not unduly affected by these factors. Information-theoretic loss functions, such as maximum mutual information methods, can reduce the impact of noise by increasing intra-cluster compactness and inter-cluster separation.

8.1.4 Scalability:

Some loss functions, such as contrastive loss, can be trained in batches and optimized on small batches of data to enable the algorithm's application to large-scale datasets. As the dataset size increases, the loss function should be capable of efficiently handling large-scale data without leading to excessive computational costs. An excellent loss function should have this ability as well.

8.2 Future Directions:

8.2.1 Deep Clustering Assumptions:

Currently, many deep clustering algorithms rely on traditional clustering assumptions in the embedding space, such as subspace clustering assumptions, clustering density assumptions, and information entropy-based assumptions. These deep clustering algorithms often inherit the limitations of these assumptions. For example, the clustering center-based assumption may lead to problems such as local optima. Therefore, deep clustering assumptions based on the perspective of deep learning will play a significant role in the future development of deep clustering techniques. New assumptions that better align with the capabilities of deep learning models can potentially improve the robustness and accuracy of clustering results.

8.2.2 Deep Representations Based on KAN:

Unlike others neural networks based on approximate expression theorems, Liu et al. [54] (2024) proposed neural networks based on the Kolmogorov-Arnold representation theorem (KAN). Liu claims that KAN has advantages over current neural networks, such as higher data transmission rates, stronger cross-modal integration capabilities, enhanced robustness, and improved model interpretability. While existing deep clustering algorithms typically use neural networks to learn representations, the introduction of KAN enriches the choices for deep clustering representations. Combining KAN with deep clustering may lead to the extraction of superior latent representations, thereby improving clustering performance.

REFERENCES

[1] Xu R, Wunsch D. Survey of clustering algorithms[J]. IEEE Transactions on neural networks, 2005, 16(3): 645-678.
[2] Bahmani B, Moseley B, Vattani A, et al. Scalable k-means++[J]. arXiv preprint arXiv:1203.6402, 2012.
[3] Radovanovic M, Nanopoulos A, Ivanovic M. Hubs in space: Popular nearest neighbors in high-dimensional data[J]. Journal of Machine Learning Research, 2010, 11(sept): 2487-2531.
[4] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(8): 1798-1828.
[5] Tibshirani R. Regression shrinkage and selection via the lasso[J]. Journal of the Royal Statistical Society Series B: Statistical Methodology, 1996, 58(1): 267-288.
[6] Hoerl A E, Kennard R W. Ridge regression: Biased estimation for nonorthogonal problems[J]. Technometrics, 1970, 12(1): 55-67.
[7] MacQueen J. Some methods for classification and analysis of multivariate observations[C]//Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1967, 1(14): 281-297.
[8] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//kdd. 1996, 96(34): 226-231.
[9] Ng A, Jordan M, Weiss Y. On spectral clustering: Analysis and an algorithm[J]. Advances in neural information processing systems, 2001, 14.
[10] Agrawal R, Gehrke J, Gunopulos D, et al. Automatic subspace clustering of high dimensional data for data mining applications[C]//Proceedings of the 1998 ACM SIGMOD international conference on Management of data. 1998: 94-105.
[11] Banerjee A, Dhillon I S, Ghosh J, et al. Clustering on the Unit Hypersphere using von Mises-Fisher Distributions[J]. Journal of Machine Learning Research, 2005, 6(9).
[12] McLachlan G J, Basford K E. Mixture models: Inference and applications to clustering[M]. New York: M. Dekker, 1988.
[13] Fred A L N, Jain A K. Data clustering using evidence accumulation[C]//2002 International Conference on Pattern Recognition. IEEE, 2002, 4: 276-280.
[14] Hinton G E, Zemel R. Autoencoders, minimum description length and Helmholtz free energy[J]. Advances in neural information processing systems, 1993, 6.
[15] Yang J, Parikh D, Batra D. Joint unsupervised learning of deep representations and image clusters[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 5147-5156.
[16] Bengio Y, Lamblin P, Popovici D, et al. Greedy layer-wise training of deep networks[J]. Advances in neural information processing systems, 2006, 19.
[17] Rumelhart D E, Hinton G E, Williams R J. Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986[J]. Biometrika, 1986, 71(599-607): 6.
[18] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. science, 2006, 313(5786): 504-507.

[19] Kingma D P. Auto-encoding variational bayes[J]. arXiv preprint arXiv:1312.6114, 2013.

[20] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[J]. Advances in neural information processing systems, 2014, 27.

[21] Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model[J]. IEEE transactions on neural networks, 2008, 20(1): 61-80.

[22] Hadsell R, Chopra S, LeCun Y. Dimensionality reduction by learning an invariant mapping[C]//2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06). IEEE, 2006, 2: 1735-1742.

[23] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[24] Xie J, Girshick R, Farhadi A. Unsupervised deep embedding for clustering analysis[C]//International conference on machine learning. PMLR, 2016: 478-487.

[25] Guo X, Gao L, Liu X, et al. Improved deep embedded clustering with local structure preservation[C]//Ijcai. 2017, 17: 1753-1759.

[26] Yang B, Fu X, Sidiropoulos N D, et al. Towards k-means-friendly spaces: Simultaneous deep learning and clustering[C]//international conference on machine learning. PMLR, 2017: 3861-3870.

[27] Chang J, Wang L, Meng G, et al. Deep adaptive image clustering[C]//Proceedings of the IEEE international conference on computer vision. 2017: 5879-5887.

[28] Li Y, Hu P, Liu Z, et al. Contrastive clustering[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(10): 8547-8555.

[29] Dumoulin V, Belghazi I, Poole B, et al. Adversarially learned inference[J]. arXiv preprint arXiv:1606.00704, 2016.

[30] Mukherjee S, Asnani H, Lin E, et al. Clustergan: Latent space clustering in generative adversarial networks[C]//Proceedings of the AAAI conference on artificial intelligence. 2019, 33(01): 4610-4617.

[31] Bo D, Wang X, Shi C, et al. Structural deep clustering network[C]//Proceedings of the web conference 2020. 2020: 1400-1410.

[32] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[33] Chen D, Lv J, Zhang Y. Unsupervised multi-manifold clustering by learning deep representation[C]//Workshops at the thirty-first AAAI conference on artificial intelligence. 2017.

[34] Zhang T, Ji P, Harandi M, et al. Neural collaborative subspace clustering[C]//International Conference on Machine Learning. PMLR, 2019: 7384-7393.

[35] Zhou L, Xiao B, Liu X, et al. Latent distribution preserving deep subspace clustering[C]//28th International joint conference on artificial intelligence. 2019.

[36] Fard M M, Thonet T, Gaussier E. Deep k-means: Jointly clustering with k-means and learning representations[J]. Pattern Recognition Letters, 2020, 138: 185-192.

[37] Ren Y, Wang N, Li M, et al. Deep density-based image clustering[J]. Knowledge-Based Systems, 2020, 197: 105841.

[38] Yin M, Huang W, Gao J. Shared generative latent representation learning for multi-view clustering[C]//Proceedings of the AAAI conference on artificial intelligence. 2020, 34(04): 6688-6695.

[39] Springenberg J T. Unsupervised and semi-supervised learning with categorical generative adversarial networks[J]. arXiv preprint arXiv:1511.06390, 2015.

[40] Ghasedi K, Wang X, Deng C, et al. Balanced self-paced learning for generative adversarial clustering network[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 4391-4400.

[41] Xu C, Guan Z, Zhao W, et al. Adversarial incomplete multi-view clustering[C]//IJCAI. 2019, 7: 3933-3939.

[42] Jiang Z, Zheng Y, Tan H, et al. Variational deep embedding: An unsupervised and generative approach to clustering[J]. arXiv preprint arXiv:1611.05148, 2016.

[43] Yang B, Fu X, Sidiropoulos N D, et al. Towards k-means-friendly spaces: Simultaneous deep learning and clustering[C]//international conference on machine learning. PMLR, 2017: 3861-3870.

[44] Yang J, Parikh D, Batra D. Joint unsupervised learning of deep representations and image clusters[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 5147-5156.

[45] Shaham U, Stanton K, Li H, et al. Spectralnet: Spectral clustering using deep neural networks[J]. arXiv preprint arXiv:1801.01587, 2018.

[46] Yang L, Cheung N M, Li J, et al. Deep clustering by gaussian mixture variational autoencoders with graph embedding[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 6440-6449.

[47] Zhang J, Li C G, You C, et al. Self-supervised convolutional subspace clustering network[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 5473-5482.

[48] Zhao J, Lu D, Ma K, et al. Deep image clustering with category-style representation[C]//Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16. Springer International Publishing, 2020: 54-70.

[49] Yang X, Deng C, Zheng F, et al. Deep spectral clustering using dual autoencoder network[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 4066-4075.

[50] Li Y, Hu P, Liu Z, et al. Contrastive clustering[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(10): 8547-8555.

[51] Yan Y, Hao H, Xu B, et al. Image clustering via deep embedded dimensionality reduction and probability-based triplet loss[J]. IEEE Transactions on Image Processing, 2020, 29: 5652-5661.

[52] Huang S, Ota K, Dong M, et al. MultiSpectralNet: Spectral clustering using deep neural network for multi-view data[J]. IEEE Transactions on Computational Social Systems, 2019, 6(4): 749-760.

[53] Ma Q, Zheng J, Li S, et al. Learning representations for time series clustering[J]. Advances in neural information processing systems, 2019, 32.

[54] Liu Z, Wang Y, Vaidya S, et al. Kan: Kolmogorov-arnold networks[J]. arXiv preprint arXiv:2404.19756, 2024.

[55] Kolmogorov A N. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables[M]. American Mathematical Society, 1961.

[56] Yann LeCun, "The MNIST database of handwritten digits," [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: Feb. 23, 2025].https://www.cs.toronto.edu/~kriz/cifar.html

[57] Alex Krizhevsky, "CIFAR-10 and CIFAR-100 datasets," [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: Feb. 23, 2025].

[58] Alex Krizhevsky, "CIFAR-10 and CIFAR-100 datasets," [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: Feb. 23, 2025].http://qwone.com/~jason/20Newsgroups/

[59] Zalando Research, "Fashion-MNIST," [Online]. Available: https://github.com/zalandoresearch/fashion-mnist. [Accessed: Feb. 23, 2025].https://www.cs.ucr.edu/~eamonn/time_series_data/

[60] Jason Rennie, "20 Newsgroups dataset," [Online]. Available: http://qwone.com/~jason/20Newsgroups/. [Accessed: Feb. 23, 2025]. https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcgadata

[61] KDD, "Reuters-21578 dataset," [Online]. Available: https://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html. [Accessed: Feb. 23, 2025].https://www.tensorflow.org/datasets/community_catalog/huggingface/speech_commands

[62] Eamonn Keogh, "UCR Time Series Classification Archive," [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data/. [Accessed: Feb. 23, 2025]. https://archive.ics.uci.edu/ml/datasets/iris

[63] PhysioNet, "ECG-ID Database," [Online]. Available: https://www.physionet.org/physiobank/database/ecgiddb/. [Accessed: Feb. 23, 2025].https://www.ncbi.nlm.nih.gov/PubMed

[64] National Cancer Institute, "The Cancer Genome Atlas (TCGA)," [Online]. Available: https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcgadata. [Accessed: Feb. 23, 2025].

[65] Enron, "The Enron Corpus," [Online]. Available: https://www.cs.cmu.edu/~enron/. [Accessed: Feb. 23, 2025].

[66] TensorFlow, "Speech Commands Dataset," [Online]. Available: https://www.tensorflow.org/datasets/community_catalog/huggingface/speech_commands. [Accessed: Feb. 23, 2025].

[67] LDC, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," [Online]. Available: https://www.ldc.upenn.edu/collaborations/past-projects/timit. [Accessed: Feb. 23, 2025].

[68] UCI Machine Learning Repository, "Iris Dataset," [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris. [Accessed: Feb. 23, 2025].

[69] UCI Machine Learning Repository, "Wine Dataset," [Online]. Available: https://archive.ics.uci.edu/ml/datasets/wine. [Accessed: Feb. 23, 2025].

[70] National Library of Medicine, "PubMed Database," [Online]. Available: https://www.ncbi.nlm.nih.gov/PubMed. [Accessed: Feb. 23, 2025].