The Interplay Between Clustering Algorithms and Representation Learning

1. Introduction: Defining the Pillars - Clustering and Representation Learning

1.1 Unsupervised Learning Context

Machine learning encompasses various paradigms, among which unsupervised learning stands out for its focus on discovering inherent patterns and structures within unlabeled data.[1] Unlike supervised learning, which relies on predefined labels to guide the learning process, unsupervised methods must infer relationships and groupings directly from the data itself. Within this broad domain, clustering and representation learning emerge as two fundamental and often interconnected tasks.[4] Both aim to extract meaningful information from raw data, but they approach this goal from different perspectives. This report delves into the intricate relationship between these two pillars, exploring how they influence each other and how their integration, particularly through deep learning, is advancing the capabilities of machine learning systems.

1.2 Clustering: Discovering Groups

Clustering is the quintessential unsupervised task focused on partitioning a dataset into distinct groups, known as clusters.[1] The core principle is to group data points such that items within the same cluster exhibit higher similarity to each other than to items belonging to different clusters.[6] The primary objective is not to predict a specific target variable, but rather to uncover underlying patterns, natural groupings, or latent structures inherent in the data.[1] This process allows for data exploration, summarization, and the identification of distinct categories without prior knowledge of these categories.[12] Clustering algorithms find widespread application in diverse fields, including customer segmentation for targeted marketing [1], image recognition and segmentation [1], document analysis and topic modeling [3], anomaly detection [1], and bioinformatics.[17]

1.3 Representation Learning: Finding Meaningful Features

Representation learning, often used interchangeably with feature learning, encompasses a set of techniques designed to automatically discover useful representations (features or embeddings) of data, typically transforming raw input into a format more suitable for downstream machine learning tasks like classification or clustering.[7] The necessity for representation learning arises because raw data, such

as pixels in an image or characters in text, is often high-dimensional, complex, and may not directly reveal the underlying factors of variation relevant for a given task.[9] Machines struggle to process this raw complexity directly.[18]

The overarching goals of representation learning are multi-faceted. It aims to transform data into a more informative, compact, and processable format.[18] This often involves dimensionality reduction, projecting high-dimensional data into a lower-dimensional space while preserving essential information, thereby mitigating the "curse of dimensionality".[18] It seeks to capture the salient features and underlying structure of the data [18], potentially enhancing interpretability by making learned features semantically meaningful.[18] Ultimately, effective representation learning facilitates improved performance on subsequent tasks by providing features that make patterns more apparent or separable.[20] The learned representations are frequently numerical vectors, often termed embeddings, residing in a potentially lower-dimensional latent space.[22] The quality of these representations is critical, as they form the substrate upon which downstream algorithms operate.

1.4 The Core Question

Clustering seeks groups, and representation learning seeks informative features. How are these two fundamental unsupervised learning tasks related? This report aims to dissect this relationship, moving from the basic dependence of clustering algorithms on input features to the use of representation learning as a preparatory step, and culminating in the modern paradigm of Deep Clustering, where representation learning and clustering are jointly optimized. We will explore the synergy, the specific algorithms that bridge the gap, the benefits and drawbacks of their integration, and the diverse applications that leverage this powerful combination. Understanding this interplay is crucial for effectively applying and advancing unsupervised learning techniques.

2. The Foundational Link: How Clustering Relies on Representations

2.1 Clustering Algorithms Operate on Features

Clustering algorithms, despite their diverse mechanisms, do not operate on abstract concepts or raw data formats directly. Instead, they function within a defined *feature space*, processing numerical vectors that represent the data points.[7] Whether it's K-Means calculating distances to centroids or DBSCAN assessing neighborhood density, these operations are performed on the features provided as input. The very definition of "similarity" or "distance" underpinning a clustering algorithm (e.g., the

commonly used Euclidean distance for K-Means [3]) is computed based on the coordinates of data points within this feature space. Consequently, the nature and quality of this representation fundamentally dictate how the algorithm perceives the data's structure and, ultimately, the resulting clusters.

2.2 Impact of Representation Quality

The success or failure of a clustering endeavor is inextricably linked to the quality of the input data representation.[17] A well-chosen representation can make inherent groupings in the data obvious, presenting clusters that are compact, well-separated, and perhaps aligned with the geometric assumptions of the chosen algorithm.[21] For instance, a representation that maps semantically similar items close together in the feature space directly facilitates the goal of semantic clustering.[24]

Conversely, a poor representation can obscure natural structures, leading to suboptimal or misleading results. If the features do not capture the relevant aspects of similarity for the desired clustering, the algorithm may fail to identify meaningful groups. Clusters might appear non-spherical, overlap significantly, or exhibit vastly different densities in the feature space, violating the implicit or explicit assumptions of many algorithms, particularly partition-based methods like K-Means.[1] Furthermore, the "curse of dimensionality" poses a significant challenge: in very high-dimensional spaces, the concept of distance can become less meaningful, points tend to appear equidistant, and density becomes sparse, hindering the performance of distance-based and density-based clustering algorithms alike.[9] This necessitates methods, like representation learning, to reduce dimensionality and create more tractable feature spaces.

2.3 Algorithm Sensitivity to Representation

Different clustering algorithms exhibit varying sensitivities to the characteristics of the feature space.

- **K-Means:** This popular centroid-based algorithm partitions data into a predefined number, K, of clusters by minimizing the sum of squared distances between points and their assigned cluster centroid.[1] Its effectiveness hinges on the assumption that clusters are roughly spherical, equally sized, and well-separated in the feature space.[1] K-Means struggles significantly when faced with clusters of arbitrary shapes (e.g., elongated, concentric), varying densities, or datasets containing numerous outliers, as these factors distort the calculation of centroids and the distance-based assignments.[1] The algorithm's performance is also sensitive to the scale of features; features with larger ranges can disproportionately influence distance calculations. This often necessitates feature

scaling or standardization – a direct modification of the data representation – to ensure all features contribute more equally and to improve cluster quality.[13] This requirement for pre-processing underscores the algorithm's dependence on a suitably prepared representation.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** As a density-based algorithm, DBSCAN defines clusters as dense regions of points separated by sparser regions.[1] Its strength lies in its ability to discover clusters of arbitrary shapes and its inherent mechanism for identifying noise points (outliers) that do not belong to any dense cluster.[1] Unlike K-Means, it does not require the number of clusters to be specified beforehand.[1] However, DBSCAN's performance is critically dependent on the density characteristics *within the chosen feature space*. The definition of density is controlled by two key parameters: eps (ϵ), the maximum distance defining a neighborhood, and minPts, the minimum number of points required within an ϵ-neighborhood to form a dense core point.[8] The optimal values for these parameters are entirely dependent on the scale and density distribution of the data *in its current representation*. DBSCAN may struggle to correctly identify clusters if they exhibit significantly different densities within the same feature space, as a single set of eps and minPts may not be appropriate for all clusters.[41]

The contrasting sensitivities of K-Means and DBSCAN highlight a crucial point: the geometric and density properties that these algorithms rely on are properties *of the representation*, not necessarily of the raw data itself. An algorithm's success depends on the alignment between its inherent assumptions and the structure revealed by the chosen features. Parameters like K, ϵ, and minPts act as probes into the structure of this representation space.

Table 1: Comparison of K-Means and DBSCAN Clustering Algorithms

Feature	K-Means	DBSCAN
Principle	Centroid-based; partitions data based on distance to cluster means [1]	Density-based; groups closely packed points based on neighborhood density [1]
Cluster Shape	Assumes spherical or convex shapes [6]	Can find arbitrarily shaped clusters [1]

Need to Specify K	Yes, the number of clusters (K) must be predefined [1]	No, determines the number of clusters automatically [1]
Noise Handling	Sensitive to outliers; forces all points into clusters [8]	Robust to noise; explicitly identifies outliers as noise points [13]
Initialization	Sensitive to initial centroid placement; may converge to local optima [3]	Less sensitive to the starting point; mostly deterministic (except edge points) [1]
Parameters	Number of clusters (K) [2]	Neighborhood radius (ϵ) and minimum points (minPts) [13]

3. Enhancing Clustering through Representation Learning

Given the critical dependence of clustering algorithms on the input representation, a natural strategy is to employ representation learning techniques as a preliminary step to transform the data into a more suitable format before applying a clustering algorithm.[7] This "two-stage" approach aims to mitigate issues like the curse of dimensionality, extract more salient features, or capture complex data structures, thereby creating a feature space more amenable to partitioning by clustering methods.[7]

3.1 Representation Learning as a Precursor

The core idea is to first learn a mapping from the original data space to a new, often lower-dimensional, feature space using an unsupervised representation learning technique. The resulting transformed features (embeddings) are then fed into a standard clustering algorithm like K-Means or DBSCAN. This decouples the process: representation learning focuses on creating a "good" general-purpose or dimensionality-reduced representation, while the subsequent clustering algorithm focuses solely on grouping points within that new space.[7]

3.2 Linear Dimensionality Reduction: PCA

Principal Component Analysis (PCA) is a widely used statistical technique for linear dimensionality reduction.[18] Its objective is to identify a set of orthogonal axes, called principal components, which are linear combinations of the original features, such that they capture the maximum possible variance in the data.[18] These components are the eigenvectors of the data's covariance matrix, ordered by their corresponding

eigenvalues which represent the amount of variance captured.[39]

In the context of clustering pre-processing, PCA is valuable for several reasons. By projecting the data onto a smaller number of principal components (typically those capturing the most variance), it significantly reduces dimensionality, combating the curse of dimensionality and often reducing computational cost for the subsequent clustering step.[22] If the natural separation between clusters aligns with the directions of high variance in the data, PCA can enhance cluster separability.[39] It also facilitates visualization by projecting data onto two or three principal components.[44] However, PCA's utility is limited by its linearity.[18] It cannot effectively capture complex, non-linear structures or manifolds within the data. Furthermore, its focus on maximizing variance means it might discard lower-variance components that could be crucial for separating certain clusters, potentially merging distinct groups if their separation does not align with the principal axes of variation. Often, data standardization is recommended before applying PCA to prevent variables with large variances from dominating the principal components.[39]

3.3 Non-Linear Feature Extraction: Autoencoders (AEs)

Autoencoders (AEs) offer a powerful alternative for non-linear representation learning.[5] They are a type of artificial neural network trained in an unsupervised manner to reconstruct their input.[4] An AE consists of two main parts: an *encoder* network that maps the input data x to a compressed representation z in a lower-dimensional latent space (often called the bottleneck), and a *decoder* network that attempts to reconstruct the original input x^ from the latent representation z.[4] The network is trained by minimizing a reconstruction loss, typically the difference between x and x^ (e.g., mean squared error).[5]

For clustering purposes, the key output is the latent representation z generated by the encoder. This embedding captures potentially complex, non-linear features of the data, effectively learning a representation along the data's underlying manifold.[5] These learned embeddings can then serve as input features for standard clustering algorithms.[7] Because AEs can model non-linear relationships, they are often more effective than PCA at capturing intricate structures in data like images or text, potentially leading to better cluster separation in the latent space. Various AE architectures exist, offering different properties: Denoising Autoencoders are trained to reconstruct clean data from corrupted input, learning more robust features [4]; Sparse Autoencoders impose constraints on hidden layer activations, encouraging specialized feature detectors [4]; Variational Autoencoders (VAEs) introduce a probabilistic approach, learning a distribution over the latent space, which is useful

for generative tasks and provides a principled way to handle uncertainty.[18]

3.4 Semantic Representations: Word Embeddings (Word2Vec)

In the domain of Natural Language Processing (NLP), traditional representations like bag-of-words or TF-IDF often suffer from high dimensionality and sparsity, and fail to capture semantic meaning. Word embedding techniques, notably Word2Vec, revolutionized text representation by learning dense, low-dimensional vector representations for words.[18] Word2Vec models (specifically, the Continuous Bag-of-Words (CBOW) and Skip-gram architectures [27]) are shallow neural networks trained on large text corpora. They learn word vectors such that words appearing in similar linguistic contexts have vectors that are close to each other in the embedding space (typically measured by cosine similarity).[27]

These learned embeddings capture rich semantic and syntactic relationships between words (e.g., analogies like "king - man + woman ≈ queen" can often be resolved using vector arithmetic [29]). For clustering, Word2Vec provides a way to represent text data (words, sentences, or documents by aggregating word vectors) in a continuous vector space where distance reflects semantic similarity.[34] This dramatically improves the potential for clustering algorithms like K-Means to group related documents or words effectively, as the representation itself encodes the semantic structure targeted by the clustering task.[34] Extensions like Doc2Vec learn embeddings for entire paragraphs or documents directly, further facilitating document clustering.[19] This shift from sparse, surface-level features to dense, semantic embeddings represents a prime example of how representation learning can fundamentally improve the suitability of data for clustering.

While these two-stage approaches are practical and often yield improvements over clustering on raw data, they possess an inherent limitation: the representation learning process is typically optimized for a goal other than clustering (e.g., variance maximization for PCA, reconstruction for AEs, context prediction for Word2Vec). The learned representation, while potentially better than the raw data, may still not be optimal for the specific task of separating the data into the desired clusters.[7] This motivates the development of methods that integrate representation learning and clustering more tightly.

Table 2: Overview of Representation Learning Techniques for Clustering Pre-processing

Technique	Core Idea	Type	Benefit for Clustering	Example Application
PCA	Find linear projections (principal components) maximizing data variance [25]	Linear	Dimensionality reduction, potential linear separation, visualization [22]	Reducing gene expression data dimensions before K-Means
Autoencoder (AE)	Learn compressed latent representation via reconstruction bottleneck [4]	Non-linear	Non-linear dimensionality reduction, capture complex manifolds [5]	Extracting image features for subsequent DBSCAN clustering
Word2Vec	Learn dense word vectors based on linguistic context [27]	Non-linear	Creates semantic vector space for text, distance reflects meaning [29]	Generating document embeddings for topic clustering

4. Deep Clustering: The Paradigm of Joint Optimization

4.1 Motivation: Beyond Sequential Processing

The sequential application of representation learning followed by clustering, while beneficial, suffers from a fundamental disconnect. The representation learning phase, whether using PCA, Autoencoders, or other techniques, is typically optimized based on criteria like variance maximization or reconstruction accuracy, which are agnostic to the ultimate goal of partitioning the data into meaningful clusters.[7] The features learned might preserve overall data structure or reduce dimensionality effectively, but they are not explicitly encouraged to enhance the separability of the specific groups targeted by the clustering task. This can lead to suboptimal clustering performance, as the ideal representation for grouping might differ from one optimized for reconstruction or variance.[7] Recognizing this limitation spurred the development of approaches that integrate these two steps more closely.

4.2 The Core Concept: Simultaneous Learning

Deep Clustering (DC) emerges as a paradigm shift, aiming to overcome the limitations

of sequential methods by *jointly* optimizing the processes of representation learning and cluster assignment within a unified framework.[7] Instead of treating them as separate stages, DC methods leverage the power of deep neural networks to learn data representations that are explicitly tailored or "friendly" towards the clustering objective.[7] The core idea is to create a feedback loop where the quality of the current clustering informs the learning of the representation, and the improved representation, in turn, facilitates better clustering.

4.3 Role of Deep Neural Networks (DNNs)

Deep neural networks, particularly architectures based on Autoencoders, are instrumental in enabling this joint optimization.[31] DNNs excel at learning complex, hierarchical, and non-linear mappings from high-dimensional input data to lower-dimensional latent spaces.[11] Crucially, these networks are differentiable, meaning that gradients calculated from a loss function defined at the output (or in the latent space) can be backpropagated through the network to update the parameters (weights and biases) of the representation learning layers (e.g., the encoder).[31] Autoencoders provide a natural structure for this: the encoder maps data to an embedding space where clustering can be conceptually performed, and the decoder (while sometimes discarded after pre-training [31]) can provide a reconstruction loss component.

4.4 Clustering-Specific Loss Functions

The key innovation in many DC methods is the introduction of a *clustering loss* term into the overall objective function. This loss is designed to quantify how well the current learned representation (the embeddings in the latent space) conforms to desirable clustering properties. It typically encourages intra-cluster compactness (points assigned to the same cluster should be close in the embedding space) and inter-cluster separability (points or centroids belonging to different clusters should be far apart).

Various forms of clustering losses have been proposed. For example, Deep Embedded Clustering (DEC) uses the Kullback-Leibler (KL) divergence between the current soft cluster assignment distribution (based on embedding similarity to cluster centroids) and a target distribution that sharpens assignments based on high-confidence predictions.[31] Other approaches might directly incorporate a K-means-like loss calculated on the embeddings [50], adapt contrastive learning losses to operate on cluster assignments or prototypes [7], maximize the mutual information between input views and cluster assignments [7], or use measures of sample stability relative to cluster centroids.[49] The gradient of this clustering loss is then used to update the parameters

of the representation learning network (e.g., the AE encoder), effectively guiding the network to produce embeddings that minimize this clustering objective.

4.5 The Reconstruction Loss Trade-off

In many Autoencoder-based Deep Clustering methods (like IDEC, DEPICT), the objective function is a composite one, combining both a clustering loss and the traditional AE reconstruction loss.[50] The reconstruction loss (Lrec) penalizes differences between the original input and the output reconstructed by the decoder from the latent embedding. Its inclusion serves a critical purpose: it acts as a regularizer, forcing the latent representation to retain sufficient information about the original data to allow for accurate reconstruction.[54] This helps prevent the representation learning process from collapsing into a trivial solution (a *degenerate solution*) where, for instance, all data points are mapped to a single point in the latent space, which might perfectly satisfy a naive clustering loss but discards all meaningful data structure.[43] There is thus an inherent tension: the clustering loss (Lclus) pushes the representation towards distinct clusters, while the reconstruction loss pushes it towards preserving local structure and data fidelity. The overall loss is often a weighted sum, $L=\gamma L_{clus}+(1-\gamma)L_{rec}$ (or similar formulations [11]), where the weighting parameter γ controls the trade-off. Balancing these competing objectives is a key challenge in designing and training effective AE-based DC models.[54] This joint optimization framework effectively transforms the unsupervised clustering problem into a form of self-supervised representation learning, where supervisory signals are derived internally from the data's evolving cluster structure.[31]

5. A Taxonomy of Deep Clustering Approaches

5.1 Overview

The field of Deep Clustering has rapidly evolved, leading to a diverse array of methods that implement the joint optimization of representation learning and clustering in different ways. To navigate this landscape, several taxonomies have been proposed, often categorizing methods based on how the representation learning module and the clustering module interact.[7] Following prominent surveys [7], we can classify DC approaches into four main categories based on their interaction strategy: Multi-stage, Iterative, Generative, and Simultaneous. This taxonomy reflects a general trend towards tighter integration between the two components.

5.2 Multi-stage Deep Clustering

This represents the simplest form of integration, essentially formalizing the two-stage approach discussed earlier.

- **Description:** Representation learning and clustering are performed sequentially and independently. First, a deep unsupervised learning model (e.g., Autoencoder, pre-trained contrastive model) is trained to generate fixed feature embeddings for the entire dataset. Second, a conventional clustering algorithm (e.g., K-Means, Spectral Clustering) is applied to these static embeddings.[7]
- **Interaction:** There is no direct interaction or feedback loop between the two stages during optimization. The representation is learned without knowledge of the specific clustering task that will follow.[7]
- **Examples:** Early methods involved training an Autoencoder and then applying K-Means to the bottleneck features.[7] More recent examples include using powerful representations learned via self-supervised methods like SimCLR or MoCo as input for clustering.[43] Deep subspace clustering methods that first learn representations and an affinity matrix, followed by spectral clustering, also fall here.[7] IDFD learns "clustering-friendly" representations first, then applies K-means.[7]

5.3 Iterative Deep Clustering

This category introduces an explicit feedback mechanism, optimizing the two components in an alternating fashion.

- **Description:** The process cycles between updating the clustering assignments based on the current representations and updating the representation learning network using information derived from the current clustering results.[7]
- **Interaction:** An indirect feedback loop exists. The clustering module generates information (typically pseudo-labels or relational constraints) that acts as a supervisory signal for training the representation learning module in the subsequent iteration.
- **Sub-types & Examples:**
  - *Individual Supervision (Pseudo-Labeling):* Cluster assignments obtained from the current embeddings (e.g., using K-means) are treated as pseudo-labels. The representation learning network is then trained, often like a classifier, to predict these pseudo-labels for augmented versions of the data points. DeepCluster is the canonical example, alternating between K-means clustering of CNN features and training the CNN to predict the K-means assignments.[7] Other methods like SCAN and SPICE refine this by using self-labeling based on high-confidence predictions or semi-supervised techniques.[7]
  - *Relational Supervision:* Instead of using absolute pseudo-labels, these methods leverage pairwise relationships derived from them (e.g., whether two

instances belong to the same pseudo-cluster). This information is used to train the representation learner, for instance, by training a binary classifier to predict if two points share the same pseudo-label or by using similarity of cluster assignment probabilities as a regression target.[7]

5.4 Generative Deep Clustering

These methods integrate clustering within the framework of deep generative models, such as Variational Autoencoders (VAEs) or Generative Adversarial Networks (GANs).

- **Description:** They typically assume a latent space where the data distribution is modeled as a mixture of simpler distributions (e.g., Gaussians), with each component corresponding to a cluster. The model learns to encode data into this latent space and decode it back, while simultaneously inferring the cluster assignments based on the latent mixture model.[7]
- **Interaction:** Representation learning (encoding/decoding) and clustering (latent mixture model parameters and assignments) are jointly optimized, usually by maximizing the evidence lower bound (ELBO) for VAEs or through the adversarial training process for GANs.
- **Examples:** VAE-based methods like VaDE and GMVAE explicitly model the latent space prior as a Gaussian Mixture Model (GMM), jointly learning the AE parameters and the GMM parameters.[7] ClusterGAN incorporates a latent GMM into the GAN framework.[7] The AE-CM approach theoretically links a specific GMM objective to an AE loss function.[51]

5.5 Simultaneous Deep Clustering

This category represents the tightest integration, where representation learning and clustering are optimized concurrently in an end-to-end fashion using a single, often composite, objective function. This is currently the most dynamic area of research.[7]

- **Description:** Both the representation network parameters and the cluster assignments (or parameters defining clusters, like centroids) are updated simultaneously in each training step via gradient backpropagation based on a unified loss function.[7]
- **Interaction:** Direct and concurrent feedback. The loss function is designed to simultaneously encourage clustering-friendly representations and refine cluster structures based on those representations.
- **Sub-types & Examples:**
  - *Autoencoder with Self-Training:* This includes the influential Deep Embedded Clustering (DEC) algorithm, which pre-trains an AE and then fine-tunes the encoder using a KL divergence-based clustering loss that iteratively refines

soft assignments and centroids.[7] Improved DEC (IDEC) adds the reconstruction loss back during fine-tuning.[7] DEPICT uses relative entropy minimization with regularization.[55]

- *Mutual Information Maximization:* Methods like DCCM, VCAMI, and IIC aim to maximize the mutual information between representations of different views of the data or between representations and cluster assignments.[7]
- *Contrastive Deep Clustering:* These methods adapt the principles of contrastive self-supervised learning. Examples include SwAV, which enforces consistency between cluster assignments ("codes") of different views of an image [7]; Contrastive Clustering (CC), which uses instance-to-prototype contrastive loss [7]; and numerous others like PICA, DRC, CRLC, SCL, MiCE, and NCC that explore different contrastive strategies, positive/negative sampling techniques, or non-contrastive frameworks to improve clustering.[7]
- *Hybrid Approaches:* Some methods combine elements from different categories, such as SCCL merging contrastive learning with self-training.[7]

This taxonomy highlights the evolution towards increasingly integrated approaches, driven by the understanding that tightly coupling representation learning with the clustering objective can yield significant performance gains, especially for complex data. The "Simultaneous" category, particularly methods incorporating contrastive learning techniques, represents the current research frontier, adapting powerful self-supervised learning paradigms to the specific challenges of unsupervised clustering.[7]

Table 3: Taxonomy of Deep Clustering Approaches (Based on [7])

Category	Core Idea	Optimization Strategy	Examples	Key Characteristics
Multi-stage	Sequential: Learn representation, then cluster	Independent stages	AE+K-Means, SSL Pretrain+K-Means, IDFD [7]	Simple, leverages existing methods, but representation is not cluster-optimized.
Iterative	Alternating: Update	Alternating optimization,	DeepCluster, SCAN,	Introduces feedback, relies

	representation using clustering results, and vice-versa	indirect feedback	Relational Supervision [7]	on pseudo-label quality, can be computationally intensive.
Generative	Model data generation with latent cluster structure	Joint optimization via generative model objective	VaDE, GMVAE, ClusterGAN, AE-CM [7]	Principled probabilistic framework, can generate data, model assumptions may be strong.
Simultaneous	End-to-end: Optimize representation and clustering concurrently	Joint optimization via unified loss, direct feedback	DEC, IDEC, SwAV, CC, DCCM, PICA, NCC [7]	Tightest integration, often SOTA performance, can be complex, risk of instability.

6. Key Algorithms and Techniques at the Intersection

Examining specific algorithms provides concrete examples of how the joint optimization of representation learning and clustering is realized. These methods illustrate different philosophies for integrating the two components and generating the necessary supervisory signals.

6.1 Deep Embedded Clustering (DEC) / IDEC

DEC [31] was a pioneering work in simultaneous deep clustering.

- **Mechanism:** It starts by pre-training a deep autoencoder on the input data to initialize the encoder network fθ. The decoder is then discarded. The core of DEC involves iteratively refining the encoder parameters θ and a set of K cluster centroids {μj}j=1K residing in the latent space Z. In each iteration, it first computes soft assignments qij for each data point i to each cluster j. This assignment measures the similarity between the point's embedding zi=fθ(xi) and the centroid μj, typically using the Student's t-distribution kernel: qij=Σj′(1+||zi−μj′||2/α)−2α+1(1+||zi−μj||2/α)−2α+1.[31] Second, it computes a target distribution pij by squaring and normalizing the soft assignments qij, effectively increasing the confidence of assignments and putting more emphasis on points already close to their centroids: pij=Σj′(qij′2/Σiqij′)qij2/Σiqij.[31] Finally, it minimizes the KL divergence loss L=KL(P||Q)=ΣiΣjpijlogqijpij between the target

distribution P and the current soft assignments Q.[31] This loss is minimized jointly with respect to the encoder parameters θ (via backpropagation) and the cluster centroids μj (via gradient descent).[31]

- **Interaction:** The clustering loss (KL divergence) directly guides the updates to the encoder, pushing embeddings zi closer to their assigned centroids μj. The cluster centroids are also refined based on the embeddings. This creates a self-training loop where high-confidence assignments provide supervision.
- **IDEC Improvement:** A key limitation of DEC is that optimizing only the KL clustering loss can distort the feature space, potentially corrupting the learned representation by ignoring the local structure of the data.[54] Improved DEC (IDEC) addresses this by reintroducing the autoencoder's reconstruction loss during the fine-tuning phase.[7] The joint objective becomes a combination of the KL clustering loss and the reconstruction loss, forcing the encoder to learn features that are both discriminative for clustering and faithful to the original data structure.[54]

6.2 DeepCluster

DeepCluster [64] exemplifies the iterative approach.

- **Mechanism:** It utilizes a standard convolutional neural network (CNN) as the feature extractor (encoder). The algorithm alternates between two steps in each epoch: 1) Cluster Assignment: The current CNN generates feature embeddings for all data points. These embeddings are then clustered using a standard K-means algorithm. 2) Network Update: The cluster assignments obtained from K-means are treated as pseudo-labels for the data points. The CNN is then trained for one epoch using these pseudo-labels, typically optimizing a standard classification loss (e.g., cross-entropy) to predict the pseudo-label for each input image (often augmented).[7]
- **Interaction:** This is a clear example of iterative interaction. The clustering step (K-means) provides the supervisory signal (pseudo-labels) for the representation learning step (CNN training). The updated CNN then produces potentially better features for the next K-means clustering step.

6.3 Contrastive Approaches (e.g., SwAV, CC)

These methods adapt the highly successful paradigm of contrastive self-supervised learning for the clustering task.[7]

- **General Idea:** Contrastive learning typically works by pulling representations of augmented views of the same instance ("positive pairs") closer together, while pushing representations of different instances ("negative pairs") further apart.

Deep clustering adaptations modify this core idea to incorporate cluster structure.

- **SwAV (Swapping Assignments between Views):** SwAV [63] avoids direct comparison of features and the need for explicit negative pairs within the loss. It introduces a set of learnable "prototype" vectors. During training, it computes cluster assignments ("codes") for different augmented views of an image by mapping their embeddings to these prototypes (e.g., using Sinkhorn-Knopp algorithm for balanced assignments). The core idea is a "swapped" prediction task: predict the code assigned to one view using the embedding of another view from the same original image.[63] This enforces consistency: if two views are semantically similar, their embeddings should lead to similar cluster assignments (codes). It uses an online clustering approach, avoiding the need to cluster the entire dataset at once, making it scalable.[59] SwAV also introduced the multi-crop augmentation strategy (using multiple lower-resolution crops alongside standard-resolution crops) to increase the number of positive pairs without significant computational overhead.[63]
- **CC (Contrastive Clustering):** CC [7] directly incorporates cluster centroids (prototypes) into the contrastive framework. It uses two main loss components: an instance-to-instance contrastive loss (like SimCLR) to learn good initial representations, and an instance-to-prototype contrastive loss. The latter encourages each instance's embedding to be close to the prototype of its assigned cluster and far from other prototypes. The prototypes themselves are learnable parameters updated during training. This explicitly optimizes for both instance discrimination and cluster compactness/separation.
- **Other Variations (PICA, NCC, etc.):** Further research explores refining contrastive clustering by addressing potential issues. Some methods focus on partition uncertainty.[7] Others, like NCC (Non-Contrastive Clustering) [66], build upon non-contrastive self-supervised methods like BYOL (which avoids negative pairs altogether) and adapt them for clustering using techniques like positive neighbor sampling and prototype-based contrastive losses (ProtoCL) to avoid issues like class collision (negative samples belonging to the same semantic class as the anchor) and representation collapse.[66]

6.4 Cluster Assignments as Features/Guidance

Across these diverse algorithms, a common thread is the use of cluster assignment information – whether hard assignments from K-means (DeepCluster), soft assignments based on similarity (DEC), online codes (SwAV), or relationships to prototypes (CC) – as the critical signal that guides the representation learner.[7] This information, generated dynamically from the data's current perceived structure, forms

the self-supervisory signal. Some recent work like FEC (Feature Extraction with Clustering) even proposes using the cluster assignments generated at different layers of the network as interpretable features themselves, making the feature extraction process more transparent.[68] The spectrum of approaches reflects different strategies for generating and utilizing this internal supervisory signal, from periodic batch clustering to online code prediction and direct prototype learning.

7. Benefits, Challenges, and Trade-offs

The integration of representation learning and clustering, particularly through deep clustering methods, offers significant advantages but also presents notable challenges and trade-offs that must be considered.

7.1 Advantages of Joint Optimization (Deep Clustering)

- **Improved Clustering Performance:** The primary motivation and often-realized benefit of DC is its potential to achieve superior clustering results compared to traditional methods or sequential approaches, especially on complex, high-dimensional datasets like images or text where raw features are inadequate.[7] By learning features specifically optimized for cluster separation, DC can uncover groupings that might be missed otherwise.
- **Task-Specific Representations:** Unlike generic representation learning, DC learns features tailored to the specific clustering task at hand.[7] The learned embedding space is shaped by the objective of separating the inherent groups within the given dataset, leading to representations that are potentially more discriminative for that particular partitioning.
- **End-to-End Learning:** Simultaneous DC methods offer the elegance of an end-to-end trainable system.[38] This allows for joint optimization of all components and potentially leads to better overall solutions compared to staged or iterative approaches where optimization is performed piecewise.
- **Handling Complexity and Dimensionality:** DC leverages the power of deep neural networks to model intricate non-linear relationships and effectively handle high-dimensional raw data (e.g., images, videos, text), overcoming the limitations of traditional algorithms susceptible to the curse of dimensionality.[9]

7.2 Disadvantages and Challenges

- **Increased Complexity:** Deep clustering models and their training procedures are inherently more complex than traditional clustering algorithms or simple two-stage approaches.[9] Designing effective architectures and loss functions requires significant expertise.

- **Hyperparameter Sensitivity:** The performance of DC methods can be highly sensitive to various hyperparameters, including the neural network architecture (depth, width), learning rates, batch sizes, the weighting between different loss components (e.g., clustering vs. reconstruction), and parameters specific to the clustering mechanism (e.g., number of clusters K, temperature in softmax, prototype parameters).[59] Tuning these parameters can be challenging without labeled validation data.
- **Degenerate Solutions:** A significant practical challenge is the risk of the optimization process converging to trivial or degenerate solutions.[43] For instance, the model might map all data points to a single point in the embedding space (representation collapse) or assign all points to a single cluster. This often occurs if the clustering loss component dominates inappropriately or if the optimization landscape is poorly conditioned. Preventing such outcomes requires careful design of the loss function (e.g., including regularization like reconstruction loss [54]), specific architectural choices, or careful initialization strategies.
- **Computational Cost:** Training deep neural networks, especially on large datasets, is computationally intensive. Iterative DC methods that involve repeated clustering steps (like DeepCluster with K-means) can add significant overhead.[59] Contrastive methods often require large batch sizes or memory banks to function effectively, further increasing resource demands.[21]
- **Pseudo-Label Quality and Error Propagation:** Iterative methods relying on pseudo-labels generated by clustering are susceptible to the quality of these labels.[65] If the initial clustering is poor, the resulting pseudo-labels will be noisy, potentially misleading the representation learning process in subsequent iterations and leading to error propagation.
- **Evaluation Difficulties:** Evaluating the quality of deep clustering is challenging due to the absence of ground truth labels in truly unsupervised settings. Standard clustering metrics (like Accuracy, NMI, ARI, Silhouette score, Dunn Index [3]) are often used on benchmark datasets with known classes, but these may not fully capture the quality of the learned representation itself or its suitability for downstream tasks.[21] Defining what constitutes a "good" clustering-friendly representation beyond performance on a specific metric remains an open research question.[21] There might be a trade-off between learning highly specialized features optimal for one clustering task versus learning more general, transferable representations useful for other downstream tasks.[21]

7.3 Choosing an Approach

The choice between different DC approaches (or whether to use DC at all) depends on several factors. For simpler, lower-dimensional, or tabular data where linear

separability might suffice, traditional clustering algorithms applied directly or after PCA might be adequate and more efficient.[50] Deep clustering methods shine when dealing with high-dimensional, unstructured data (images, text, etc.) where complex non-linear representations are needed.[9] The specific choice among DC methods involves trade-offs: iterative methods might be simpler to implement but potentially slower and sensitive to pseudo-label quality; generative methods offer a probabilistic framework but rely on model assumptions; simultaneous methods, especially contrastive ones, often achieve state-of-the-art results but can be complex and computationally demanding. The availability of computational resources, the dataset size, the need for interpretability, and whether some prior knowledge exists (e.g., for semi-supervised DC [23]) are all relevant considerations.

8. Applications Across Domains

The ability of deep clustering to handle complex, high-dimensional data and learn task-specific representations has led to its application in a wide variety of domains, often surpassing the capabilities of traditional clustering methods. The primary driver is the need to find meaningful groups in large, unlabeled datasets where manual feature engineering is infeasible or ineffective.[9]

8.1 Image Analysis

This is perhaps the most prominent application area for deep clustering, driven by the success of deep learning in computer vision.

- **Image Clustering:** Grouping large collections of unlabeled images based on visual similarity or semantic content.[64] Methods like DeepCluster [64], SwAV [63], and many others are frequently benchmarked on standard image datasets (e.g., ImageNet, CIFAR). This is useful for organizing photo libraries, visual search, or discovering categories in visual data.
- **Image Segmentation:** Clustering pixels within an image based on learned features to identify and delineate objects or regions.[15] Some architectures like FEC explicitly integrate clustering into the feature extraction process for vision tasks.[68]
- **Medical Imaging:** Deep clustering is applied to analyze medical images like histopathology slides (Whole Slide Images - WSIs) for identifying tissue types or abnormalities [33], or segmenting tumors in MRI scans.[16] The learned representations and clusters can aid in diagnosis or retrieval of similar cases.

8.2 Natural Language Processing (NLP)

Deep clustering, often leveraging powerful pre-trained language models or jointly

learned embeddings, finds applications in text analysis.

- **Document Clustering:** Grouping large sets of documents (news articles, scientific papers, emails) based on their topic or content.[4] This relies on obtaining meaningful document representations, often derived from word embeddings (like Word2Vec [34]) or sentence/document encoders (like BERT or Doc2Vec [19]), which are then clustered, potentially with joint optimization.[34]
- **Topic Modeling:** Discovering latent thematic structures within text corpora, where clusters correspond to topics.
- **Semantic Search and Information Retrieval:** Clustering can help organize search results into meaningful groups or improve retrieval systems by identifying semantically related content based on learned embeddings.[4]

8.3 Anomaly Detection

Anomaly detection, the identification of rare or unusual data points that deviate significantly from the norm, has strong conceptual links to clustering and representation learning.[73]

- **Connection:** Anomalies can often be characterized as points that do not belong to any well-formed cluster (like noise points identified by DBSCAN [35]) or reside in low-density regions of the feature space. Alternatively, they might form very small, distinct clusters.
- **Deep Learning Role:** Deep learning methods, particularly Autoencoders (AEs), VAEs, and GANs, are widely used for anomaly detection.[47] They are typically trained on normal data to learn a compressed representation or generative model of normality. Anomalies are then detected based on high reconstruction error (the model struggles to reconstruct unseen, abnormal patterns), low likelihood under the learned distribution, or by falling outside the clusters of normal data in the latent space.[52] Deep clustering methods can contribute by explicitly modeling the clusters corresponding to normal behavior, making deviations easier to spot.[7] This synergy makes anomaly detection a key application where representation learning and clustering concepts converge.[7]

8.4 Other Domains

The principles of deep clustering extend to various other fields:

- **Bioinformatics:** Clustering genes based on complex expression patterns across different conditions [17], analyzing protein sequences or motifs [17], or grouping patients based on multi-modal health data.
- **Customer Segmentation:** Identifying distinct groups of customers based on purchasing behavior, website interactions, or demographic data for personalized

marketing or service design.[1]

- **Recommendation Systems:** Grouping users with similar tastes or items with similar characteristics to improve recommendation accuracy.[16]
- **Time Series Analysis:** Clustering temporal data streams, such as sensor readings, financial market data, or physiological signals (e.g., ECG), requires handling the temporal dimension, often through recurrent networks or specialized AE architectures within a deep clustering framework.[62]
- **Data Management:** Emerging applications include using deep clustering for tasks like schema inference (grouping similar table structures), entity resolution (identifying records referring to the same real-world entity), and domain discovery (finding columns with similar semantic types) in large databases.[32]

Across these applications, the common thread is the need to extract meaningful structure from complex, often high-dimensional, unlabeled data. While the final clusters are often the primary goal, the learned representation itself, produced as a byproduct of the deep clustering process, can sometimes be valuable for transfer learning or other downstream tasks [33], subject to the potential trade-off between clustering specificity and representational generality.

9. Conclusion and Future Directions

9.1 Synthesis

This report has explored the multifaceted relationship between clustering algorithms and representation learning. We established that traditional clustering methods fundamentally rely on the quality of the input data representation; their performance is contingent on how well the feature space aligns with their inherent assumptions about cluster structure. Representation learning techniques like PCA, Autoencoders, and Word2Vec serve as crucial pre-processing tools, transforming raw data into lower-dimensional or more semantically meaningful spaces, thereby often enhancing the effectiveness of subsequent clustering. However, the paradigm of Deep Clustering marks a significant advancement by enabling the joint optimization of representation learning and cluster assignment. This allows for the creation of feature spaces explicitly tailored to the task of grouping data, often leading to state-of-the-art performance on complex, high-dimensional unsupervised learning problems.

9.2 Importance of the Synergy

The interplay between discovering groups (clustering) and discovering informative features (representation learning) is undeniable and synergistic. Good representations facilitate better clustering, while the evolving cluster structure can provide valuable

supervisory signals to refine representations, as demonstrated by various deep clustering frameworks (Iterative, Generative, Simultaneous). This joint optimization, despite its inherent complexities and challenges, represents a powerful approach for unlocking insights from the vast amounts of unlabeled data generated in numerous scientific and industrial domains. It moves beyond generic feature extraction towards learning representations with a specific purpose – revealing the underlying group structure in data.

9.3 Open Challenges & Future Directions

Despite substantial progress, deep clustering remains an active area of research with several open challenges and promising future directions:

- **Scalability and Efficiency:** Developing DC algorithms that can efficiently handle truly massive datasets (billions of data points) remains a challenge, requiring innovations in optimization, distributed training, and potentially sub-sampling or approximation techniques.[38]
- **Robustness and Stability:** Improving the robustness of DC methods to hyperparameter choices, initialization, and noise, while mitigating the risk of convergence to degenerate solutions (e.g., cluster collapse), is crucial for practical deployment.[59]
- **Interpretability and Explainability:** Deep clustering models often function as black boxes. Developing methods that provide insights into why certain clusters are formed or what the learned features represent is essential for building trust and facilitating scientific discovery.[24] Techniques like visualizing cluster assignments across layers [68] or analyzing learned prototypes offer potential avenues.
- **Meaningful Evaluation:** Moving beyond standard clustering metrics applied to labeled benchmarks is necessary. Developing robust evaluation protocols for truly unsupervised settings, potentially assessing representation quality, stability, transferability, or alignment with domain-specific knowledge, remains an important goal.[21]
- **Broader Applicability:** Extending and rigorously evaluating DC methods on a wider range of complex data types (e.g., graphs [19], heterogeneous data, streaming data, time series [74]) and real-world data management tasks [32] is needed to demonstrate their practical utility beyond standard benchmarks.
- **Theoretical Foundations:** Further developing the theoretical understanding of deep clustering, including convergence guarantees, the properties of learned representations, and the conditions under which joint optimization provides significant benefits over sequential approaches, will strengthen the field.[51]

- **Incorporating Domain Knowledge and Interaction:** Exploring frameworks that can effectively incorporate partial supervision, domain constraints, multi-view information [23], transfer learning from related tasks [23], or even interactive user feedback [10] into the joint learning process holds promise for creating more tailored and effective clustering solutions.

In conclusion, the fusion of clustering and representation learning, particularly through deep learning, has significantly advanced our ability to analyze complex unlabeled data. While challenges remain, the ongoing research promises further breakthroughs, leading to more powerful, robust, and interpretable unsupervised learning systems applicable across a vast spectrum of scientific and technological endeavors.

Works cited

1. DBSCAN vs. K-Means: A Guide in Python - New Horizons, accessed April 28, 2025, https://www.newhorizons.com/resources/blog/dbscan-vs-kmeans-a-guide-in-python
2. K-Means Clustering Algorithm - Analytics Vidhya, accessed April 28, 2025, https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/
3. What is k-means clustering? - IBM, accessed April 28, 2025, https://www.ibm.com/think/topics/k-means-clustering
4. Autoencoder - Wikipedia, accessed April 28, 2025, https://en.wikipedia.org/wiki/Autoencoder
5. Introduction to autoencoders. - Jeremy Jordan, accessed April 28, 2025, https://www.jeremyjordan.me/autoencoders/
6. Comparing DBSCAN, k-means, and Hierarchical Clustering: When and Why To Choose Density-Based Methods | Hex, accessed April 28, 2025, https://hex.tech/blog/comparing-density-based-methods/
7. [2206.07579] A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions - ar5iv, accessed April 28, 2025, https://ar5iv.labs.arxiv.org/html/2206.07579
8. DBSCAN Clustering Algorithm in Machine Learning - KDnuggets, accessed April 28, 2025, https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html
9. A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions, accessed April 28, 2025, https://mn.cs.tsinghua.edu.cn/xinwang/PDF/papers/2024_A%20Comprehensive%20Survey%20on%20Deep%20Clustering%20Taxonomy%20Challenges%20and%20Future%20Directions.pdf
10. Personalized Clustering via Targeted Representation Learning - arXiv, accessed

April 28, 2025, https://arxiv.org/html/2412.13690v1
11. A Survey on Deep Learning Based Clustering Methods and Taxonomy for Deep Clustering - AIP Publishing, accessed April 28, 2025, https://pubs.aip.org/aip/acp/article-pdf/doi/10.1063/5.0175632/18194786/050003_1_5.0175632.pdf
12. Introduction to K-means Clustering - Oracle Blogs, accessed April 28, 2025, https://blogs.oracle.com/ai-and-datascience/post/introduction-to-k-means-clustering
13. Exploring Clustering Algorithms: Explanation and Use Cases - neptune.ai, accessed April 28, 2025, https://neptune.ai/blog/clustering-algorithms
14. K-Means Clustering Explained - neptune.ai, accessed April 28, 2025, https://neptune.ai/blog/k-means-clustering
15. How to Master the Popular DBSCAN Clustering Algorithm for Machine Learning?, accessed April 28, 2025, https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/
16. A Guide to the DBSCAN Clustering Algorithm - DataCamp, accessed April 28, 2025, https://www.datacamp.com/tutorial/dbscan-clustering-algorithm
17. Deep learning-based clustering approaches for bioinformatics - Oxford Academic, accessed April 28, 2025, https://academic.oup.com/bib/article/22/1/393/5721075
18. Representation Learning: Unlocking the Hidden Structure of Data - viso.ai, accessed April 28, 2025, https://viso.ai/deep-learning/representation-learning/
19. Feature learning - Wikipedia, accessed April 28, 2025, https://en.wikipedia.org/wiki/Feature_learning
20. [2412.03471] Cluster Specific Representation Learning - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2412.03471
21. Cluster Specific Representation Learning - arXiv, accessed April 28, 2025, https://arxiv.org/html/2412.03471v1
22. What is Embedding? - Embeddings in Machine Learning Explained - AWS, accessed April 28, 2025, https://aws.amazon.com/what-is/embeddings-in-machine-learning/
23. Deep Clustering: A Comprehensive Survey - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2210.04142
24. Representation Learning Pre-training, accessed April 28, 2025, https://courses.cs.washington.edu/courses/cse543/23au/schedule/lecture14.pdf
25. Principal Component Analysis and Autoencoders - Texas A&M University, accessed April 28, 2025, https://people.tamu.edu/~sji/classes/PCA.pdf
26. What Is an Autoencoder? - IBM, accessed April 28, 2025, https://www.ibm.com/think/topics/autoencoder
27. Word2vec - Wikipedia, accessed April 28, 2025, https://en.wikipedia.org/wiki/Word2vec
28. Word2Vec Explained: How Does It Work? - Swimm, accessed April 28, 2025, https://swimm.io/learn/large-language-models/what-is-word2vec-and-how-does-it-work
29. A Beginner's Guide to Word2Vec and Neural Word Embeddings | Pathmind,

accessed April 28, 2025, http://wiki.pathmind.com/word2vec
30. Word2Vec Explained - Hacker's Blog, accessed April 28, 2025, https://israelg99.github.io/2017-03-23-Word2Vec-Explained/
31. Unsupervised Deep Embedding for Clustering Analysis - Proceedings of Machine Learning Research, accessed April 28, 2025, https://proceedings.mlr.press/v48/xieb16.pdf
32. Deep Clustering for Data Cleaning and Integration - OpenProceedings.org, accessed April 28, 2025, https://openproceedings.org/2024/conf/edbt/paper-101.pdf
33. Improving Representation Learning for Histopathologic Images with Cluster Constraints - CVF Open Access, accessed April 28, 2025, https://openaccess.thecvf.com/content/ICCV2023/papers/Wu_Improving_Representation_Learning_for_Histopathologic_Images_with_Cluster_Constraints_ICCV_2023_paper.pdf
34. Learn The Big Picture: Representation Learning for Clustering - ACL Anthology, accessed April 28, 2025, https://aclanthology.org/2021.repl4nlp-1.15.pdf
35. DBSCAN Clustering in ML | Density based clustering - GeeksforGeeks, accessed April 28, 2025, https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/
36. What is k-means clustering? | Machine Learning - Google for Developers, accessed April 28, 2025, https://developers.google.com/machine-learning/clustering/kmeans/overview
37. The framework of our deep embedded clustering with data augmentation... - ResearchGate, accessed April 28, 2025, https://www.researchgate.net/figure/The-framework-of-our-deep-embedded-clustering-with-data-augmentation-DEC-DA-The_fig2_327881284
38. Deep clustering framework review using multicriteria evaluation - Free, accessed April 28, 2025, http://ser.gui.free.fr/homepage/pdf/ros24deep.pdf
39. What Is Principal Component Analysis (PCA)? - IBM, accessed April 28, 2025, https://www.ibm.com/think/topics/principal-component-analysis
40. Principal Component Analysis (PCA) Explained | Built In, accessed April 28, 2025, https://builtin.com/data-science/step-step-explanation-principal-component-analysis
41. DBSCAN Clustering Algorithm Demystified - Built In, accessed April 28, 2025, https://builtin.com/articles/dbscan
42. DBSCAN - Wikipedia, accessed April 28, 2025, https://en.wikipedia.org/wiki/DBSCAN
43. (PDF) Representation Learning for Clustering via Building Consensus - ResearchGate, accessed April 28, 2025, https://www.researchgate.net/publication/351342589_Representation_Learning_for_Clustering_via_Building_Consensus
44. What Is Principal Component Analysis (PCA) and How It Is Used? - Sartorius, accessed April 28, 2025, https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-co

mponent-analysis-pca-and-how-it-is-used-507186
45. setosa.io, accessed April 28, 2025, https://setosa.io/ev/principal-component-analysis/#:~:text=Principal%20component%20analysis%20(PCA)%20is,easy%20to%20explore%20and%20visualize.
46. Principal component analysis: a review and recent developments - PMC - PubMed Central, accessed April 28, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC4792409/
47. Introduction to Autoencoders: From The Basics to Advanced Applications in PyTorch, accessed April 28, 2025, https://www.datacamp.com/tutorial/introduction-to-autoencoders
48. Intro to Autoencoders | TensorFlow Core, accessed April 28, 2025, https://www.tensorflow.org/tutorials/generative/autoencoder
49. Deep Embedding Clustering Driven by Sample Stability - IJCAI, accessed April 28, 2025, https://www.ijcai.org/proceedings/2024/0426.pdf
50. Joint optimization of an autoencoder for clustering and embedding - Semantic Scholar, accessed April 28, 2025, https://www.semanticscholar.org/paper/Joint-optimization-of-an-autoencoder-for-clustering-Boubekki-Kampffmeyer/3dc862110b6303619e213d85e6bbed635c7fdb11
51. [2012.03740] Joint Optimization of an Autoencoder for Clustering and Embedding - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2012.03740
52. Explore Image Anomaly Detection with Deep Learning - RidgeRun.ai, accessed April 28, 2025, https://www.ridgerun.ai/post/a-journey-into-image-anomaly-detection-with-deep-learning
53. Word Embedding: Word2Vec Explained - KNIME, accessed April 28, 2025, https://www.knime.com/blog/word-embedding-word2vec-explained
54. Improved Deep Embedded Clustering with Local Structure Preservation - IJCAI, accessed April 28, 2025, https://www.ijcai.org/proceedings/2017/0243.pdf
55. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization - CVF Open Access, accessed April 28, 2025, https://openaccess.thecvf.com/content_ICCV_2017/papers/Dizaji_Deep_Clustering_via_ICCV_2017_paper.pdf
56. Joint optimization of an autoencoder for clustering and embedding Boubekki, Ahcène; Kampffmeyer, Michael - Leuphana Universität Lüneburg, accessed April 28, 2025, http://fox.leuphana.de/portal/files/21975611/repo_18008743_oa_by.pdf
57. [2210.04142] Deep Clustering: A Comprehensive Survey - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2210.04142
58. [2206.07579] A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2206.07579
59. How does deep clustering relate to self-supervised learning? - Milvus Blog, accessed April 28, 2025, https://blog.milvus.io/ai-quick-reference/how-does-deep-clustering-relate-to-selfsupervised-learning

60. A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions | Request PDF - ResearchGate, accessed April 28, 2025, https://www.researchgate.net/publication/384994189_A_Comprehensive_Survey_on_Deep_Clustering_Taxonomy_Challenges_and_Future_Directions
61. Autoencoded UMAP-Enhanced Clustering for Unsupervised Learning - arXiv, accessed April 28, 2025, https://arxiv.org/html/2501.07729v1
62. Research on load clustering algorithm based on variational autoencoder and hierarchical clustering | PLOS One, accessed April 28, 2025, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0303977
63. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments, accessed April 28, 2025, https://proceedings.neurips.cc/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf
64. DeepCluster Explained | Papers With Code, accessed April 28, 2025, https://paperswithcode.com/method/deepcluster
65. Deep Clustering via Community Detection - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2501.02036
66. Exploring Non-Contrastive Representation Learning for Deep Clustering - OpenReview, accessed April 28, 2025, https://openreview.net/forum?id=JZrETJlgyq
67. Clustering-friendly Representation Learning for Enhancing Salient Features - arXiv, accessed April 28, 2025, https://arxiv.org/html/2408.04891v1
68. Neural Clustering based Visual Representation Learning - CVF Open Access, accessed April 28, 2025, https://openaccess.thecvf.com/content/CVPR2024/papers/Chen_Neural_Clustering_based_Visual_Representation_Learning_CVPR_2024_paper.pdf
69. [2403.17409] Neural Clustering based Visual Representation Learning - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2403.17409
70. Assessing Representation Learning and Clustering Algorithms for Computer-Assisted Image Annotation—Simulating and Benchmarking MorphoCluster - MDPI, accessed April 28, 2025, https://www.mdpi.com/1424-8220/22/7/2775
71. Learning Embedding Space for Clustering From Deep Representations - YouTube, accessed April 28, 2025, https://www.youtube.com/watch?v=d8clWY7Y1-o
72. Full article: Deep Clustering of Remote Sensing Scenes through Heterogeneous Transfer Learning - Taylor & Francis Online, accessed April 28, 2025, https://www.tandfonline.com/doi/full/10.1080/01431161.2025.2465917?af=R
73. Deep Learning for Anomaly Detection, accessed April 28, 2025, https://ff12.fastforwardlabs.com/
74. End-to-end deep representation learning for time series clustering: a comparative study - Germain Forestier, accessed April 28, 2025, https://germain-forestier.info/publis/dmkd2021.pdf
75. Interpretable Clustering: A Survey - arXiv, accessed April 28, 2025, https://arxiv.org/html/2409.00743v1
76. Interpretable Clustering: A Survey - arXiv, accessed April 28, 2025,

https://arxiv.org/pdf/2409.00743