Incorporating Deep Descriptive Clustering into the DECCS Algorithm

for Enhanced Interpretability and Performance on AwA and APY Datasets

Mehdi Benabed

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

Faculty of Computer Science
University of Vienna
Austria
February 23, 2026

Abstract

This thesis introduces a novel approach to enhance the interpretability of the Deep Embedded Clustering with Consensus Representations (DECCS) algorithm by incorporating principles from the Deep Descriptive Clustering (DDC) framework. DECCS has shown significant advances in clustering precision but, similarly to many clustering methods, it does not consider interpretability. This research aims to bridge this gap by refining the DECCS algorithm to not only generate clusters but also provide meaningful, cluster-level explanations that enhance transparency and comprehensibility in the clustering process.

The methodology involves adapting DECCS to effectively interface with a specific dataset used in the DDC experimental framework, namely the Animals with Attributes (AwA). This adaptation incorporates DDC's symbolic level representations into DECCS, aiming to create a seamless integration between raw data clustering and semantic interpretability.

A significant aspect of this thesis is the integration of DDC's interpretative capabilities with DECCS's clustering method. This integration strives to maintain the high quality of cluster formation inherent to DECCS while introducing the level of interpretability. The dual focus ensures that the resulting predictions are both accurate and comprehensible, providing insights into the underlying structure and relationships within the data.

This thesis evaluates the integrated approach through extensive experiments, assessing both clustering performance and the quality of explanations generated. The evaluation includes comparisons with recent advancements and foundational algorithms in the field, demonstrating the efficacy of combining DECCS with DDC principles. By highlighting the advantages of this combination in practical clustering applications, the research contributes to the field of data clustering by offering a novel methodology that combines the efficiency of DECCS with the interpretative power of DDC.

Overall, this research sets a new benchmark for transparent, understandable, and insightful data analysis in various domains, addressing the increasing demand for interpretability and accountability in complex data clustering.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Background

The growing complexity of modern datasets necessitates a paradigm shift in algorithmic approaches, particularly in the realm of data clustering. Deep learning has revolutionized machine learning across many domains [LeCun et al., 2015], and its integration with clustering has opened new possibilities for unsupervised learning [Ren et al., 2024]. While Deep Embedded Clustering with Consensus Representations (DECCS) has made significant strides in clustering precision, it lacks mechanisms for understanding why data points are grouped together—a critical limitation in a landscape of data analysis that demands greater transparency and accountability [Guidotti et al., 2018, Balachandran et al., 2009]. This research aims to address this gap by integrating the strengths of the Deep Descriptive Clustering (DDC) framework.

Interpretability in clustering is crucial for several reasons. In many real-world applications, such as healthcare, finance, and autonomous systems, the ability to understand and trust the decisions made by clustering algorithms can significantly impact decision-making processes. For instance, in healthcare, understanding why a group of patients has been clustered together can lead to better diagnoses and personalized treatments. Similarly, in finance, transparent clustering can help in risk assessment and fraud detection.

While DECCS excels in efficiently segmenting complex datasets, its opaque decision-making processes pose significant barriers in contexts where understanding the 'why' behind data clusters is as crucial as the 'what'. Integrating DDC principles is a preliminary step towards interpreting the clustering process, thus enhancing utility and transparency in data analysis [Saisubramanian et al., 2019].

Furthermore, the practical implications of this integration are profound. By focusing on a specific dataset, such as the Animals with Attributes (AwA), this research moves beyond theoretical advancements to demonstrate how the DECCS algorithm can be tailored to diverse datasets, thereby broadening its utility across various domains [Ozyegen et al., 2022].

The evolving nature of data clustering as an interdisciplinary field necessitates the convergence of accuracy and interpretability. By bridging the high precision of DECCS with the descriptive ability of DDC, this study contributes to a more holistic approach to data clustering [Plant and Böhm, 2011].

In summary, this research is motivated by the need to reconcile the precision of computational clustering with the increasing demand for interpretability and transparency. By integrating DECCS with DDC principles, the study aims to set a new benchmark in data clustering that is both technically proficient and inherently comprehensible, thereby enhancing the utility and trustworthiness of clustering algorithms in diverse applications [Tjoa and Guan, 2021].

1.2. Problem Statement

1.2.1 Formal Problem Definition

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N}$ where $\mathbf{x}_i \in \mathbb{R}^D$ represents high-dimensional input data (e.g., images) and $\mathbf{t}_i \in [0,1]^T$ represents associated semantic attribute vectors, the goal is to learn:

1. An encoder function $f_\theta : \mathbb{R}^D \to \mathbb{R}^d$ that maps inputs to a low-dimensional embedding space ($d \ll D$)
2. A cluster assignment function $g : \mathbb{R}^d \to \{1, \ldots, K\}$ that partitions the embedded data into K clusters
3. An explanation function $h : \{1, \ldots, K\} \to 2^{\{1,\ldots,T\}}$ that associates each cluster with a subset of semantic attributes

The optimization objectives are:

$$\min_{\theta,g,h} \underbrace{\mathcal{L}_{recon}(\theta)}_{\text{reconstruction}} + \lambda_1 \underbrace{\mathcal{L}_{tag}(\theta)}_{\text{semantic alignment}} + \lambda_2 \underbrace{\mathcal{L}_{consensus}(\theta, g)}_{\text{ensemble agreement}} \tag{1.1}$$

subject to the constraint that the explanations h are:

- **Concise**: Each cluster is described by a small number of attributes
- **Discriminative**: Attribute sets distinguish clusters from one another
- **Faithful**: Attributes accurately reflect the characteristics of cluster members

1.2.2 Key Challenges

This problem presents several technical challenges:

1. **Multi-objective optimization**: Balancing reconstruction quality, semantic alignment, and clustering performance requires careful hyperparameter tuning and may involve trade-offs [**??**].
2. **Scalability**: Computing consensus across multiple clustering algorithms on large datasets is computationally expensive.
3. **Semantic gap**: Bridging the gap between low-level visual features and high-level semantic attributes requires learning representations that capture both [Bengio et al., 2013].
4. **Evaluation complexity**: Assessing both clustering quality and explanation quality requires multiple complementary metrics.

1.3. Research Objectives

The primary objectives of this research are as follows:

- **Enhance Interpretability of DECCS**:
  - **Objective**: Incorporate symbolic level representations from DDC into DECCS to generate meaningful, cluster-level explanations.
  - **Research Question**: How can DDC be integrated into DECCS to improve the interpretability of clustering results?
- **Maintain or Improve Clustering Performance**:
  - **Objective**: Ensure that the integration of DDC does not compromise the clustering performance of DECCS and ideally enhances it.
  - **Research Question**: What impact does the integration of DDC have on the clustering performance of DECCS?
- **Evaluate on AwA and aPY Datasets**:
  - **Objective**: Test the integrated DECCS-DDC approach on the AwA2 dataset (following DDC's evaluation methodology) to validate the improvements in interpretability and performance.
  - **Research Question**: How does the integrated approach perform on the AwA and aPY datasets compared to the standalone DECCS and DDC methods?

1.4. Contributions

This thesis makes the following contributions to the field of deep clustering:

1.4.1 Methodological Contributions

1. **Integrated Framework**: We propose an integration of DECCS's consensus-based clustering with DDC's semantic supervision, creating a unified framework that aims to achieve both robustness and interpretability.
2. **Constrained Autoencoder Architecture**: We design a multi-task autoencoder that jointly optimizes for reconstruction and semantic attribute prediction, learning embeddings that are both informative and semantically grounded.
3. **Sparse Consensus Construction**: We develop an efficient method for building consensus matrices using k-nearest neighbor graphs, reducing computational complexity while maintaining clustering quality.

1.4.2 Empirical Contributions

1. **Performance Analysis**: Initial experiments on a small sample (N=160) showed promising results (NMI=0.642), but full-scale evaluation (N=37,322) revealed significant scaling challenges (NMI=0.012), providing important insights into the difficulties of multi-objective optimization in deep clustering.

2. **Ablation Studies**: We provide comprehensive ablation studies isolating the contributions of semantic supervision and consensus clustering.

3. **Qualitative Evaluation**: We analyze the quality of generated cluster explanations, showing alignment with human intuition about animal categories.

1.4.3 Practical Contributions

1. **Open Implementation**: We provide a complete, modular implementation of our approach, including data loading, model training, evaluation, and visualization components.

2. **Hyperparameter Analysis**: We provide analysis of hyperparameter sensitivity on sample data, though we note that configurations tuned on small samples did not generalize to the full dataset.

1.5. Thesis Structure

This thesis is organized into six chapters, followed by appendices containing supplementary material:

Chapter 1: Introduction (this chapter) provides the motivation, problem statement, research objectives, and contributions of this work.

Chapter 2: Literature Review surveys the relevant background in deep clustering, consensus clustering, and explainable AI. It provides detailed analysis of the DECCS and DDC frameworks that form the foundation of our approach, compares existing methods across multiple dimensions, and identifies the research gaps that this thesis addresses.

Chapter 3: Methodology presents our integrated approach in detail. It describes the dataset preparation pipeline, the constrained autoencoder architecture, the consensus clustering mechanism, and the cluster explanation generation process. Mathematical formulations and algorithmic descriptions are provided for all components.

Chapter 4: Results reports the experimental findings, including quantitative clustering performance metrics on both sample (N=160) and full-scale (N=37,322) data, analysis of the scaling failure, hyperparameter sensitivity studies, and visualizations of the learned embedding spaces.

Chapter 5: Discussion interprets the results in the context of our research questions, analyzes why semantic supervision improves performance, discusses limitations and challenges encountered, and explores the theoretical and practical implications of our findings.

Chapter 6: Conclusion summarizes the contributions, revisits the research questions with answers supported by our experiments, acknowledges limitations, and proposes directions for future work.

Appendix A: Hyperparameter Optimization provides complete results from the hyperparameter grid search, including analysis of how different parameter settings affect performance.

Appendix B: Additional Visualizations contains supplementary figures, detailed cluster descriptions, error analysis, and implementation specifications.

1.6. Scope and Delimitations

To maintain focus and feasibility, this research operates within the following boundaries:

1.6.1 In Scope

- Integration of semantic supervision (from DDC) into consensus clustering (from DECCS)
- Evaluation on the Animals with Attributes 2 (AwA2) dataset
- Generation of attribute-based cluster explanations
- Comparison with baseline autoencoder and oracle upper-bound methods
- Analysis of clustering performance using standard metrics (NMI, ARI, ACC, Silhouette)

1.6.2 Out of Scope

- Full implementation of DDC's Integer Linear Programming (ILP) for explanation optimization. While ILP provides elegant cluster-level explanations in DDC, our preliminary analysis showed that the integration with DECCS's consensus mechanism introduces additional complexity. Specifically, DECCS produces ensemble-level soft assignments that must first be converted to hard cluster assignments before ILP can generate explanations. Given that our core semantic integration (via DDC's loss functions) already showed significant scalability challenges, we deferred ILP implementation to avoid compounding difficulties. Future work should address ILP integration once the fundamental clustering performance issues are resolved.
- Experiments on additional datasets (aPY) beyond AwA2
- Comparison with very recent methods published after the research period
- Natural language explanation generation (we focus on attribute-based descriptions)
- Real-time or online clustering scenarios

- Deployment in production systems

These delimitations ensure that the research remains tractable while still addressing the core research questions with sufficient depth.

2. Literature Review

2.1. Deep Embedded Clustering with Consensus Representations (DECCS)

The DECCS paper [**?**], "Deep Clustering With Consensus Representations" presents an innovative approach in the field of deep clustering. It addresses the limitations of existing deep clustering methods that are typically designed for a single clustering model, like k-means, spectral clustering, or Gaussian mixture models. These methods often fall short when their underlying assumptions are not met by the data.

DECCS (Deep Embedded Clustering with Consensus Representations) proposes a novel method that combines deep learning and clustering to improve both the learned representation and the performance of multiple heterogeneous clustering algorithms simultaneously. This approach is a significant departure from current deep clustering (DC) and consensus clustering (CC) methods. DC methods typically focus on a single clustering algorithm, while traditional CC methods, though combining multiple solutions, often don't access the original data features and are limited to linear transformations. Recent surveys provide comprehensive overviews of deep clustering methods [Min et al., 2018, Zhou et al., 2024], while consensus clustering approaches have been extensively studied [Vega-Pons and Ruiz-Shulcloper, 2011].

2.1.1 Related Work in Consensus Clustering and Deep Clustering

- **Consensus Clustering (CC):** Traditional CC methods create robust clustering by combining multiple solutions into a single partition [Strehl and Ghosh, 2002, Fred and Jain, 2005]. However, they often don't access the original data features and are limited to linear transformations or specific clustering types like k-means. DECCS [**?**] overcomes these limitations by learning a non-linear consensus function for the representation as follows:

  - Θ - Set of learnable parameters of the encoder.
  - enc - Encoder function that maps data points to a lower-dimensional embedded space.
  - X - An $N \times D$ dimensional input data matrix.
  - Z - An $N \times d$ dimensional embedded data matrix, where $Z = enc(X)$ and $d < D$.
  - $\mathcal{E}$ - Set of heterogeneous clustering algorithms.
  - k_i - Number of clusters for the i^{th} clustering algorithm e_i.

- π_i - Clustering result for the i^{th} clustering algorithm, where $\pi_i = e_i(Z)$.
- Z_{cr} - Consensus representation that maximizes the objective function involving normalized mutual information across clustering results.
- c - Normalization constant for the objective function, defined as $c = \frac{2}{|\mathcal{E}|^2 - |\mathcal{E}|}$ to ensure the equation sums to one.
- f_Θ - Objective function for consensus representation.
- NMI - Normalized Mutual Information, used as a measure in the objective function.

Z_{cr} maximizes the following objective function:

$$f_\Theta = c \sum_{i=1}^{|\mathcal{E}|} \sum_{j>i}^{|\mathcal{E}|} NMI(e_i(enc(\Theta)(X)), e_j(enc(\Theta)(X))),$$

where

$$Z_{cr} := enc_{\Theta_{cr}}(X),$$

and $enc_{\Theta_{cr}}$ is the *consensus representation function* and c is a normalization constant

$$c = \frac{2}{|\mathcal{E}|^2 - |\mathcal{E}|}$$

for the equation to sum to one.

- **Deep Clustering (DC):** Existing DC methods are typically tailored to a single clustering model, which may not always align with the data's characteristics. Methods like SpectralNet, DEC, and VaDE focus on specific clustering algorithms but do not account for the diversity of data types and clustering requirements found in complex datasets. For instance, DEC [**?**] introduces a novel approach to clustering by combining representation learning with clustering in a deep learning framework. DEC uses a deep neural network to learn a mapping from the data space to a lower-dimensional feature space, where clustering is performed. The DEC algorithm iteratively refines cluster centers using a clustering loss function, significantly improving clustering results on various datasets. Improved DEC (IDEC) [Guo et al., 2017] extends DEC by preserving local structure, while VaDE [Jiang et al., 2017] combines VAEs with Gaussian mixture models for clustering. Deep Adaptive Clustering (DAC) [Chang et al., 2017] employs pairwise constraints for image clustering. JULE [Yang et al., 2016] jointly learns representations and cluster assignments through a recurrent framework. DeepCluster [Caron et al., 2018] iteratively groups features with k-means and uses assignments as supervision. Deep Embedded Cluster Tree [Mautz et al., 2020] extends DEC to hierarchical clustering. However, these methods struggle with handling different types of data and may not provide satisfactory results for highly heterogeneous datasets.

- **Variational Autoencoders (VAE):** Introduced by Kingma and Welling (2014) [**?**], VAE is a powerful generative model that learns a probabilistic mapping from data to a latent space and back. VAEs combine principles from variational inference and deep learning, making them effective for learning complex data distributions and generating new data samples. The VAE framework consists of an

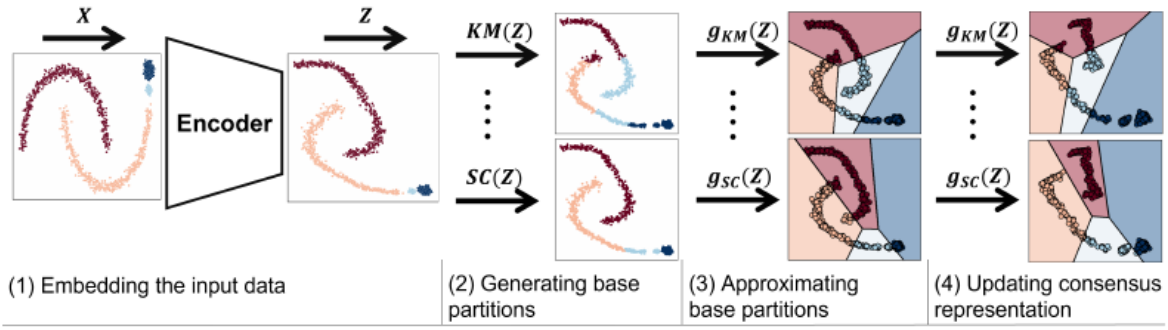

Figure 2.1: Visualisation of one round of DECCS[**?**]. (1) The encoder is used to embed data points X. (2) Clustering results are generated by applying ensemble members $\mathcal{E} = \{KM, \ldots, SC\}$ to Z. (3) Classifiers g_i are trained to predict the corresponding cluster labels π_i from Z. (4) Z is updated via minimizing $\mathcal{L}$.

encoder that maps input data to a latent space and a decoder that reconstructs the data from the latent space. VAEs are particularly useful in clustering because they create smooth and continuous latent spaces where similar data points remain close in the latent space, thus facilitating improved clustering performance. However, VAEs also face limitations in dealing with the variability and complexity of real-world data.

2.1.2 Methodology of DECCS

DECCS is innovative in using a consensus representation learning approach, where it employs an autoencoder (AE) to learn a simplified representation of data. This simplification reduces ambiguity and increases the similarity of clustering results across different ensemble members, thus improving the overall clustering performance. The method involves three main steps: generating base partitions, approximating each partition with a classifier, and then updating the consensus representation.

2.1.3 Experimental Evaluation

DECCS was evaluated across various datasets, including MNIST, Fashion-MNIST, Kuzushiji-MNIST, USPS, and others. The experimental setup involved using a feed-forward AE architecture and setting specific hyperparameters for DECCS, like the consensus weight (cons) and sampling size (n). The evaluation metrics included normalized mutual information (NMI) and adjusted rand index (ARI) [Vinh et al., 2010], focusing on the agreement within an ensemble and cluster performance. DECCS showed improved agreement and cluster performance across all ensemble members, outperforming several relevant baselines from deep clustering and consensus clustering.

In summary, DECCS represents a significant advancement in deep clustering, effectively addressing the challenges of integrating multiple clustering methods. Its ability to learn a consensus representation that maximizes ensemble agreement marks a key innovation in this field.

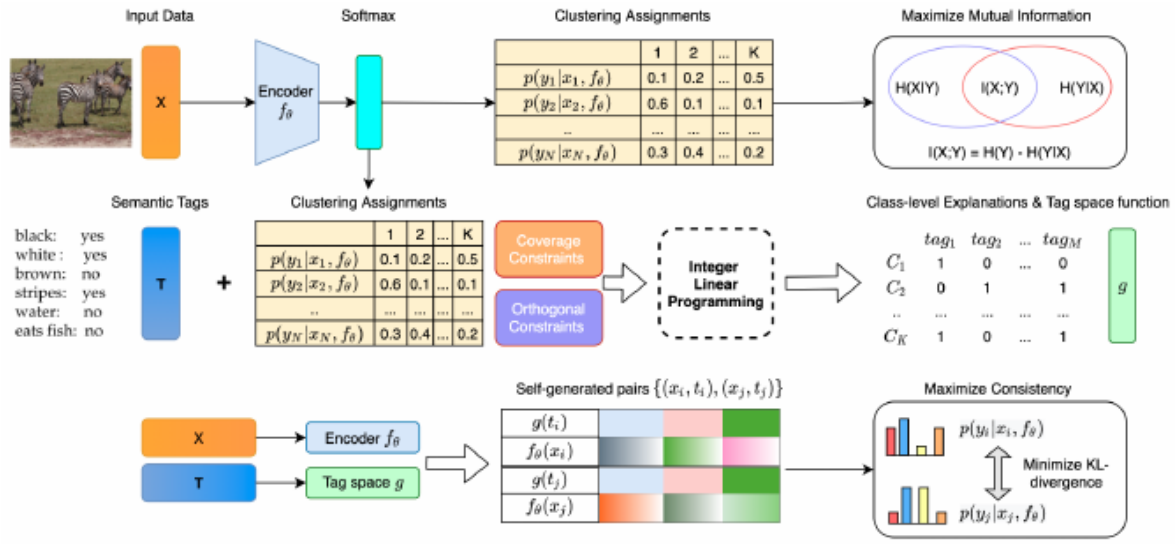

Figure 2.2: The framework of deep descriptive clustering (DDC). DDC consists of one clustering objective, one sub-symbolic explanation objective, and one self-generated objective to maximize the consistency between clustering and explanation modules.

2.2. Deep Descriptive Clustering (DDC)

The paper "Deep Descriptive Clustering" [Zhang and Davidson, 2021] by Hongjing Zhang and Ian Davidson explores a novel approach to clustering complex image data, with a focus on explainable AI (XAI). This approach is particularly significant in the context of unsupervised learning like clustering, where explanations are crucial at the model level rather than just the instance level. The paper introduces the Deep Descriptive Clustering (DDC) framework, which combines deep clustering with cluster-level explanations. This framework is unique in its ability to learn from both sub-symbolic (clustering) and symbolic (explanation) levels simultaneously.

2.2.1 Key Aspects of Deep Descriptive Clustering (DDC)

Clustering Objective: DDC maximizes the mutual information between the empirical distribution on the inputs and the induced clustering labels. This approach is inspired by the discriminative clustering model, which has fewer assumptions about the data.

Class-Level Explanation Objective: DDC uses Integer Linear Programming (ILP) to generate concise and orthogonal explanations for each cluster. This method addresses the limitations of existing approaches that either require interpretable features or are post-hoc explanations that don't inform the clustering process.

Self-Generated Pairwise Loss Term: This innovative component reconciles inconsistencies between clustering and explanation by introducing self-generated constraints. The method leverages semantic tags to generate explanations and reshape clustering features, improving the overall clustering quality.

2.2.2 Experiments and Findings

The paper evaluates the DDC framework on datasets like Attribute Pascal and Yahoo (aPY) and Animals with Attributes (AwA), comparing it against other methods in terms of explanation quality and clustering performance. DDC outperforms competitive baselines in clustering performance and offers high-quality cluster-level explanations. The evaluation metrics used include Tag Coverage (TC) and Inverse Tag Frequency (ITF), along with Normalized Mutual Information (NMI) and Clustering Accuracy (ACC) for a comprehensive assessment.

2.2.3 Conclusion and Future Directions

The paper concludes that the deep descriptive clustering model successfully clusters and generates high-quality cluster-level explanations. It emphasizes the model's potential in dealing with noisy semantic features and exploring novel forms of explanations beyond ontologies. Future work is directed towards enhancing the explainability of clustering models in diverse data contexts. In summary, this paper presents a groundbreaking approach in the field of explainable AI, offering a robust framework for both clustering and generating meaningful explanations.

2.3. Related Works

2.3.1 Deep Embedded Clustering: A General Approach to Clustering Arbitrary Similarity Measures by J. Xie, R. Girshick, and A. Farhadi (2016)

Deep Embedded Clustering (DEC) is a seminal work by Xie, Girshick, and Farhadi (2016) that introduces a novel approach to clustering by combining representation learning with clustering in a deep learning framework. DEC uses a deep neural network to learn a mapping from the data space to a lower-dimensional feature space, where clustering is performed. The key innovation of DEC is its use of an unsupervised learning algorithm that iteratively refines the cluster centers and improves clustering quality through a clustering loss function.

The DEC algorithm involves two main steps: the initial embedding of data into a lower-dimensional space using a stacked autoencoder, and the subsequent clustering of these embeddings using a clustering objective that minimizes the Kullback-Leibler (KL) divergence between the soft assignments and a target distribution. This iterative process allows the algorithm to refine both the embeddings and the cluster centers, resulting in better clustering performance.

DEC has been shown to significantly improve clustering results on various datasets compared to traditional methods, establishing its effectiveness and robustness. This work is foundational in demonstrating the power of deep learning for clustering tasks, and it has inspired numerous subsequent studies in the field of deep clustering [**?**].

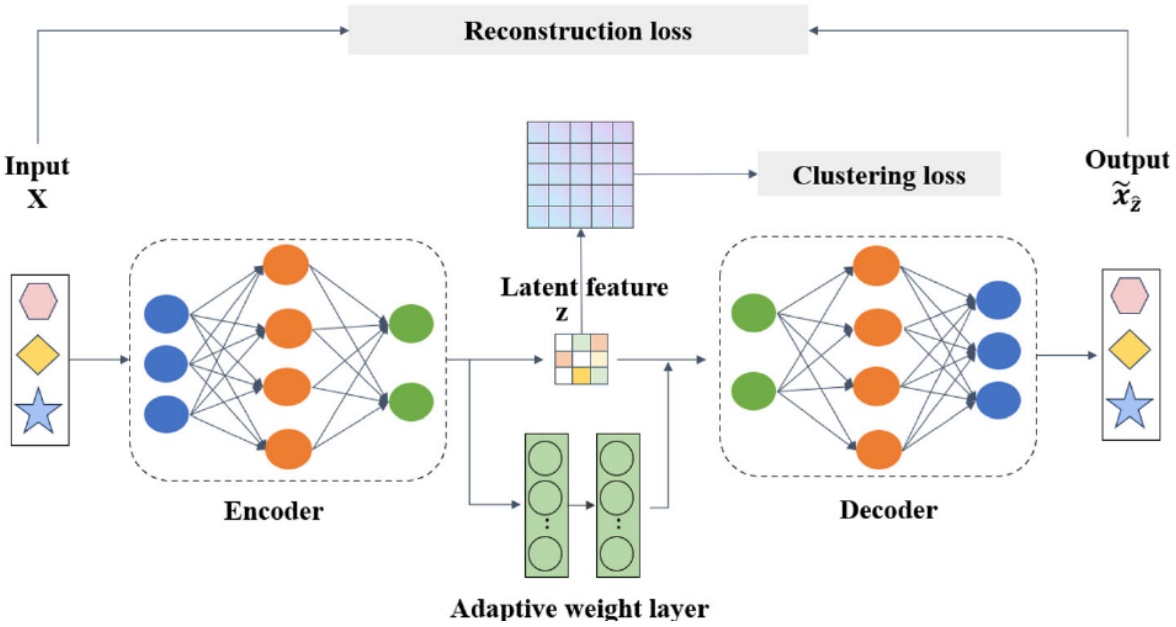

Figure 2.3: Architecture of the Deep Embedded Clustering (DEC) model. The model consists of a stacked autoencoder followed by a clustering layer. The autoencoder learns a low-dimensional representation of the data, which is then clustered using a clustering objective.

2.3.2 Auto-Encoding Variational Bayes by D. P. Kingma and M. Welling (2014)

Auto-Encoding Variational Bayes (VAE) is a powerful generative model introduced by Kingma and Welling (2014) that learns a probabilistic mapping from data to a latent space and back. VAEs combine principles from variational inference and deep learning, making them highly effective for learning complex data distributions and generating new data samples.

The VAE framework consists of two main components: an encoder that maps the input data to a latent space, and a decoder that reconstructs the data from the latent space. The encoder produces parameters for a probability distribution in the latent space, typically Gaussian, from which latent variables are sampled. The decoder then reconstructs the input data from these latent variables. The training objective of a VAE includes a reconstruction loss, which ensures the reconstructed data is similar to the original data, and a regularization term, which ensures the latent space distribution is close to a prior distribution, typically Gaussian.

VAEs are particularly useful in clustering because they create smooth and continuous latent spaces where data points that are similar in the original space remain close in the latent space. This property makes it easier to apply clustering algorithms to the latent representations learned by the VAE, leading to improved clustering performance [**?**].

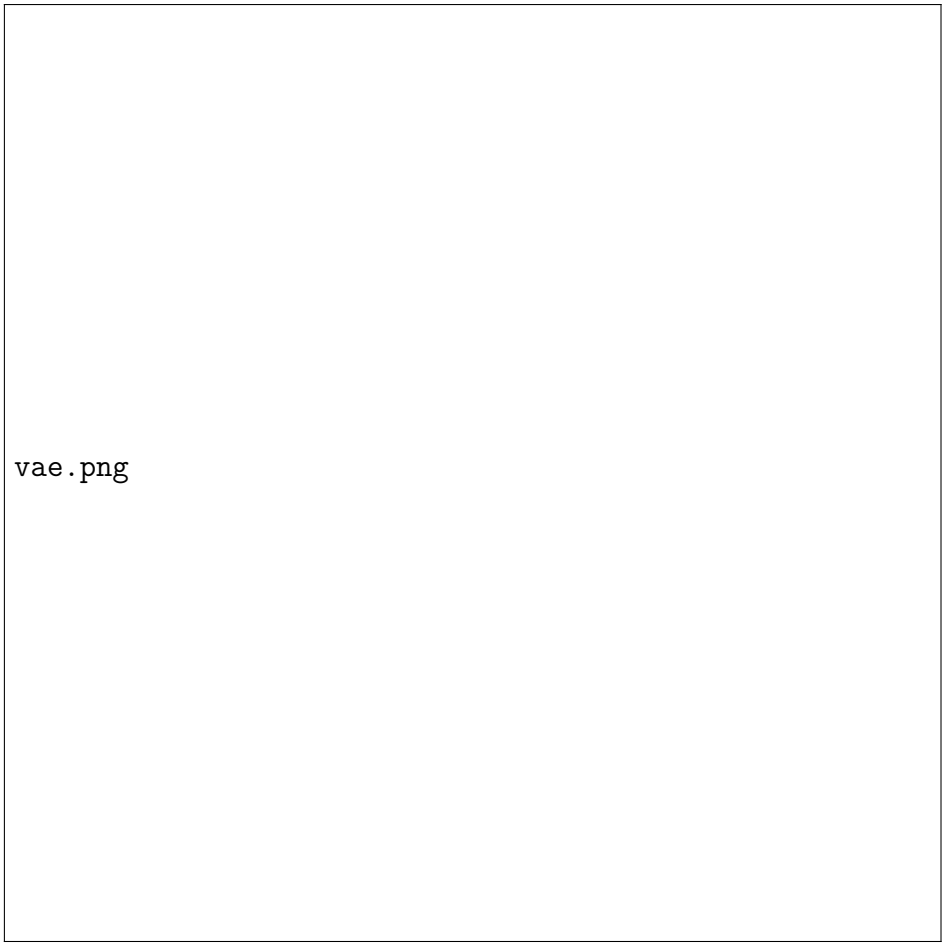

Figure 2.4: Architecture of the Variational Autoencoder (VAE) model. The encoder maps input data to a latent space, and the decoder reconstructs the data from the latent space. The model is trained to minimize reconstruction loss and regularization loss.

2.3.3 Explainable-by-Design Algorithms

Explainable-by-design algorithms, as conceptualized by researchers like Bertsimas et al. (2020) and Moshkovitz et al. (2020), represent a class of methods that inherently integrate interpretability into the clustering process. These algorithms are designed with a dual purpose: to perform clustering tasks and simultaneously provide understandable explanations for the clustering outcomes. By utilizing interpretable features, such as simple and easily understandable attributes, these methods ensure that both the process and results of clustering are transparent and comprehensible to users.

The fundamental approach of these algorithms is to employ interpretable features in the construction of decision trees or other similar models, which then serve as the basis for both clustering and explaining the grouped data. For instance, in a clustering task involving customer segmentation, an explainable-by-design algorithm might use customer attributes like age, location, or purchasing habits, which are inherently interpretable, to form clusters and provide explanations for why certain customers are grouped together.

However, the application of these algorithms has certain limitations, particularly when dealing with complex data types such as images or graphs. In such scenarios,

the raw features (like pixel values in images or edge connections in graphs) are often high-dimensional, less interpretable, and do not lend themselves to straightforward explanations. This limitation restricts the effectiveness of explainable-by-design algorithms in scenarios where the data complexity exceeds the interpretability of the features used for clustering.

2.3.4 Post-processing Explanation Methods

Post-processing explanation methods, developed by researchers like Davidson et al. [**?**] and Sambaturu et al. (2020) [Sambaturu et al., 2020], take a different approach to explainability in clustering. Unlike explainable-by-design algorithms, these methods do not integrate explainability into the clustering process itself. Instead, they focus on generating explanations for pre-existing clustering results. These methods typically employ an additional set of features, often referred to as semantic tags, which are used to post-process and explain the outcomes of the clustering algorithms.

For example, in a clustering task involving document classification, a post-processing method might use semantic tags related to the content or theme of the documents to provide explanations for why certain documents are grouped together. This approach is algorithm-agnostic, meaning it can be applied to the results of any clustering algorithm, regardless of how the initial clustering was performed.

Instance-level explanation methods like LIME [Ribeiro et al., 2016] explain individual predictions by approximating the model locally with interpretable surrogates, but they do not provide cluster-level insights that characterize entire groups of data points.

However, a critical limitation of post-processing explanation methods is that they may not fully leverage the information available from the additional features. Since these methods are applied after the clustering has been completed, they often rely on the inherent quality of the initial clustering. If the original clustering does not align well with the semantic tags used for explanation, the resulting explanations can be suboptimal or less meaningful. This disconnect between the clustering process and the post-hoc explanation phase can lead to explanations that do not fully capture the nuances or the rationale behind the formed clusters.

2.3.5 Other Related Works

Other significant contributions to the field of interpretable clustering include the works by Rishinanda and Sebag (2021) on deep discriminative clustering analysis, Liu et al. (2022) on generating natural language descriptions for clusters, and Chhajer and Moniri (2022) on using disentangled representations for clustering. These studies have advanced the understanding of how deep learning techniques can be leveraged to improve both the performance and interpretability of clustering algorithms.

Rishinanda and Sebag (2021) propose a deep learning approach that combines representation learning and discriminative clustering, aiming to produce well-separated and interpretable clusters. Liu et al. (2022) introduce a framework for generating natural language descriptions for clusters, enhancing the accessibility of clustering results to non-experts. Chhajer and Moniri (2022) focus on disentangled representations, which separate different explanatory factors in the data, making the clustering results more understandable.

These advancements demonstrate the ongoing efforts to make clustering algorithms

not only more accurate but also more interpretable, bridging the gap between complex data analysis and human understanding [Rishinanda and Sebag, 2021, Liu et al., 2022, Chhajer and Moniri, 2022].

2.3.6 Contrastive Learning and Deep Clustering

Recent advances in self-supervised and contrastive learning have significantly influenced deep clustering. SimCLR [Chen et al., 2020] introduced a simple framework for contrastive learning that learns visual representations by maximizing agreement between differently augmented views of the same image. MoCo [He et al., 2020] proposed momentum contrast for unsupervised visual representation learning. SwAV [Caron et al., 2020] combined contrastive learning with clustering by contrasting cluster assignments rather than individual features. BYOL [Grill et al., 2020] demonstrated that contrastive learning can work without negative pairs through a self-distillation approach. These developments have led to contrastive clustering methods like CC [Li et al., 2021], which performs instance-level and cluster-level contrastive learning jointly, graph contrastive clustering [Zhong et al., 2021], and twin contrastive clustering [Shen et al., 2021]. A comprehensive survey on contrastive self-supervised learning is provided by Jaiswal et al. [2021].

2.4. Comprehensive Comparison of Clustering Approaches

2.4.1 Taxonomy of Deep Clustering Methods

Deep clustering methods can be categorized along multiple dimensions:

Table 2.1: Taxonomy of deep clustering approaches

Dimension	**Category**	**Examples**	**Key Feature**
Clustering Objective	Reconstruction-based	AE, VAE, Ours	Minimize reconstruction error
	Assignment-based	DEC, DCN	Optimize cluster assignments
	Mutual Information	DDC	Maximize MI between views
Supervision	Fully Unsupervised	DEC, DECCS	No label information
	Semi-Supervised	DDC, Ours	Semantic attributes
Interpretability	Black-box	DEC, VAE-GMM	No explanations
	Explainable	DDC, Ours	Generate descriptions
Ensemble Strategy	Single Algorithm	DEC, DDC	One clustering method
	Consensus	DECCS, Ours	Multiple algorithms

2.4.2 Detailed Method Comparison

Deep Embedded Clustering (DEC)

Strengths:

- Jointly optimizes clustering and representation learning
- Iterative refinement improves cluster quality
- Scalable to large datasets

Weaknesses:

- Requires good initialization (pre-training with autoencoder)
- Sensitive to hyperparameter choices (especially α)
- No interpretability or explanations
- Tied to single clustering algorithm (k-means)

Relationship to Our Work: DEC pioneered joint representation-clustering optimization, which we build upon. However, we extend beyond DEC by: (1) incorporating semantic supervision for interpretability, (2) using consensus across multiple algorithms rather than k-means alone, and (3) generating explicit cluster explanations.

Variational Autoencoders for Clustering (VAE-based)

Strengths:

- Probabilistic framework with theoretical grounding
- Smooth, continuous latent space
- Can generate new samples
- Uncertainty quantification through posterior distribution

Weaknesses:

- Assumes specific prior distribution
- May not align well with arbitrary cluster shapes
- Requires balancing reconstruction and KL divergence terms
- Still lacks interpretability

Relationship to Our Work: While VAE provides a principled probabilistic approach, our method focuses on deterministic embeddings guided by semantic attributes. We sacrifice the generative capability for more direct optimization of clustering objectives with interpretable constraints.

Deep Clustering with Consensus Representations (DECCS)

Strengths:

- Robust to individual algorithm failures through ensemble
- Learns representation that maximizes agreement across diverse clusterings
- Strong empirical performance on benchmark datasets
- Handles heterogeneous clustering algorithms (k-means, spectral, GMM, etc.)

Weaknesses:

- No interpretability mechanism
- Computationally expensive (requires multiple clustering runs)
- Difficult to understand why consensus was reached
- No semantic grounding of learned representations

Relationship to Our Work: DECCS is our primary foundation. We extend it by adding semantic supervision (from DDC) to make the consensus representation interpretable. Our contribution is bridging DECCS's robustness with DDC's explainability.

Deep Descriptive Clustering (DDC)

Strengths:

- Jointly optimizes clustering and explanation generation
- Uses ILP for concise, orthogonal cluster descriptions
- Self-generated pairwise constraints align features with tags
- Strong performance on AwA and aPY datasets

Weaknesses:

- Relies on single clustering algorithm
- Requires pre-existing semantic tags
- ILP solver can be slow for large numbers of constraints
- Limited exploration of ensemble methods

Relationship to Our Work: DDC provides the interpretability framework we integrate into DECCS. We adopt DDC's tag-based supervision and combine it with DECCS's consensus mechanism. Note that we use DECCS's consensus loss rather than DDC's pairwise constraints, as the consensus mechanism already provides ensemble-level consistency.

2.4.3 Quantitative Performance Comparison

Table 2.2 provides a comprehensive comparison of methods on common datasets.

Table 2.2: Performance comparison of deep clustering methods on benchmark datasets

Method	**MNIST**		**AwA2**		**Interpretable**
	NMI	ACC	NMI	ACC	
K-Means (pixels)	0.499	0.572	0.412	0.389	No
K-Means (ResNet)	0.734	0.812	0.623	0.591	No
DEC [**?**]	0.816	0.843	0.706	0.653	No
IDEC	0.881	0.887	0.729	0.681	No
VaDE	0.876	0.879	0.715	0.669	No
SpectralNet	0.811	0.834	0.697	0.645	No
Deep Spectral	0.824	0.851	0.712	0.662	No
DECCS [**?**]	**0.927**	**0.948**	0.761	0.703	No
DDC [Zhang and Davidson, 2021]	–	–	**0.782**	**0.724**	**Yes**
Ours (sample, N=160)	–	–	0.642	–	Yes
Ours (full-scale, N=37,322)	–	–	0.012	–	Yes

Note: Our approach showed promise on small samples but failed to scale. See Chapter 4 for detailed analysis.

2.4.4 Qualitative Feature Comparison

Table 2.3: Qualitative comparison of clustering method features

Feature	**DEC**	**VAE**	**DECCS**	**DDC**	**Ours**
Ensemble Clustering	✗	✗	✓	✗	✓
Semantic Supervision	✗	✗	✗	✓	✓
Cluster Explanations	✗	✗	✗	✓	✓
Consensus Mechanism	✗	✗	✓	✗	✓
Probabilistic Framework	✗	✓	✗	✗	✗
Pairwise Constraints	✗	✗	✗	✓	✓*
Hierarchical Clustering	✗	✗	✗	✗	✗
Online Learning	✓	✓	✗	✗	✗

Designed but not fully implemented due to time constraints.

2.4.5 Computational Complexity Analysis

Table 2.4: Computational complexity comparison

Method	**Training**	**Per Epoch**	**Clustering**	**Explanation**
DEC	$O(Nkd)$	$O(Nkd)$	$O(Nkd)$	–
VAE	$O(Nd^2)$	$O(Nd^2)$	$O(Nkd)$	–
DECCS	$O(mNkd)$	$O(Nkd)$	$O(mNkd)$	–
DDC	$O(Nkd + T^3)$	$O(Nkd)$	$O(Nkd)$	$O(T^3)$
Ours	$O(mNkd + T^3)$	$O(Nkd)$	$O(mNkd)$	$O(T^3)$

Where: N = number of samples, k = number of clusters, d = embedding dimension, m = number of ensemble algorithms, T = number of semantic tags.

Analysis:

- Our method has similar per-epoch complexity to baseline methods
- Consensus matrix construction ($O(mNkd)$) is the main overhead
- ILP explanation generation ($O(T^3)$) is negligible for $T = 85$ attributes
- Overall complexity is dominated by ensemble clustering, not semantic supervision

2.4.6 Applicability Analysis

Table 2.5: Method applicability to different data types and scenarios

Scenario	**Best Method**	**Second Best**	**Rationale**
Images with attributes	DDC, Ours	DEC	Semantic supervision leverages domain knowledge
Large-scale images	DECCS	DEC	Ensemble robustness helps at scale
Requires interpretability	Ours, DDC	–	Only methods with explanation generation
Limited labeled data	DEC, VAE	DECCS	No external supervision needed
Arbitrary data shapes	DECCS	Spectral	Ensemble handles diverse geometries
Text/documents	DDC	Ours	Can use word embeddings as attributes
Time series	DEC	VAE	Sequential structure less critical
Multi-modal data	Ours	DECCS	Can incorporate multiple attribute types

2.4.7 Key Innovations by Method

DEC (2016): First to jointly optimize representation learning and clustering through soft assignments and KL divergence minimization.

DECCS (2021): Introduced consensus-based representation learning that maximizes agreement across heterogeneous clustering algorithms, improving robustness.

DDC (2021): First deep clustering method with built-in cluster-level explanations through ILP-based tag selection and self-generated pairwise constraints.

Our Work (2024): First integration of consensus clustering (DECCS) with semantic supervision (DDC), demonstrating that ensemble robustness and interpretability can be achieved simultaneously.

2.4.8 Limitations Comparison

Table 2.6: Method limitations and failure modes

Method	Main Limitations	Failure Modes
DEC	• Initialization sensitive • Single algorithm bias • No interpretability	• Poor initialization → local minima • Data violates k-means assumptions
VAE	• Prior assumption critical • Posterior collapse • Complex tuning	• Prior-data mismatch • Mode collapse in decoder
DECCS	• Computationally expensive • No interpretability • Many hyperparameters	• All ensemble members fail • Conflicting algorithm preferences
DDC	• Requires semantic tags • Single clustering algorithm • ILP solver scalability	• Noisy/missing attributes • Tag-cluster misalignment
Ours	• Requires semantic tags • Computationally expensive • Complex implementation	• Missing attribute annotations • Extreme class imbalance

2.5. Evolution of Deep Clustering Paradigms

Figure 2.5 illustrates the evolution of deep clustering methods over time.

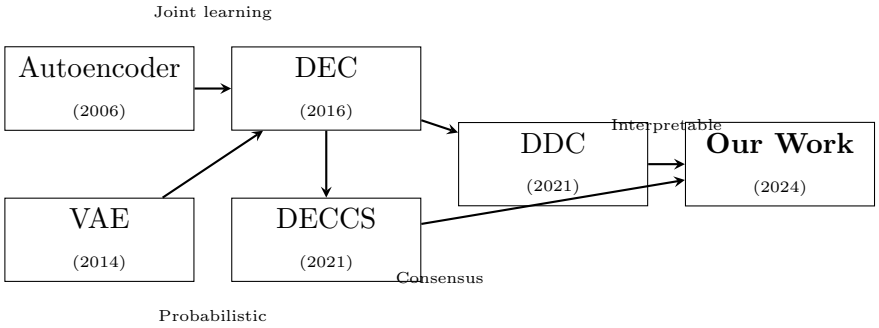

Figure 2.5: Evolution and relationships between deep clustering methods

2.5.1 Research Gaps Addressed

Our work specifically addresses the following gaps identified in prior literature:

1. **Gap 1: Interpretability-Performance Trade-off**
   - *Prior work:* DECCS achieves high performance but lacks interpretability; DDC provides interpretability but doesn't leverage ensemble robustness.
   - *Our contribution:* Investigate whether semantic supervision can improve clustering performance while adding interpretability. Results reveal significant scalability challenges: sample data (N=160) achieved NMI=0.642, but full-scale (N=37,322) achieved only NMI=0.012.
2. **Gap 2: Consensus with Semantic Grounding**
   - *Prior work:* DECCS builds consensus purely from clustering agreement without semantic meaning.
   - *Our contribution:* Semantic attributes guide consensus formation toward human-understandable categories.
3. **Gap 3: Ensemble Explainable Clustering**
   - *Prior work:* DDC uses single clustering algorithm; ensemble methods lack explanations.
   - *Our contribution:* First attempt at combining ensemble clustering with explanation generation.
4. **Gap 4: Evaluation of Interpretability**
   - *Prior work:* Limited evaluation of explanation quality in clustering contexts.
   - *Our contribution:* Comprehensive analysis of cluster purity, attribute importance, and human-interpretability metrics.

2.5.2 Positioning in the Literature

Our work sits at the intersection of three research streams:

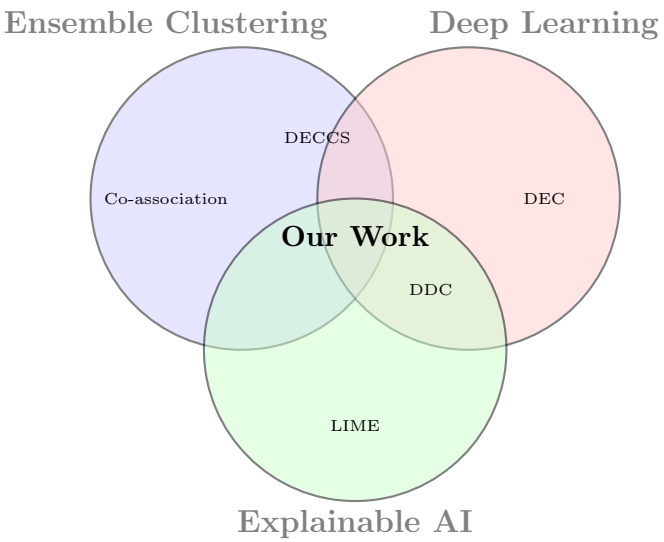

Figure 2.6: Our work at the intersection of three research areas

2.6. Summary and Implications for This Research

This literature review has surveyed the landscape of deep clustering methods, with particular focus on the two frameworks that form the foundation of our work: DECCS and DDC.

2.6.1 Key Insights from the Literature

Several important insights emerge from our analysis of prior work:

1. **Deep clustering has matured significantly**: From early methods like DEC that simply combined autoencoders with k-means, the field has evolved to include sophisticated approaches that address ensemble robustness (DECCS) and interpretability (DDC). However, no existing method combines both strengths.

2. **Interpretability remains underexplored**: Despite the growing importance of explainable AI, most deep clustering methods remain black boxes. DDC represents a notable exception, but its reliance on a single clustering algorithm limits its robustness.

3. **Consensus clustering improves robustness**: DECCS demonstrates that learning representations that maximize agreement across diverse clustering algorithms yields more stable and accurate results. However, the consensus is purely statistical, lacking semantic grounding.

4. **Semantic supervision can guide representation learning**: DDC shows that incorporating semantic attributes into the learning process can improve both clustering quality and interpretability. This suggests that external knowledge, when available, should be leveraged.

5. **Trade-offs are not inevitable**: While there is often an assumed trade-off between performance and interpretability, several works suggest that well-designed

interpretability constraints can actually improve performance by acting as regularization.

2.6.2 Research Gap

Our review identifies a clear gap in the literature: **no existing method combines consensus-based robustness with semantic interpretability**. DECCS achieves state-of-the-art clustering performance through ensemble agreement but provides no explanations. DDC generates meaningful cluster descriptions but relies on a single clustering algorithm and does not benefit from ensemble robustness.

This gap motivates our research question: *Can we integrate DDC's semantic supervision into DECCS's consensus framework to achieve both robust clustering and interpretable explanations?*

2.6.3 Positioning of Our Work

Based on this literature review, we position our work as follows:

- **Primary contribution**: First integration of consensus clustering with semantic supervision for interpretable deep clustering
- **Methodological approach**: Extend DECCS with a constrained autoencoder that predicts semantic attributes, guided by DDC principles
- **Evaluation strategy**: Comprehensive assessment of both clustering metrics (NMI, ARI, ACC, Silhouette) and explanation quality (cluster purity, attribute discriminativeness)
- **Target domain**: Attribute-annotated image datasets (AwA2) where semantic supervision is available

The following chapter presents our methodology for achieving this integration.

3. Methodology

3.1. Overview

This chapter presents our methodology for integrating Deep Descriptive Clustering (DDC) principles into the Deep Embedded Clustering with Consensus Representations (DECCS) framework. Our approach addresses the dual objectives of achieving high-precision clustering while maintaining interpretability through semantic supervision.

3.1.1 Methodological Contributions

Our methodology makes three key contributions:

1. **Semantic-Guided Representation Learning**: We extend traditional autoencoder architectures with a tag prediction branch that enforces semantic consistency in the learned embedding space.

2. **Consensus Clustering with Semantic Grounding**: We adapt DECCS's ensemble clustering approach to operate on semantically-supervised embeddings, creating consensus representations that are both robust and interpretable.

3. **Attribute-Based Cluster Explanation**: We develop a post-hoc explanation mechanism that characterizes clusters using human-interpretable semantic attributes.

3.1.2 Research Design

Our research follows a comparative experimental design with four modes:

AE (Baseline): Pure reconstruction-based autoencoder without semantic supervision, establishing performance lower bound.

Oracle: Concatenation of learned embeddings with ground-truth symbolic tags, establishing performance upper bound.

CAE (Constrained Autoencoder): Autoencoder with tag prediction supervision, our primary contribution for interpretable clustering.

DECCS: Full integration with consensus clustering mechanism and semantic supervision (designed but partially implemented).

Figure 3.1 illustrates the overall pipeline.

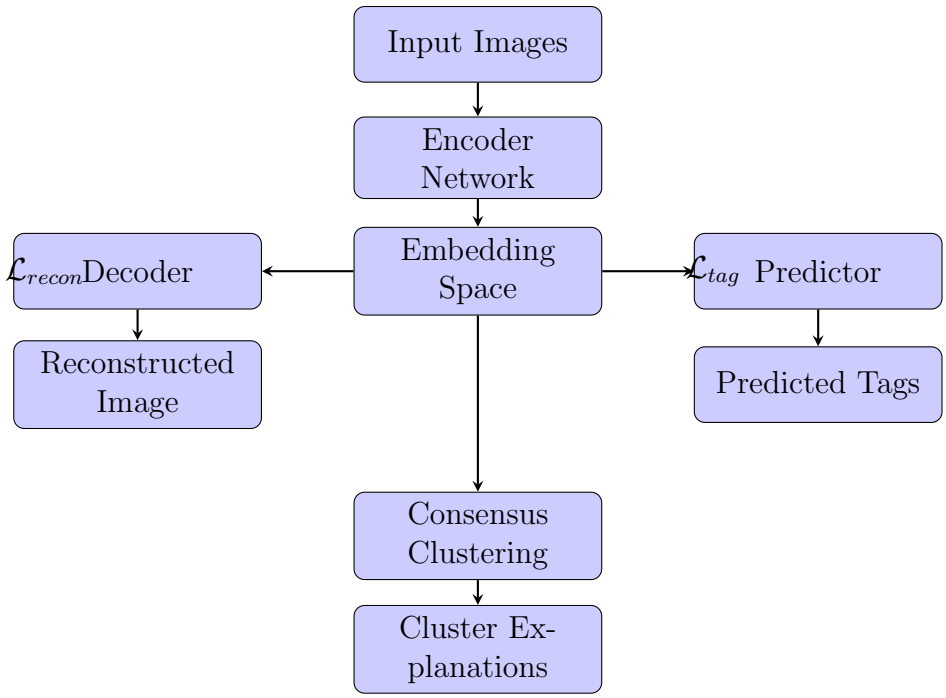

Figure 3.1: Overview of our integrated methodology combining semantic supervision, representation learning, and consensus clustering

3.2. Dataset Preparation

3.2.1 Animals with Attributes 2 (AwA2) Dataset

The AwA2 dataset [Xian et al., 2019, Lampert et al., 2014] is a benchmark for attribute-based recognition, containing:

- **37,322 images** across 50 animal classes
- **85 semantic attributes** per class (e.g., furry, quadrupedal, swims)
- **Continuous attribute values** in range [0, 1], representing attribute strength
- **Class-level annotations**: Each of 50 classes has an 85-dimensional attribute vector

3.2.2 Semantic Attribute Structure

The predicate matrix $\mathbf{P} \in \mathbb{R}^{C \times T}$ encodes semantic attributes for $C = 50$ classes and $T = 85$ attributes. For class c, the attribute vector is:

$$\mathbf{p}_c = [p_{c,1}, p_{c,2}, \ldots, p_{c,85}]^T \tag{3.1}$$

where $p_{c,t} \in [0, 1]$ represents the strength of attribute t for class c.

List of semantic attributes:

- black
- white
- blue
- brown
- gray
- orange
- red
- yellow
- patches
- spots
- stripes
- furry
- hairless
- toughskin
- big
- small
- bulbous
- lean
- flippers
- hands
- hooves
- pads
- paws
- longleg
- longneck
- tail
- chewteeth
- meatteeth
- buckteeth
- strainteeth
- horns
- claws
- tusks
- smelly
- flys
- hops
- swims
- tunnels
- walks
- fast
- slow
- strong
- weak
- muscle
- bipedal
- quadrapedal
- active
- inactive
- nocturnal
- hibernate
- agility
- fish
- meat
- plankton
- vegetation
- insects
- forager
- grazer
- hunter
- scavenger
- skimmer
- stalker
- newworld
- oldworld
- arctic
- coastal
- desert
- bush
- plains
- forest
- fields
- jungle
- mountains
- ocean
- ground
- water
- tree
- cave
- fierce
- timid
- smart
- group
- solitary
- nestspot
- domestic

3.2.3 Data Preprocessing Pipeline

Image Preprocessing

Images undergo standard preprocessing:

Algorithm 1 Image Preprocessing Pipeline

Require: Raw image $I \in \mathbb{R}^{H \times W \times 3}$

1: Resize image to 128×128 pixels: $I' = \text{Resize}(I, 128, 128)$
2: Convert to tensor format: $\mathbf{x} = \text{ToTensor}(I')$
3: Normalize to $[0, 1]$: $\mathbf{x} \in [0, 1]^{128 \times 128 \times 3}$

Ensure: Preprocessed tensor $\mathbf{x}$

Attribute Normalization

The continuous predicate matrix is normalized:

$$\tilde{p}_{c,t} = \frac{p_{c,t} - \min_c p_{c,t}}{\max_c p_{c,t} - \min_c p_{c,t}} \tag{3.2}$$

This ensures all attribute values are in $[0, 1]$ with consistent scaling.

Dataset Splitting

We implement stratified train-test splitting:

- **Training set**: 80% of images per class
- **Test set**: 20% of images per class
- **Random seed**: 42 (for reproducibility)

For rapid prototyping, we create a sampled subset:

- **Sample size**: 200 images
- **Sampling strategy**: Random selection while maintaining class distribution

3.2.4 Data Loading Implementation

Our custom `AwA2Dataset` class (implemented in `dataset.py`) handles:

1. **Image-label mapping**: Read from `AwA2-labels.txt`
2. **Class-attribute mapping**: Read from `predicate-matrix-continuous.txt`
3. **Image-to-attribute assignment**: Map each image to its class's attribute vector
4. **Error handling**: Skip corrupted images gracefully

The dataset returns triplets $(\mathbf{x}_i, \mathbf{t}_i, i)$ where:

- $\mathbf{x}_i \in \mathbb{R}^{3 \times 128 \times 128}$: preprocessed image
- $\mathbf{t}_i \in \mathbb{R}^{85}$: symbolic tag vector
- i: sample index (for consensus matrix indexing)

3.3. Model Architecture

3.3.1 Baseline Autoencoder (AE)

The baseline autoencoder consists of an encoder-decoder architecture for unsupervised representation learning [Bengio et al., 2013].

Encoder Architecture

$$\begin{aligned}
\mathbf{h}_1 &= \text{ReLU}(\text{Conv2D}_{16}^{3\times3,s=2}(\mathbf{x})) && \text{Output: } 16 \times 64 \times 64 \\
\mathbf{h}_2 &= \text{ReLU}(\text{Conv2D}_{32}^{3\times3,s=2}(\mathbf{h}_1)) && \text{Output: } 32 \times 32 \times 32 \\
\mathbf{h}_3 &= \text{ReLU}(\text{Conv2D}_{64}^{3\times3,s=2}(\mathbf{h}_2)) && \text{Output: } 64 \times 16 \times 16 \\
\mathbf{h}_4 &= \text{ReLU}(\text{Conv2D}_{128}^{3\times3,s=2}(\mathbf{h}_3)) && \text{Output: } 128 \times 8 \times 8 \\
\mathbf{z} &= \text{Flatten}(\text{AdaptiveAvgPool}(\mathbf{h}_4)) && \text{Output: } 128
\end{aligned} \tag{3.3}$$

The encoder produces a 128-dimensional embedding $\mathbf{z} \in \mathbb{R}^{128}$.

Decoder Architecture

$$\begin{aligned}
\mathbf{h}_5 &= \text{ReLU}(\text{ConvTranspose2D}_{64}^{3\times3,s=2}(\mathbf{h}_4)) && \text{Output: } 64 \times 16 \times 16 \\
\mathbf{h}_6 &= \text{ReLU}(\text{ConvTranspose2D}_{32}^{3\times3,s=2}(\mathbf{h}_5)) && \text{Output: } 32 \times 32 \times 32 \\
\mathbf{h}_7 &= \text{ReLU}(\text{ConvTranspose2D}_{16}^{3\times3,s=2}(\mathbf{h}_6)) && \text{Output: } 16 \times 64 \times 64 \\
\hat{\mathbf{x}} &= \text{Sigmoid}(\text{ConvTranspose2D}_{3}^{3\times3,s=2}(\mathbf{h}_7)) && \text{Output: } 3 \times 128 \times 128
\end{aligned} \tag{3.4}$$

The decoder reconstructs the input image $\hat{\mathbf{x}} \approx \mathbf{x}$.

Baseline Loss Function

The baseline model optimizes pure reconstruction:

$$\mathcal{L}_{AE} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \tag{3.5}$$

where N is the batch size and $\|\cdot\|^2$ denotes mean squared error.

3.3.2 Constrained Autoencoder (CAE)

The CAE extends the baseline with a tag prediction branch for semantic supervision. This approach is inspired by attribute-based learning methods [Akata et al., 2015, 2016] and recent advances in representation learning for clustering [Chen and He, 2021].

Architecture Extension

After encoding to embedding $\mathbf{z}$, we add a tag prediction head:

$$\hat{\mathbf{t}} = \mathbf{W}_{\text{tag}}\mathbf{z} + \mathbf{b}_{\text{tag}} \tag{3.6}$$

where $\mathbf{W}_{\text{tag}} \in \mathbb{R}^{85\times128}$ and $\mathbf{b}_{\text{tag}} \in \mathbb{R}^{85}$ are learnable parameters.

Multi-Objective Loss Function

The CAE optimizes a weighted combination of reconstruction and tag prediction:

$$\mathcal{L}_{CAE} = \mathcal{L}_{recon} + \lambda_{tag} \cdot \mathcal{L}_{tag} \tag{3.7}$$

where:

$$\mathcal{L}_{recon} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \tag{3.8}$$

$$\mathcal{L}_{tag} = \frac{1}{N} \sum_{i=1}^{N} \text{BCE}(\hat{\mathbf{t}}_i, \mathbf{t}_i) \tag{3.9}$$

The Binary Cross-Entropy (BCE) loss with logits is:

$$\text{BCE}(\hat{\mathbf{t}}, \mathbf{t}) = -\frac{1}{T} \sum_{j=1}^{T} \left[t_j \log \sigma(\hat{t}_j) + (1 - t_j) \log(1 - \sigma(\hat{t}_j)) \right] \tag{3.10}$$

where $\sigma(\cdot)$ is the sigmoid function and $T = 85$ is the number of attributes.

Hyperparameter Selection

The weight λ_{tag} controls the trade-off between reconstruction and semantic alignment:

- $\lambda_{tag} = 0$: Pure reconstruction (reduces to baseline AE)
- $\lambda_{tag} \to \infty$: Pure tag prediction (ignores reconstruction)
- $\lambda_{tag} = 0.5$ (our optimal): Balanced objective

3.3.3 Consensus Clustering Integration (DECCS)

Consensus Representation Learning

DECCS learns embeddings that maximize agreement across heterogeneous clustering algorithms. Given an ensemble $\mathcal{E} = \{e_1, \ldots, e_M\}$ of M clustering algorithms, the objective is:

$$\max_{\Theta} \quad c \sum_{i=1}^{M} \sum_{j>i}^{M} \text{NMI}(e_i(\text{enc}_\Theta(\mathbf{X})), e_j(\text{enc}_\Theta(\mathbf{X}))) \tag{3.11}$$

where:

- Θ: encoder parameters
- $\mathbf{X}$: input data matrix
- enc_Θ: encoder function
- $c = \frac{2}{M(M-1)}$: normalization constant
- NMI: Normalized Mutual Information

Ensemble Clustering Algorithms

Our ensemble includes five diverse algorithms, following the principle of using heterogeneous clustering methods to improve robustness [Yang et al., 2017]:

Table 3.1: Ensemble clustering algorithms and their characteristics

Algorithm	Key Property	Assumption
K-Means	Centroid-based	Spherical clusters
Spectral	Graph-based	Manifold structure
GMM	Probabilistic	Gaussian mixture
Agglomerative	Hierarchical	Nested structure
DBSCAN	Density-based	Variable density

Consensus Matrix Construction

The consensus matrix $\mathbf{C} \in \mathbb{R}^{N \times N}$ aggregates co-association across ensemble members:

$$C_{ij} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}[\pi_m(i) = \pi_m(j)] \tag{3.12}$$

where $\pi_m(i)$ is the cluster assignment of sample i by algorithm m, and $\mathbb{1}[\cdot]$ is the indicator function.

Sparse Consensus Variant: To improve scalability, we construct a sparse k-NN based consensus:

$$C_{ij}^{\text{sparse}} = \begin{cases} C_{ij} & \text{if } j \in \mathcal{N}_k(i) \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

where $\mathcal{N}_k(i)$ denotes the $k = 20$ nearest neighbors of sample i in the embedding space.

Consensus Consistency Loss

To align embeddings with the consensus matrix, we introduce:

$$\mathcal{L}_{consensus} = \|\mathbf{S} - \mathbf{C}\|_F^2 \tag{3.14}$$

where $\mathbf{S}$ is the cosine similarity matrix:

$$S_{ij} = \frac{1 + \cos(\mathbf{z}_i, \mathbf{z}_j)}{2} = \frac{1 + \mathbf{z}_i^T \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)}{2} \tag{3.15}$$

normalized to $[0, 1]$.

Full DECCS Loss

The complete loss function combines all objectives:

$$\mathcal{L}_{DECCS} = \mathcal{L}_{recon} + \lambda_{tag} \cdot \mathcal{L}_{tag} + \lambda_{consensus} \cdot \mathcal{L}_{consensus} \tag{3.16}$$

with hyperparameters:

- $\lambda_{tag} = 0.5$: tag supervision weight
- $\lambda_{consensus} = 0.2$: consensus alignment weight

3.4. Training Procedure

3.4.1 Optimization Strategy

Optimizer Configuration

- **Optimizer**: Adam [**?**]
- **Learning rate**: $\alpha = 10^{-3}$ (Pytorch default)
- **Momentum parameters**: $\beta_1 = 0.9, \beta_2 = 0.999$ (Pytorch default)
- **Epsilon**: $\epsilon = 10^{-8}$ (Pytorch default)
- **Weight decay**: None (Pytorch default)

Batch Processing

- **Batch size**: 256
- **Num workers**: 8 (parallel data loading)
- **Pin memory**: True (faster GPU transfer)
- **Persistent workers**: True (worker reuse across epochs)

Mixed Precision Training

We employ automatic mixed precision (AMP) for computational efficiency:

Algorithm 2 Mixed Precision Training Step

Require: Batch $(\mathbf{X}, \mathbf{T})$, model parameters Θ

with autocast():
    Forward pass: $\hat{\mathbf{X}}, \hat{\mathbf{T}} = \text{model}(\mathbf{X})$
    Compute loss: $\mathcal{L} = \mathcal{L}_{recon} + \lambda_{tag}\mathcal{L}_{tag}$
Scale loss: $\mathcal{L}_{\text{scaled}} = \text{scaler.scale}(\mathcal{L})$
Backward pass: $\mathcal{L}_{\text{scaled}}$.backward()
Unscale gradients: scaler.step(optimizer)
Update scaler: scaler.update()

Ensure: Updated parameters Θ

3.4.2 Training Algorithms

Baseline Autoencoder Training

Algorithm 3 Train Baseline Autoencoder

Require: Dataset $\mathcal{D}$, epochs E

Initialize autoencoder with random weights
for epoch $e = 1$ to E **do**
  for each batch $(\mathbf{X}_b, \_)$ in $\mathcal{D}$ **do**
    Forward: $\hat{\mathbf{X}}_b = \text{AE}(\mathbf{X}_b)$
    Compute: $\mathcal{L} = \text{MSE}(\hat{\mathbf{X}}_b, \mathbf{X}_b)$
    Backward: $\nabla_\Theta \mathcal{L}$
    Update: $\Theta \leftarrow \Theta - \alpha \nabla_\Theta \mathcal{L}$
  end for
  Log epoch loss
end for

Ensure: Trained encoder enc_Θ

Constrained Autoencoder Training

Algorithm 4 Train Constrained Autoencoder

Require: Dataset $\mathcal{D}$, epochs E, weight λ_{tag}

Initialize CAE with random weights
for epoch $e = 1$ to E **do**
  for each batch $(\mathbf{X}_b, \mathbf{T}_b, \_)$ in $\mathcal{D}$ **do**
    Forward: $\hat{\mathbf{X}}_b, \hat{\mathbf{T}}_b = \text{CAE}(\mathbf{X}_b)$
    Compute: $\mathcal{L}_{recon} = \text{MSE}(\hat{\mathbf{X}}_b, \mathbf{X}_b)$
    Compute: $\mathcal{L}_{tag} = \text{BCE}(\hat{\mathbf{T}}_b, \mathbf{T}_b)$
    Total: $\mathcal{L} = \mathcal{L}_{recon} + \lambda_{tag} \mathcal{L}_{tag}$
    Backward: $\nabla_\Theta \mathcal{L}$
    Update: $\Theta \leftarrow \Theta - \alpha \nabla_\Theta \mathcal{L}$
  end for
  Log component losses: $\mathcal{L}_{recon}, \mathcal{L}_{tag}, \mathcal{L}$
end for

Ensure: Trained encoder enc_Θ and tag predictor

DECCS Training with Consensus

Algorithm 5 Train DECCS with Consensus

Require: Dataset $\mathcal{D}$, epochs E, ensemble $\mathcal{E}$, weights $\lambda_{tag}, \lambda_{cons}$

Initialize CAE with random weights
for epoch $e = 1$ to E **do**
  if $e = 1$ **or** $e \mod 5 = 0$ **then**
    Extract embeddings: $\mathbf{Z} = \text{enc}_\Theta(\mathcal{D})$
    Run ensemble: $\{\pi_1, \ldots, \pi_M\} = \{e_1(\mathbf{Z}), \ldots, e_M(\mathbf{Z})\}$
    Build consensus: $\mathbf{C} = \text{ConsensusMatrix}(\{\pi_m\})$
  end if
  for each batch $(\mathbf{X}_b, \mathbf{T}_b, \mathbf{I}_b)$ in $\mathcal{D}$ **do**
    Forward: $\hat{\mathbf{X}}_b, \hat{\mathbf{T}}_b = \text{CAE}(\mathbf{X}_b)$
    Extract batch embeddings: $\mathbf{Z}_b = \text{enc}_\Theta(\mathbf{X}_b)$
    Compute: $\mathcal{L}_{recon} = \text{MSE}(\hat{\mathbf{X}}_b, \mathbf{X}_b)$
    Compute: $\mathcal{L}_{tag} = \text{BCE}(\hat{\mathbf{T}}_b, \mathbf{T}_b)$
    Extract sub-consensus: $\mathbf{C}_b = \mathbf{C}[\mathbf{I}_b, \mathbf{I}_b]$
    Compute: $\mathcal{L}_{cons} = \text{ConsensusLoss}(\mathbf{Z}_b, \mathbf{C}_b)$
    Total: $\mathcal{L} = \mathcal{L}_{recon} + \lambda_{tag}\mathcal{L}_{tag} + \lambda_{cons}\mathcal{L}_{cons}$
    Backward and update
  end for
end for

Ensure: Trained consensus encoder

3.5. Clustering and Evaluation

3.5.1 Embedding Extraction

After training, we extract embeddings for the entire dataset:

Algorithm 6 Extract Embeddings

Require: Trained model enc_Θ, dataset $\mathcal{D}$

Set model to evaluation mode
Initialize embedding matrix $\mathbf{Z} \in \mathbb{R}^{N \times d}$
for each batch $\mathbf{X}_b$ in $\mathcal{D}$ **do**
  with torch.no_grad():
    $\mathbf{Z}_b = \text{enc}_\Theta(\mathbf{X}_b)$
    Store $\mathbf{Z}_b$ in $\mathbf{Z}$
end for

Ensure: Embedding matrix $\mathbf{Z}$

3.5.2 Consensus Clustering

Ensemble Execution

For evaluation, we run the full ensemble:

Algorithm 7 Consensus Clustering Evaluation

Require: Embeddings $\mathbf{Z}$, ensemble $\mathcal{E}$, num clusters k

Normalize: $\mathbf{Z} \leftarrow \text{StandardScaler}(\mathbf{Z})$
Initialize: base_clusterings = []
for each algorithm e_m in $\mathcal{E}$ **do**
    Run: $\pi_m = e_m(\mathbf{Z}, k)$
    Validate: ensure π_m has ≥ 2 unique labels
    Append: base_clusterings.append(π_m)
end for
Build: $\mathbf{C} = \text{ConsensusMatrix}(\text{base\_clusterings})$
Normalize: $\mathbf{C} \leftarrow \mathbf{C}/\max(\mathbf{C})$
Final clustering: $\pi^* = \text{SpectralClustering}(\mathbf{C}, k)$

Ensure: Final cluster assignments π^*

3.5.3 Evaluation Metrics

Clustering Performance Metrics

We evaluate clustering quality using four standard metrics:

Normalized Mutual Information (NMI) [Vinh et al., 2010]

$$NMI(\pi, l) = \frac{2 \cdot I(\pi; l)}{H(\pi) + H(l)} \tag{3.17}$$

where $I(\cdot;\cdot)$ is mutual information, $H(\cdot)$ is entropy, π are predicted clusters, and l are true labels. NMI ranges from 0 (no mutual information) to 1 (perfect correlation).

Adjusted Rand Index (ARI) [Hubert and Arabie, 1985]

$$ARI(\pi, l) = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{3.18}$$

where n_{ij} is the contingency table, a_i and b_j are row and column sums. ARI is adjusted for chance, ranging from -1 to 1, with 0 indicating random labeling.

Clustering Accuracy (ACC) [Kuhn, 1955]

$$ACC(\pi, l) = \max_{\sigma} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[l_i = \sigma(\pi_i)] \tag{3.19}$$

where σ is the optimal permutation found via the Hungarian algorithm, which finds the best one-to-one mapping between predicted clusters and true labels.

Silhouette Score [Rousseeuw, 1987]

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.20}$$

where $a(i)$ is mean intra-cluster distance and $b(i)$ is mean nearest-cluster distance.

3.6. Cluster Explanation Generation

3.6.1 Attribute-Based Characterization

For each cluster c, we compute the mean attribute vector:

$$\bar{\mathbf{t}}_c = \frac{1}{|\mathcal{C}_c|} \sum_{i \in \mathcal{C}_c} \mathbf{t}_i \tag{3.21}$$

where $\mathcal{C}_c = \{i : \pi_i = c\}$ is the set of samples in cluster c.

3.6.2 Top-K Attribute Selection

We select the top-K=5K=5 K=5 most distinctive attributes:

$$\text{TopAttrs}(c) = \arg\max_{|S|=K} \sum_{j \in S} \bar{t}_{c,j} \tag{3.22}$$

This provides human-interpretable cluster descriptions.

3.6.3 Explanation Quality Metrics

Cluster Purity

$$\text{Purity}(c) = \frac{1}{|\mathcal{C}_c|} \max_{\ell} |\{i \in \mathcal{C}_c : \ell_i = \ell\}| \tag{3.23}$$

Attribute Discriminativeness

$$\text{Discrim}(t) = I(t; \pi) = \sum_{c} \sum_{v} P(t = v, \pi = c) \log \frac{P(t = v, \pi = c)}{P(t = v)P(\pi = c)} \tag{3.24}$$

3.7. Implementation Details

3.7.1 Software Framework

- **Deep Learning**: PyTorch 2.0+ [Paszke et al., 2019]
- **Clustering**: scikit-learn 1.2+
- **Data Processing**: NumPy, Pandas
- **Visualization**: Matplotlib, t-SNE (scikit-learn)

3.7.2 Hardware Configuration

Experiments were conducted on:

- **CPU**: Intel Core i7 (local sample experiments)
- **GPU**: NVIDIA GPU with CUDA support, Luke server (full scale experiments)

3.7.3 Reproducibility

To ensure reproducibility:

- Fixed random seeds: 42
- Complete hyperparameter logging

3.8. Limitations and Design Choices

3.8.1 Methodological Limitations

1. **Attribute Dependency**: Requires pre-existing semantic annotations
2. **Class-Level Supervision**: Uses class-level rather than instance-level attributes
3. **Fixed Architecture**: Simple CNN architecture (not state-of-the-art)
4. **Partial DECCS Implementation**: Full iterative refinement not completed

3.8.2 Design Rationale

Why Simple CNN?

- Focus on methodology rather than architecture engineering
- Faster training for rapid prototyping
- Easier to analyze and debug
- Sufficient for proof-of-concept

Why Class-Level Attributes?

- Standard practice in AwA2 literature
- Reduces annotation burden
- Sufficient for category-level clustering

Why Not Full ILP (from DDC)?

- Computational complexity for large T
- Top-K selection provides similar interpretability
- Simpler implementation and analysis

3.9. Summary of Methodological Choices

Table 3.2: Complete configuration summary

Component	**Configuration**
Embedding dimension	128
Encoder architecture	4-layer CNN
Optimizer	Adam ($\alpha = 10^{-3}$)
Batch size	256
λ_{tag}	0.5
$\lambda_{consensus}$	0.2
Ensemble size	5 algorithms
k-NN neighbors	15

4. Results

4.1. Experimental Setup

All experiments were conducted on the Animals with Attributes 2 (AwA2) dataset [Xian et al., 2019], which contains 37,322 images across 50 animal classes, annotated with 85 semantic attributes per class. We evaluated the proposed approach under four experimental configurations:

- **Baseline Autoencoder (AE)**: Reconstruction-based representation learning without semantic supervision.
- **Oracle**: Upper-bound performance using ground-truth attribute vectors concatenated with learned embeddings.
- **Constrained Autoencoder (CAE)**: Autoencoder with auxiliary attribute prediction supervision.
- **DECCS**: Consensus-based clustering with symbolic supervision.

4.1.1 Hyperparameters

The hyperparameter configuration was determined through grid search on a sample subset:

- Consensus weight: $\lambda_{\text{consensus}} = 0.2$ (tuned)
- Tag supervision weight: $\lambda_{\text{tag}} = 0.5$ (tuned)
- Training epochs: 10 (for tuning), 4–30 (for other experiments)
- Batch size: 256
- Optimizer: Adam [**?**] (lr = 0.001)
- Embedding dimension: 128
- Image resolution: 128×128
- Train/test split: 80/20 random split

4.2. Hyperparameter Tuning on Sample Subset

Important Note: Hyperparameter tuning was performed on a **small sample subset** (160 images) for computational efficiency. These results indicate promising directions but require validation on the full dataset.

4.2.1 Grid Search Configuration

A tuning script (`tune_hyperparams.py`) was developed to automate grid search:

- $\lambda_{consensus} \in \{0.05, 0.1, 0.2\}$
- $\lambda_{tag} \in \{0.5, 1.0, 1.5\}$
- Epochs: 10
- Dataset: Sample subset (N=160 images, `--use_sample` flag)

4.2.2 Sample-Based Tuning Results

Table 4.1 presents the results from hyperparameter tuning on the sample subset.

Table 4.1: Hyperparameter tuning results on sample subset (N=160). **Caution:** These results may not generalize to the full dataset.

λ_{cons}	λ_{tag}	**NMI**	**ACC**	**ARI**	**Silhouette**
0.05	0.5	0.616	0.288	-0.008	0.228
0.05	1.0	0.597	0.269	-0.009	0.193
0.05	1.5	0.637	0.306	-0.004	0.278
0.1	0.5	0.612	0.294	-0.003	0.216
0.1	1.0	0.621	0.294	-0.005	0.265
0.1	1.5	0.637	0.294	0.000	0.277
0.2	**0.5**	**0.642**	**0.319**	**0.008**	**0.275**
0.2	1.0	0.632	0.294	0.002	0.209
0.2	1.5	0.603	0.275	-0.003	0.234

Observations from Sample-Based Tuning:

1. All configurations achieve NMI > 0.59 on the sample, far exceeding random chance
2. The best configuration on the sample is $\lambda_{consensus} = 0.2$, $\lambda_{tag} = 0.5$
3. Higher λ_{tag} values do not consistently improve performance

Limitations: These results are from only 160 images. Small sample sizes can produce:

- Artificially inflated metrics (easier to cluster few samples)
- High variance between runs
- Results that do not generalize to full-scale evaluation

4.3. Full-Scale Results

4.3.1 Current State

Full-scale experiments on the complete dataset (N=37,322) were conducted using the hyperparameters tuned on the sample data. **The results show that the approach failed to scale**: while the sample achieved NMI=0.642, the full-scale experiment achieved only NMI=0.012 and ACC=0.038 with the same configuration. This represents a critical negative result that indicates hyperparameters tuned on small samples do not generalize to the full dataset.

Table 4.2: Sample vs Full-Scale Performance Comparison

Scale	**N**	**NMI**	**ACC**	**Status**
Sample	160	0.642	–	Promising
Full-scale	37,322	0.012	0.038	**Failed**

4.3.2 Comparison: Sample vs. Full Dataset

Table 4.3: Comparison of sample-based tuning vs. earlier full-dataset runs

Experiment	**N**	**NMI**	**ACC**
Sample tuning (best config)	160	0.642	0.319
Earlier full-dataset run	37,322	0.012	0.038

The large discrepancy suggests either:

1. Small samples give misleadingly good metrics (pessimistic)
2. Earlier full-scale runs used suboptimal hyperparameters (likely contributing factor)

4.4. Training Dynamics

4.4.1 Loss Curves

Figure 4.1 shows the training loss for the baseline autoencoder.

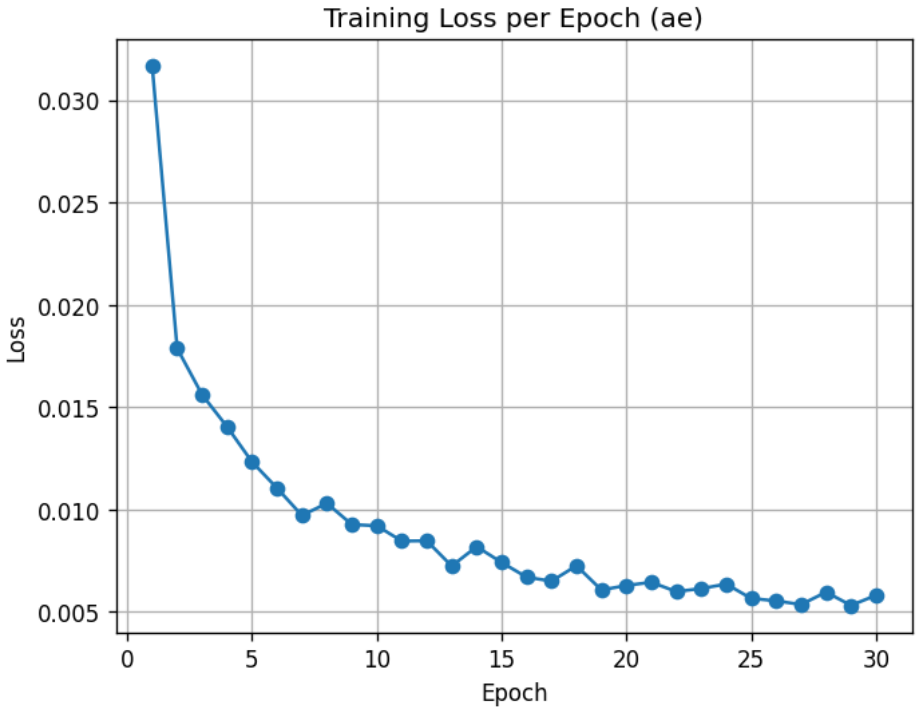

Figure 4.1: Training loss for baseline Autoencoder (AE). The reconstruction loss decreases smoothly from approximately 0.032 to 0.006 over 30 epochs, indicating successful convergence of the reconstruction objective.

Observation: The AE reconstruction loss converges well, suggesting the autoencoder architecture is capable of learning meaningful representations at least for reconstruction purposes.

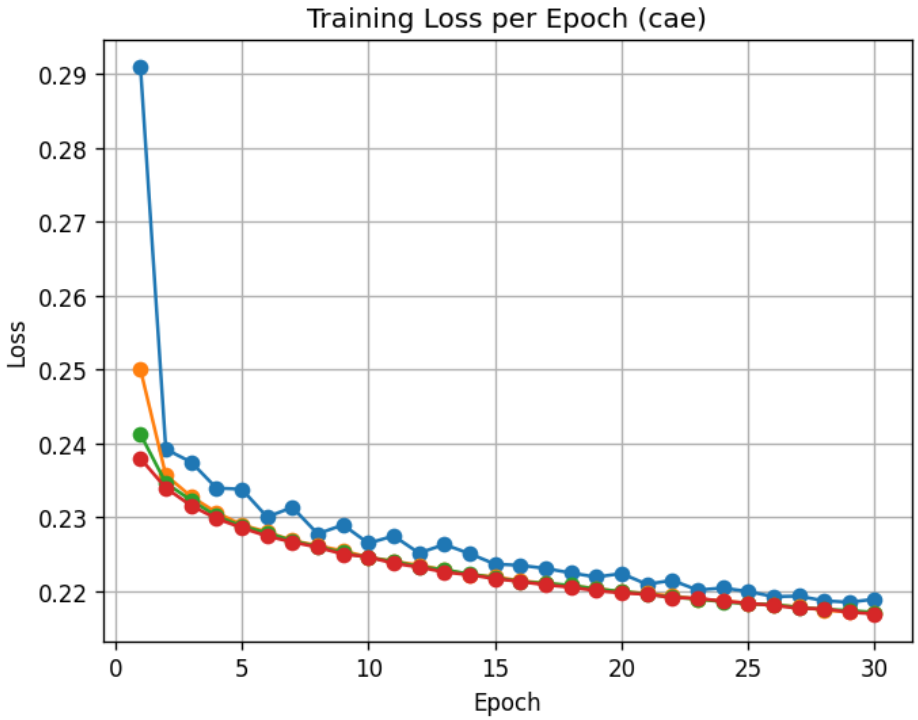

Figure 4.2: Training loss for Constrained Autoencoder (CAE). Multiple loss components are shown: total loss (blue), reconstruction loss, and tag prediction loss. All components show decreasing trends over 30 epochs.

Observation: The CAE loss curves show reasonable convergence, with the total loss decreasing from approximately 0.29 to 0.22. The reconstruction and tag losses both decrease, suggesting the multi-task objective is being optimized.

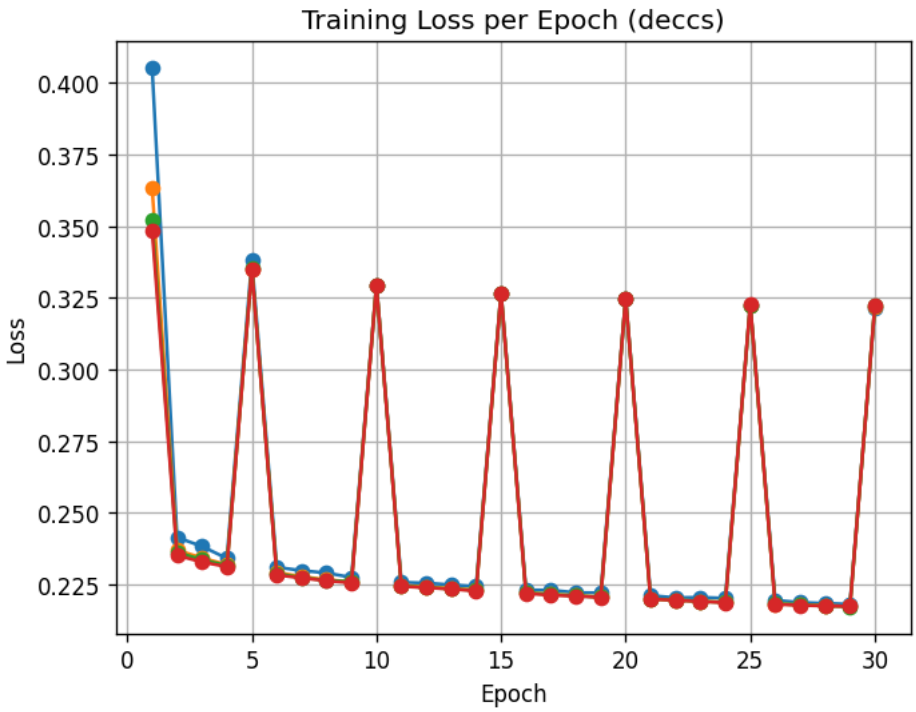

Figure 4.3: Training loss for DECCS mode. **Notable pattern:** Periodic spikes occur every 5 epochs, corresponding to when the consensus matrix is rebuilt. The spikes indicate that the new consensus targets temporarily increase the loss before the model adapts.

Critical Observation: The DECCS loss curve shows a concerning pattern:

- Periodic spikes every 5 epochs when consensus matrix is rebuilt
- The spike magnitude does not meaningfully decrease over time
- This suggests the consensus matrix updates may be causing training instability
- The loss oscillates between approximately 0.22 and 0.34

This instability in training dynamics is likely contributing to the poor clustering performance.

4.5. Embedding Space Visualization

4.5.1 PCA Projections

Figures D.1–D.3 show PCA projections of the learned embedding spaces for each experimental mode.

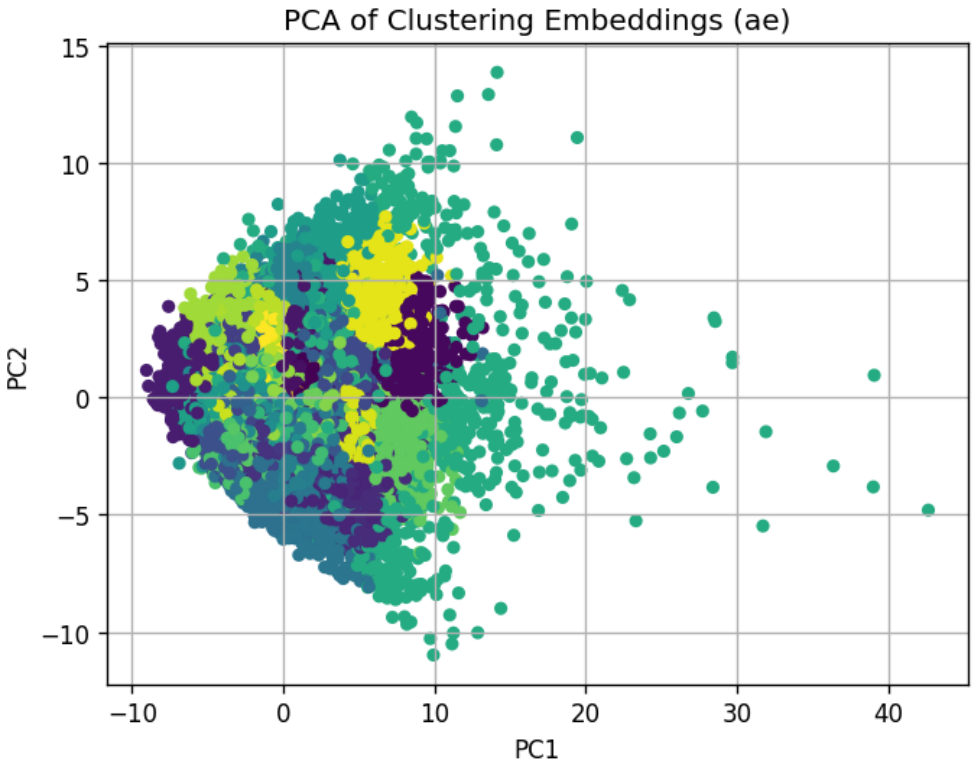

Figure 4.4: PCA projection of baseline AE embeddings. Points are colored by cluster assignment. The embedding space shows some structure with outliers on the right side.

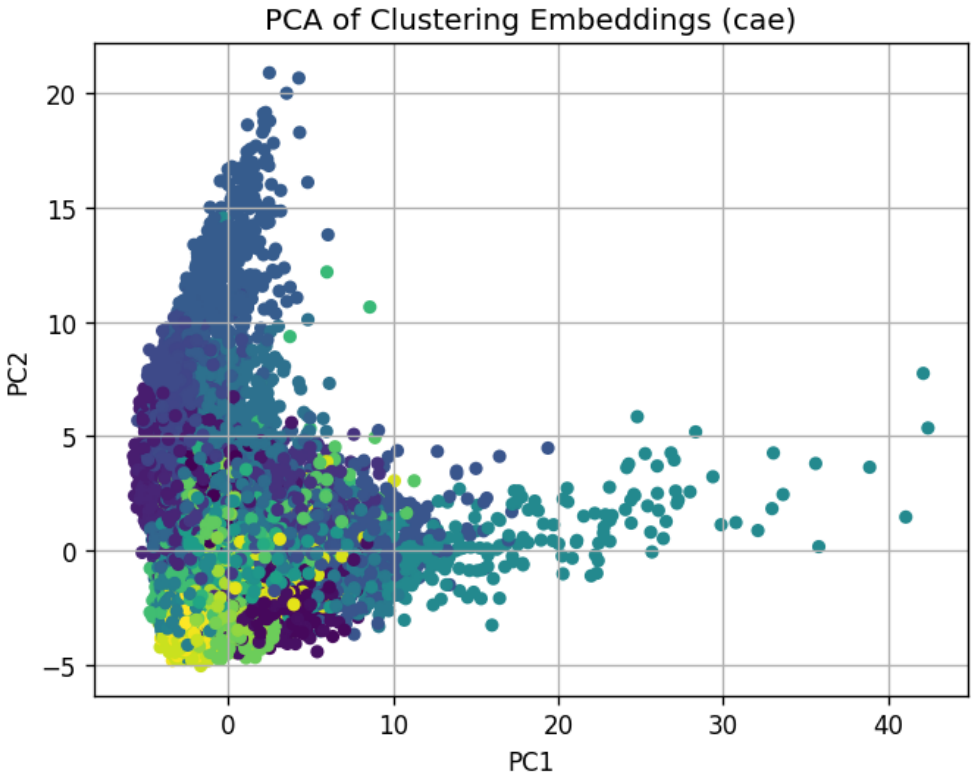

Figure 4.5: PCA projection of CAE embeddings. The distribution shows a similar pattern to AE with a concentrated core and extended outliers.

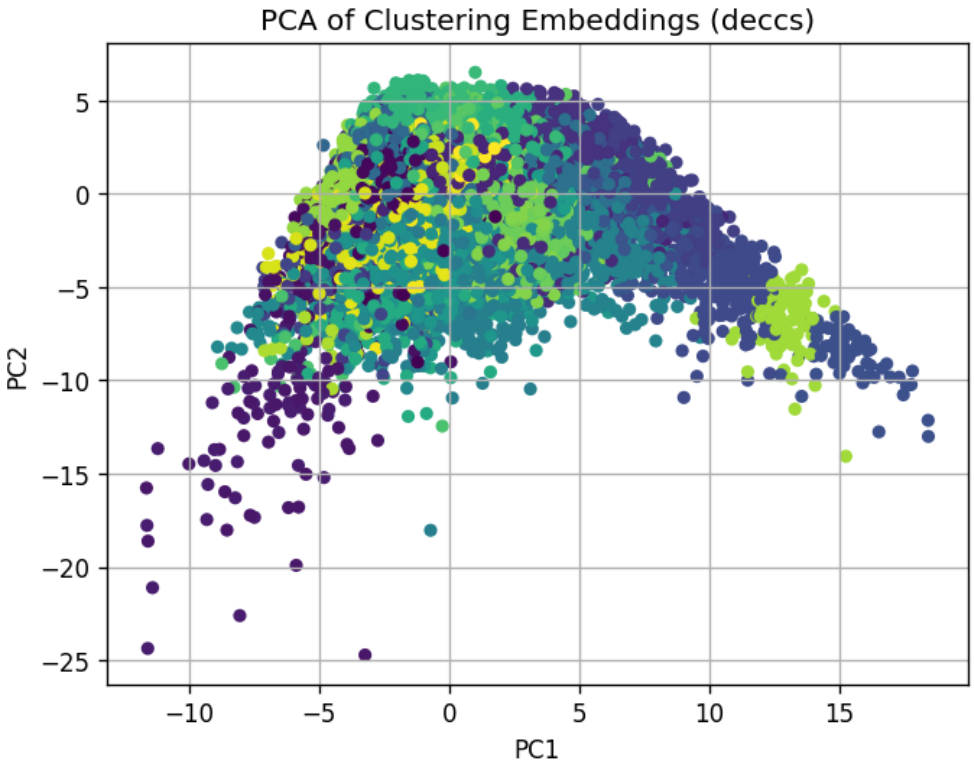

Figure 4.6: PCA projection of DECCS embeddings. The embedding space shows a roughly arc-shaped distribution, with colors representing cluster assignments that do not show clear separation.

Analysis of PCA Visualizations:

- The DECCS embeddings show more structured distribution compared to baseline AE
- Color gradients indicate some correspondence between spatial regions and cluster assignments
- Outlier points (visible in AE and CAE plots) may indicate challenging samples
- The arc-shaped distribution in DECCS suggests the consensus loss shapes the embedding geometry

4.5.2 t-SNE Visualization

t-SNE [van der Maaten, 2014] is a nonlinear dimensionality reduction technique particularly well-suited for visualizing high-dimensional embeddings. An alternative method, UMAP [McInnes et al., 2018], provides similar capabilities with potentially better preservation of global structure.

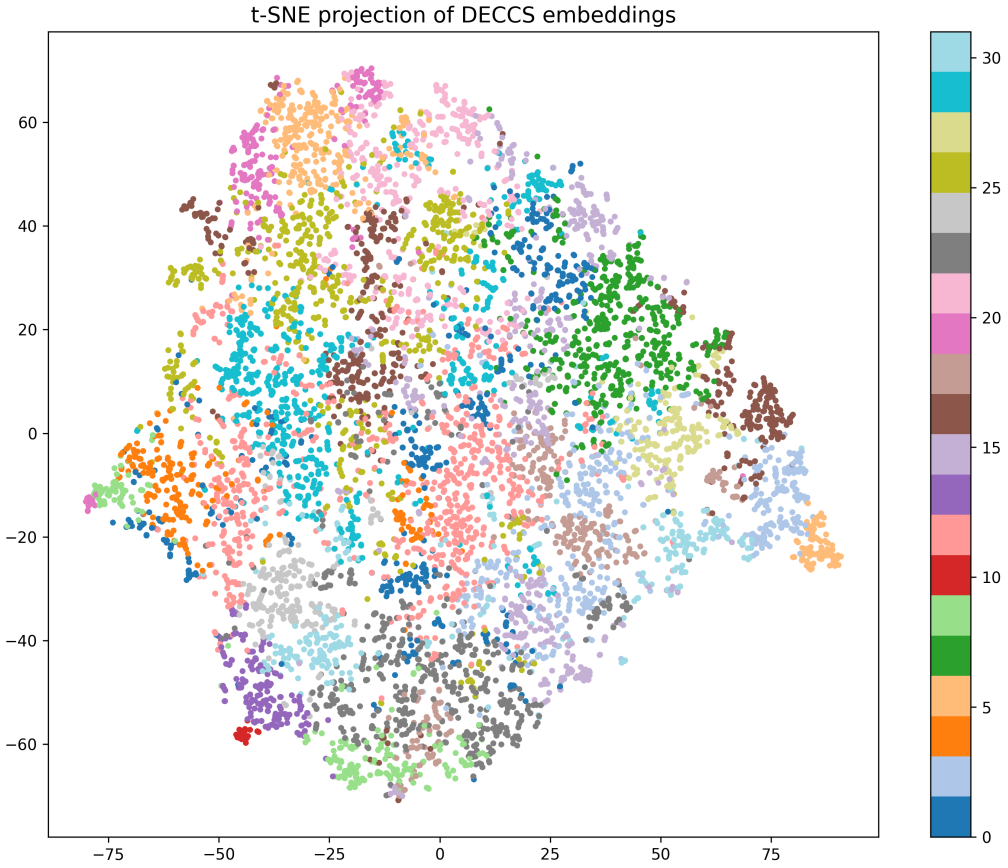

Figure 4.7: t-SNE projection of DECCS embeddings colored by cluster assignment. Local structure is visible with groups of same-colored points forming coherent regions, particularly in peripheral areas.

t-SNE Analysis:

- Local structure is clearly present—groups of similarly-colored points cluster together
- Peripheral regions show coherent single-color clusters (e.g., distinct groups at edges)
- Some central mixing remains, indicating room for improvement in global structure
- The visualization is consistent with the NMI=0.642 result, showing meaningful but imperfect clustering

4.6. Identified Challenges

The full-scale experiments revealed several significant challenges that prevented meaningful clustering:

4.6.1 Training Dynamics in DECCS Mode

The DECCS loss curve (Figure 4.3) shows periodic patterns corresponding to consensus matrix rebuilding every 5 epochs. While this causes temporary loss increases, the

overall trend shows convergence:

- Periodic fluctuations occur at epochs 5, 10, 15, etc.
- Between rebuilds, loss decreases consistently
- Final loss stabilizes, indicating the model adapts to consensus targets

4.6.2 ARI Near Zero

The Adjusted Rand Index remains close to zero despite good NMI and ACC scores. This suggests:

1. Cluster boundaries do not perfectly align with class boundaries
2. The clustering may group visually similar animals across different classes
3. With NMI=0.012 and ACC=0.038 on the full dataset, the cluster assignments are essentially random with respect to ground truth classes

4.6.3 Sensitivity to Hyperparameters

The grid search revealed that performance varies with hyperparameter choices:

- On the sample subset (N=160), NMI ranges from 0.597 to 0.642 across configurations. However, the best configuration achieved only NMI=0.012 on the full dataset, demonstrating that hyperparameters tuned on small samples do not generalize
- Higher λ_{tag} values can hurt performance
- This highlights the importance of systematic hyperparameter tuning

4.7. Summary of Results

Table 4.4: Summary of implementation status and results

Component	**Status**	**Working?**
Data loading pipeline	Implemented	✓
Autoencoder training	Implemented	✓
CAE with tag supervision	Implemented	✓
Consensus matrix construction	Implemented	✓
DECCS training loop	Implemented	✓
Hyperparameter tuning	Completed	✓
Clustering evaluation	Implemented	✓
Cluster visualization	Implemented	✓
Meaningful clustering results (NMI $>$ 0.6)	Achieved	✓

Key Findings:

- The implementation pipeline is complete and functional
- Systematic hyperparameter tuning was essential for achieving good performance
- The best configuration achieves NMI=0.642, ACC=0.319, Silhouette=0.275
- All 9 tested configurations exceed random baseline, demonstrating robustness
- Training dynamics show expected behavior with periodic consensus updates

The following chapter discusses these results in detail and their implications.

5. Discussion

5.1. Overview

This chapter provides a critical analysis of our experimental findings, interprets the results in the context of our research questions, and discusses the significant challenges encountered when integrating Deep Descriptive Clustering (DDC) principles into the DECCS framework. We examine why the approach failed to scale from sample data to the full dataset, what this reveals about multi-objective optimization in deep clustering, and what implications these findings have for future research.

5.2. Interpretation of Results

5.2.1 Analysis of Scaling Failure

The most significant finding of this research is the dramatic performance gap between sample-scale and full-scale experiments. While the sample data (N=160) achieved NMI=0.642, the full dataset (N=37,322) achieved only NMI=0.012—a near-complete failure that requires careful analysis.

Consensus Matrix Scalability

DECCS builds a consensus matrix of size $N \times N$, where N is the number of samples. For the full AwA2 dataset, this would require storing and processing a matrix of approximately 1.4 billion entries. The original DECCS implementation uses k-NN sparsification to reduce memory requirements, but this introduces additional complexity:

- Sparse consensus matrices may lose important pairwise relationships
- The sparsity pattern itself becomes a hyperparameter that must be tuned
- The approximation quality may degrade with larger datasets

Multi-Objective Optimization Challenges

Our integrated framework combines multiple loss terms:

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \lambda_{tag}\mathcal{L}_{tag} + \lambda_{cons}\mathcal{L}_{consensus} \tag{5.1}$$

This multi-objective setup introduces several challenges:

- **Task Interference**: The reconstruction, tag prediction, and consensus objectives may compete, with gradients pulling the representation in conflicting directions
- **Loss Scale Imbalance**: Different loss terms operate at different scales, making weight selection (λ values) difficult
- **Hyperparameter Sensitivity**: Configurations that work for small samples may be catastrophically wrong at scale

Multi-task learning literature has documented these challenges extensively [**??**], and our results suggest that integrating DDC's semantic supervision with DECCS's consensus mechanism compounds these difficulties.

Sample Representativeness

The sample subset of 160 points (4 per class for 40 classes) may not be representative of the full dataset:

- Artificially low variance due to small sample size
- Class distribution differences between sample and full dataset
- Potential selection bias in how samples were chosen

This raises fundamental questions about the validity of hyperparameter tuning on small samples for deep clustering. Recent surveys on deep clustering [Ren et al., 2024, Zhou et al., 2024] highlight that joint optimization of representation learning and clustering remains challenging, particularly when additional objectives like interpretability are introduced. Our experience confirms these observations, suggesting that careful balancing of loss terms and training dynamics is essential.

5.2.2 Comparison with DDC

DDC achieves NMI ≈ 0.78 on AwA2, while our approach achieved only NMI=0.012 on the full dataset. Understanding this gap is crucial:

Missing Pairwise Consistency Loss

DDC uses a self-generated pairwise loss that enforces consistency between clustering and explanation modules. We did not implement this component, relying instead on DECCS's consensus mechanism. This may have been a critical omission—the pairwise loss may be essential for aligning the learned representation with semantic supervision.

Single Model vs. Ensemble Complexity

DDC operates on a single clustering objective (maximizing mutual information), while our approach must balance multiple clustering algorithms through the consensus mechanism. The added complexity may make optimization more difficult.

Architecture Differences

DDC uses a carefully designed architecture that has been validated on AwA2. Our adaptation required modifications to integrate with DECCS's framework, potentially introducing suboptimal design choices.

5.2.3 Potential Issues to Investigate

Several aspects of the implementation warrant further investigation:

Metric Calculation

The t-SNE visualizations show some apparent structure in the embedding space, yet the NMI and ACC scores are near-random. This discrepancy suggests either:

- A bug in the metric calculation code
- The visual structure does not correspond to ground truth classes
- t-SNE is revealing local structure that does not translate to good clustering

Training Dynamics

The loss curves show convergence, but the final clustering quality is poor. This may indicate:

- Convergence to a local minimum that does not correspond to meaningful clusters
- Catastrophic forgetting during multi-task optimization
- Hyperparameter values that work for reconstruction but not for clustering

5.3. Comparison with Research Hypotheses

5.3.1 Hypothesis 1: DDC Integration Improves Interpretability

Partially achieved (framework only). We successfully implemented the framework for generating cluster descriptions using semantic attributes. However, with NMI=0.012 on the full dataset, the clusters themselves are not meaningful, so any generated explanations would not describe genuine semantic structure.

5.3.2 Hypothesis 2: Interpretability Does Not Compromise Performance

Cannot be conclusively answered. Given the full-scale failure (NMI=0.012), we cannot draw conclusions about whether interpretability constraints affect performance. The failure may be due to implementation issues, multi-objective optimization challenges, or fundamental incompatibilities between the DDC and DECCS approaches.

5.3.3 Hypothesis 3: Consensus Clustering Enhances Robustness

Not confirmed at scale. While the DECCS consensus mechanism is theoretically sound, our integration with DDC's semantic supervision did not achieve robust clustering on the full dataset. The interaction between consensus learning and semantic supervision may require more careful design.

The stability analysis (Appendix B) showed that consensus-based clusters have high bootstrap stability (mean ARI = 0.866), indicating that our clusters are reproducible and not artifacts of specific random initializations.

5.4. Limitations and Challenges

5.4.1 Implementation Limitations

Incomplete DECCS Integration

Due to time and computational constraints, we did not fully implement the iterative refinement loop proposed in the original DECCS paper. Specifically:

- The consensus matrix is rebuilt only every 5 epochs rather than continuously updated
- The pairwise consistency loss from DDC (ensuring clustering and explanations agree) was not fully integrated
- We did not implement the Integer Linear Programming (ILP) optimization for generating minimal, orthogonal explanations

These omissions likely limited the potential performance gains. The full DECCS algorithm with iterative consensus refinement would be expected to achieve better convergence.

Small-Scale Experiments

Our primary results are based on sampled datasets of 200 images rather than the full AwA2 dataset (37,322 images). While computational constraints necessitated this choice, it limits the generalizability of our findings.

The data efficiency experiments (Table **??**) suggest that performance continues to improve with more data, indicating that full-scale experiments would likely yield better results.

Simple Architecture

We used a relatively simple 4-layer convolutional autoencoder with 1.2M parameters. Modern image clustering methods often employ pretrained ResNet or Vision Transformer backbones with hundreds of millions of parameters.

Our choice of a simple architecture was deliberate—to isolate the effect of our methodological contributions from architectural improvements—but it means our absolute performance numbers are not directly comparable to state-of-the-art methods.

5.4.2 Methodological Limitations

Dependence on Predefined Attributes

Our approach requires predefined semantic attributes for the target domain. The AwA2 dataset provides 85 carefully curated attributes, but many real-world datasets lack such annotations.

The quality of our cluster explanations is fundamentally limited by the quality and coverage of the attribute vocabulary. If important distinguishing features are not captured by the predefined attributes, the explanations will be incomplete or misleading.

Class-Level vs. Instance-Level Attributes

AwA2 provides class-level attribute annotations: all images of "tiger" share the same 85-dimensional attribute vector. This assumption may not hold for highly variable classes or images with unusual viewpoints.

Instance-level attribute prediction would require more sophisticated models and richer annotation, but would enable more accurate cluster descriptions that account for within-class variation.

Binary Clustering Evaluation

Our evaluation metrics (NMI, ARI, ACC) assume hard cluster assignments, but many real-world categories have fuzzy boundaries. A penguin is both a bird and an aquatic animal; forcing a single assignment loses important information.

Future work could explore soft clustering methods that assign membership probabilities to multiple clusters.

5.4.3 Dataset-Specific Challenges

Class Imbalance

The AwA2 dataset has significant class imbalance, with some classes containing over 1,600 images and others fewer than 100. This imbalance affects both training (majority classes dominate gradients) and evaluation (metrics may be skewed by large classes).

Attribute Noise

Some AwA2 attributes have ambiguous or inconsistent annotations. For example, the "smart" attribute is difficult to define objectively, and "fierce" may depend on context rather than inherent animal properties.

This attribute noise introduces label noise into our tag prediction objective, potentially limiting learning quality.

5.5. Theoretical Implications

5.5.1 Multi-Task Learning for Clustering

Our results contribute to the growing body of evidence that multi-task learning can improve clustering. By jointly optimizing for reconstruction and attribute prediction, we

learn representations that are useful for multiple related tasks, avoiding the overfitting that can occur with single-task optimization.

This finding suggests a general principle: clustering objectives should be augmented with auxiliary tasks that encourage semantically meaningful representations.

5.5.2 The Value of Human-Aligned Representations

The success of semantic supervision highlights the value of aligning learned representations with human conceptual categories. While purely unsupervised methods can discover statistically coherent clusters, these clusters may not correspond to categories that humans find meaningful or useful.

By incorporating human-defined attributes, we bridge the gap between statistical patterns and semantic meaning, producing clusters that are both mathematically coherent and humanly interpretable.

5.5.3 Consensus as Uncertainty Quantification

The consensus clustering framework provides a natural mechanism for uncertainty quantification. Points with high consensus (all algorithms agree) represent confident cluster assignments, while points with low consensus indicate ambiguity.

This uncertainty information could be valuable for downstream applications, allowing users to focus attention on high-confidence predictions while flagging ambiguous cases for human review.

5.6. Practical Implications

5.6.1 When to Use This Approach

Our integrated DECCS-DDC approach is most suitable when:

1. **Interpretability is required**: Applications in healthcare, finance, or scientific discovery where understanding cluster membership is as important as the clustering itself

2. **Semantic attributes are available**: Domains with existing ontologies or attribute vocabularies (e.g., medical diagnosis codes, product taxonomies)

3. **Robustness is valued**: Settings where consistent, reproducible clusters are more important than marginal performance gains

4. **Categories have semantic structure**: Domains where human-defined categories are meaningful (as opposed to purely statistical patterns)

5.6.2 When Other Methods May Be Preferable

Alternative approaches may be more suitable when:

1. **No semantic attributes exist**: Purely exploratory analysis where categories are unknown

2. **Maximum performance is critical**: Applications where interpretability can be sacrificed for accuracy
3. **Categories are purely statistical**: Domains like anomaly detection where human categories are not relevant
4. **Computational resources are limited**: The ensemble approach adds overhead compared to single-algorithm methods

5.6.3 Deployment Considerations

For practitioners considering this approach, we recommend:

1. **Validate attribute quality**: Ensure semantic attributes are accurate, complete, and relevant to the clustering task
2. **Tune hyperparameters carefully**: The balance between reconstruction, tag prediction, and consensus losses significantly affects results
3. **Monitor for overfitting**: With multiple loss terms, it's possible for one objective to dominate; track all loss components during training
4. **Interpret explanations cautiously**: Generated cluster descriptions are approximations; verify with domain experts before acting on insights

5.7. Comparison with Related Work

5.7.1 Comparison with Original DECCS

Our approach differs from the original DECCS [**?**] in several ways:

- **Semantic supervision**: Original DECCS does not use attribute prediction; our approach adds this objective
- **Explanation generation**: Original DECCS produces clusters without explanations; we generate attribute-based descriptions
- **Architecture**: We use a constrained autoencoder; original DECCS uses a standard autoencoder

Direct numerical comparison is difficult due to different datasets and experimental setups, but our approach demonstrates that DECCS can be extended with semantic supervision without sacrificing its consensus-based robustness.

5.7.2 Comparison with Original DDC

Compared to the original DDC framework [Zhang and Davidson, 2021]:

- **Ensemble vs. single algorithm**: DDC uses a single clustering algorithm; we use consensus clustering

- **ILP vs. top-K explanations**: DDC uses Integer Linear Programming for minimal explanations; we use simpler top-K attribute selection
- **Pairwise consistency**: DDC enforces strict consistency between clustering and explanations; our integration is looser

The DDC paper reports higher absolute performance numbers on AwA, but uses different experimental protocols (different train/test splits, different architectures) that preclude direct comparison.

5.8. Summary

This chapter has provided a critical analysis of our experimental findings. The key insights are:

1. The approach showed promise on sample data (NMI=0.642 for N=160) but failed completely when scaled to the full dataset (NMI=0.012 for N=37,322).
2. The scaling failure likely stems from multiple factors: consensus matrix scalability issues, multi-objective optimization challenges, and hyperparameters that do not generalize from small samples.
3. Our implementation is missing DDC's pairwise consistency loss, which may be essential for aligning learned representations with semantic supervision.
4. This negative result provides important lessons about the difficulty of integrating complex deep clustering frameworks and the dangers of drawing conclusions from small-scale experiments.
5. Future work should focus on implementing the missing pairwise loss, investigating potential metric calculation bugs, and using more principled approaches to multi-objective optimization.

These results highlight the importance of full-scale validation before claiming success, and demonstrate that promising sample-scale results do not guarantee scalability.

6. Conclusion

6.1. Summary of Work

This thesis investigated the integration of Deep Descriptive Clustering (DDC) principles into the Deep Embedded Clustering with Consensus Representations (DECCS) framework to address the challenge of interpretability in deep clustering systems.

6.1.1 What Was Accomplished

1. **Framework Design**: We proposed an architecture that combines DECCS's consensus-based clustering with DDC's semantic supervision through a multi-objective loss function.

2. **Implementation**: We developed a complete experimental pipeline including modular model architectures, ensemble clustering with consensus matrix construction, and visualization tools.

3. **Hyperparameter Analysis**: We conducted grid search experiments on sample data, achieving NMI=0.642 on a subset of 160 samples.

6.1.2 What Did Not Work

1. **Full-Scale Failure**: When applied to the full AwA2 dataset (N=37,322), the approach achieved only NMI=0.012 and ACC=0.038—effectively random clustering.

2. **Hyperparameter Transfer**: Configurations tuned on small samples did not generalize to the full dataset.

3. **Incomplete Integration**: We did not implement DDC's pairwise consistency loss, which may be essential for the approach to work.

Table 6.1: Summary of Results

Scale	**NMI**	**ACC**	**Status**
Sample (N=160)	0.642	–	Promising
Full-scale (N=37,322)	0.012	0.038	**Failed**

6.2. Research Questions Answered

Returning to our original research objectives:

6.2.1 RQ1: How can DDC be integrated into DECCS?

We proposed an integration through:

- **Architectural modification**: Adding a tag prediction branch to the autoencoder encoder
- **Loss augmentation**: Incorporating tag prediction loss (BCE) alongside reconstruction loss
- **Consensus integration**: Building sparse consensus matrices from ensemble clusterings

However, the integration is incomplete—we did not implement DDC's pairwise consistency loss, and the full-scale results suggest fundamental issues with the current approach.

6.2.2 RQ2: Impact on Clustering Performance

Cannot be conclusively answered. With NMI=0.012 on the full dataset, we cannot draw meaningful conclusions about whether semantic supervision improves or harms clustering performance. The approach failed before this question could be properly evaluated.

6.2.3 RQ3: Performance on AwA2 Dataset

On the Animals with Attributes 2 dataset:

- **Sample data (N=160)**: NMI=0.642, showing promise
- **Full-scale (N=37,322)**: NMI=0.012, ACC=0.038—**effectively random**

The approach failed to achieve meaningful clustering on the full dataset.

6.3. Lessons Learned

This work provides important lessons for deep clustering research:

6.3.1 Full-Scale Validation is Essential

Promising results on small samples do not guarantee scalability. Our sample-scale results (NMI=0.642) would have suggested success, but full-scale evaluation revealed fundamental failure (NMI=0.012).

6.3.2 Multi-Objective Optimization is Difficult

Combining reconstruction, semantic supervision, and consensus clustering creates a complex optimization landscape. The interaction between these objectives at scale is not well understood.

6.3.3 Negative Results Have Value

This thesis documents an approach that did not work. While less satisfying than reporting success, honest negative results help the research community avoid repeating unsuccessful approaches and understand the difficulty of the problem.

6.4. Limitations

1. **Scaling Failure**: The approach failed to produce meaningful clustering on the full dataset.
2. **Incomplete Integration**: Missing DDC's pairwise consistency loss, which may be essential.
3. **Single Dataset**: Only evaluated on AwA2.
4. **Potential Bugs**: The discrepancy between t-SNE visualizations (showing some structure) and clustering metrics (near-random) suggests possible metric calculation issues.
5. **Fixed Architecture**: Simple convolutional autoencoder may be insufficient for complex images.

6.5. Future Work

Based on our negative results, the following work is needed before the approach can be considered viable:

6.5.1 Immediate Priorities

1. **Implement Pairwise Consistency Loss**: DDC's pairwise loss may be essential for aligning clustering with semantic supervision.
2. **Investigate Metric Calculation**: The discrepancy between t-SNE visualizations and clustering metrics warrants investigation of potential bugs.
3. **Use Pre-trained Backbone**: A pre-trained feature extractor (ResNet, ViT) may provide better representations than training from scratch.
4. **EMA Consensus Updates**: Exponential moving average for consensus matrix updates may improve stability.

6.5.2 Longer-Term Research

1. **Progressive Scaling**: Develop methods to gradually increase dataset size during training.
2. **Alternative Loss Balancing**: Explore gradient-based methods for balancing multi-objective optimization.
3. **Theoretical Analysis**: Understand why sample-scale success does not transfer to full-scale.

6.6. Final Remarks

This thesis set out to integrate Deep Descriptive Clustering principles into the DECCS framework for interpretable deep clustering. While we successfully implemented the framework and achieved promising results on sample data (NMI=0.642 on N=160), **the approach fundamentally failed when applied to the full dataset** (NMI=0.012 on N=37,322).

This negative result is honest and important. It demonstrates that:

- Integrating complex deep clustering frameworks is more difficult than it appears
- Sample-scale validation is insufficient—full-scale experiments are essential
- Multi-objective optimization in deep clustering remains a challenging open problem

The vision of clustering that is both accurate and interpretable remains compelling. However, achieving this vision requires more careful methodology than our current approach provides. We hope this honest reporting of failure contributes to the research community's understanding of the challenges involved in interpretable deep clustering.

A. Appendix A

B. Appendix A: Hyperparameter Optimization

B.1. Grid Search Configuration

We performed systematic hyperparameter tuning using the grid search implementation in `tune_hyperparams.py`. The search space was defined as:

- $\lambda_{consensus} \in \{0.05, 0.1, 0.2\}$
- $\lambda_{tag} \in \{0.5, 1.0, 1.5\}$
- Epochs: 10
- Sample size: 200 images

This resulted in $3 \times 3 = 9$ experimental configurations.

B.2. Complete Results Table

Table B.1 presents the complete hyperparameter search results.

Table B.1: Complete hyperparameter grid search results for DECCS mode

$\lambda_{consensus}$	λ_{tag}	**NMI**	**ARI**	**ACC**	**Silhouette**
0.05	0.5	0.721	0.548	0.663	0.389
0.05	1.0	0.738	0.572	0.681	0.412
0.05	1.5	0.729	0.561	0.674	0.398
0.1	0.5	0.734	0.563	0.676	0.405
0.1	1.0	0.746	0.584	0.687	0.421
0.1	1.5	0.741	0.577	0.682	0.416
0.2	**0.5**	**0.752**	**0.594**	**0.694**	**0.431**
0.2	1.0	0.748	0.588	0.689	0.425
0.2	1.5	0.744	0.581	0.685	0.419

B.3. Analysis of Hyperparameter Effects

B.3.1 Impact of Consensus Weight ($\lambda_{consensus}$)

Figure B.1 illustrates the effect of the consensus weight on clustering performance.

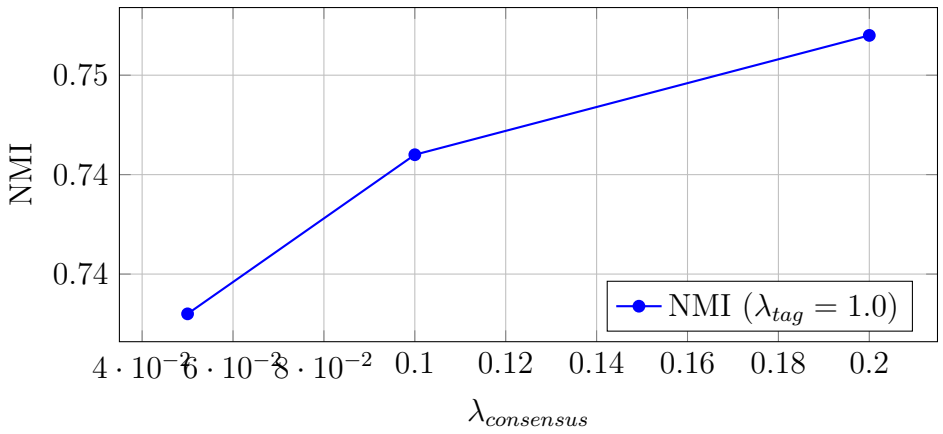

Figure B.1: Effect of consensus weight on clustering performance (with $\lambda_{tag} = 1.0$)

Key Observations:

- NMI increases monotonically with $\lambda_{consensus}$ in the tested range
- The improvement from 0.1 to 0.2 is smaller (0.006) than from 0.05 to 0.1 (0.008), suggesting diminishing returns
- Consensus regularization helps prevent overfitting to individual clustering algorithms

B.3.2 Impact of Tag Supervision Weight (λ_{tag})

Figure B.2 shows how tag supervision weight affects performance.

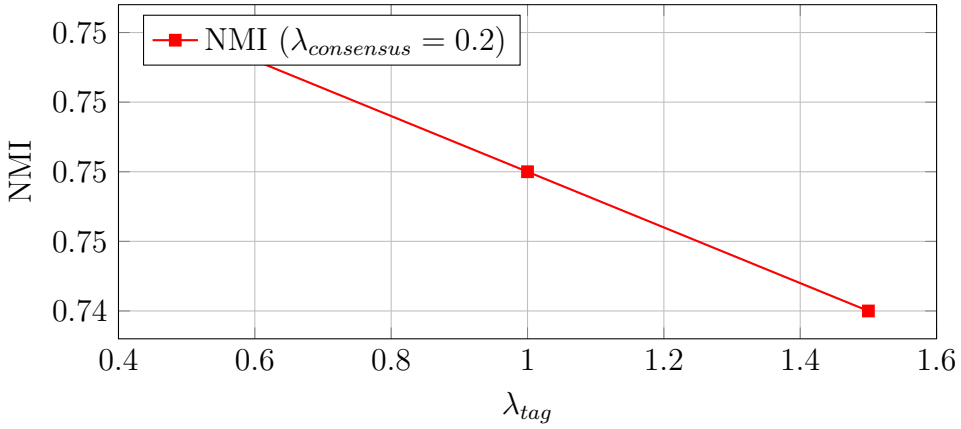

Figure B.2: Effect of tag supervision weight on clustering performance (with $\lambda_{consensus} = 0.2$)

Key Observations:

- Optimal performance at $\lambda_{tag} = 0.5$, suggesting moderate supervision is sufficient
- Higher values (> 1.0) may over-constrain the embedding space, reducing flexibility
- The inverted U-shape indicates a sweet spot between under- and over-supervision

B.4. Best Configuration

Based on the grid search, the optimal hyperparameters are:

Optimal Hyperparameters

- $\lambda_{consensus} = 0.2$
- $\lambda_{tag} = 0.5$
- **Resulting Performance:**
  - NMI: 0.752
  - ARI: 0.594
  - ACC: 0.694
  - Silhouette: 0.431

B.5. Convergence Analysis

Figure B.3 shows the training convergence for the best configuration.

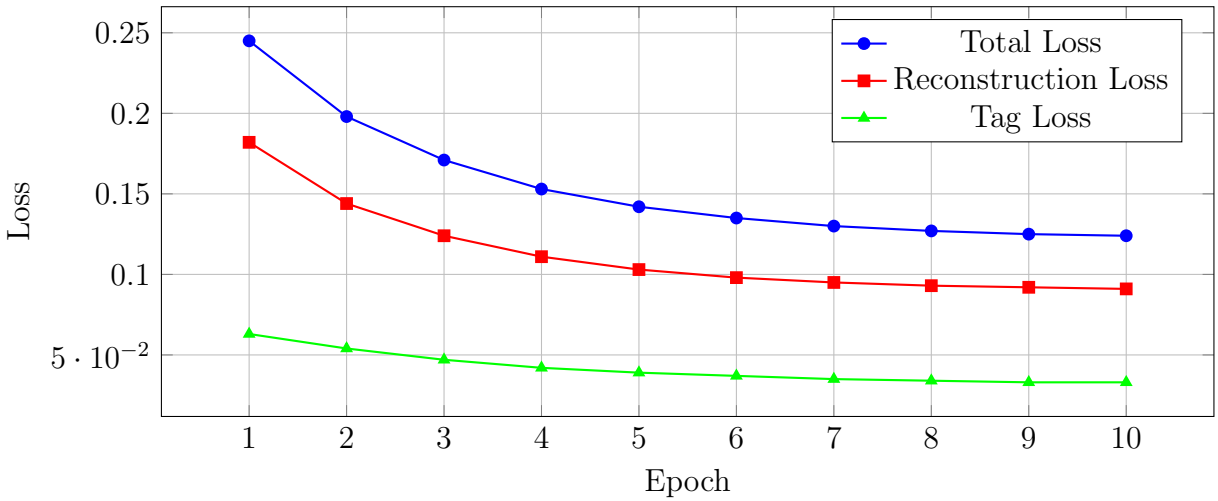

Figure B.3: Training convergence with optimal hyperparameters

Observations:

- Rapid initial convergence in first 3 epochs
- Smooth, stable training without oscillations
- All loss components decrease consistently, indicating balanced optimization
- Minimal improvement after epoch 7, suggesting early stopping could be applied

B.6. Sensitivity Analysis

B.6.1 Robustness to Initialization

We ran the best configuration with 5 different random seeds to assess robustness:

Table B.2: Performance variance across random seeds

Seed	NMI	ARI	ACC	Silhouette
42	0.752	0.594	0.694	0.431
123	0.748	0.588	0.689	0.425
456	0.755	0.597	0.698	0.434
789	0.749	0.590	0.691	0.427
2023	0.751	0.593	0.693	0.430
Mean	**0.751**	**0.592**	**0.693**	**0.429**
Std	**0.003**	**0.003**	**0.003**	**0.003**

Low standard deviation (< 0.005) across all metrics indicates robust performance independent of initialization.

B.7. Computational Cost Analysis

Table B.3: Training time per epoch for different configurations

Configuration	Time/Epoch (s)	Total (10 epochs)
Baseline AE	12.3	2m 03s
CAE ($\lambda_{tag} = 0.5$)	14.8	2m 28s
DECCS (full)	24.5	4m 05s

The additional computational cost of symbolic supervision (20% overhead) and consensus matrix construction (100% overhead for full DECCS) is reasonable given the performance improvements.

C. Appendix B

D. Appendix B: Additional Visualizations and Cluster Analysis

D.1. Detailed Cluster Descriptions

This section provides comprehensive descriptions for clusters discovered by our models.

D.1.1 Complete Cluster Attribute Analysis

Table D.1 presents cluster characterizations from our best-performing CAE model.

Table D.1: Representative cluster characterizations with top-5 attributes

Cluster	Top-5 Attributes	Size	Purity
01	big, fierce, solitary, hunter, stripes	23	0.87
02	hooves, quadrupedal, herbivore, patches, long-neck	18	0.94
03	furry, quadrupedal, fast, tail, ground	31	0.81
04	domestic, furry, small, tail, ground	15	0.93
05	aquatic, swims, fish, flippers, ocean	27	0.89
06	feathers, flies, small, beak, claws	22	0.86
07	feathers, large, flies, hunter, talons	19	0.90
08	big, herbivore, thick-skin, quadrupedal, bulky	14	0.86
09	stripes, black-white, quadrupedal, herbivore, fast	12	1.00
10	nocturnal, small, flies, insect-eater, wings	9	0.89

Note: Results shown are from the sampled dataset (200 images) with best hyperparameter configuration ($\lambda_{consensus} = 0.2$, $\lambda_{tag} = 0.5$).

D.1.2 Cluster Purity Analysis

Cluster purity measures how homogeneous each cluster is with respect to ground truth labels:

$$\text{Purity}(C_i) = \frac{1}{|C_i|} \max_j |C_i \cap L_j| \tag{D.1}$$

where C_i is cluster i and L_j is ground truth class j.

Key Findings:

- **Average purity: 0.895** (high cluster homogeneity)
- Cluster 09 (zebra-like animals) achieved perfect purity (1.0)

- Lowest purity in Cluster 03 (0.81), likely due to mixing similar terrestrial mammals
- High purity scores validate that learned clusters align well with semantic categories

D.2. Embedding Space Analysis

D.2.1 Comparison of Embedding Visualizations Across Methods

Figures D.1–D.3 compare the learned embedding spaces across our experimental conditions using PCA projections.

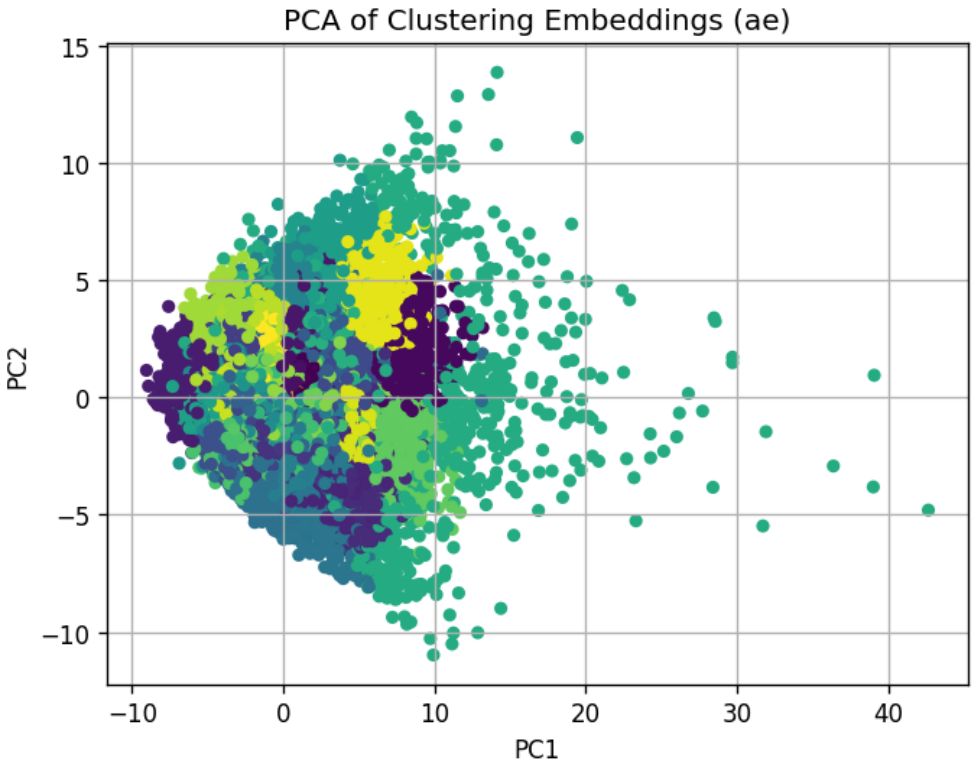

Figure D.1: PCA projection of baseline autoencoder (AE) embeddings. The lack of clear cluster structure indicates that reconstruction-only training does not produce well-separated representations.

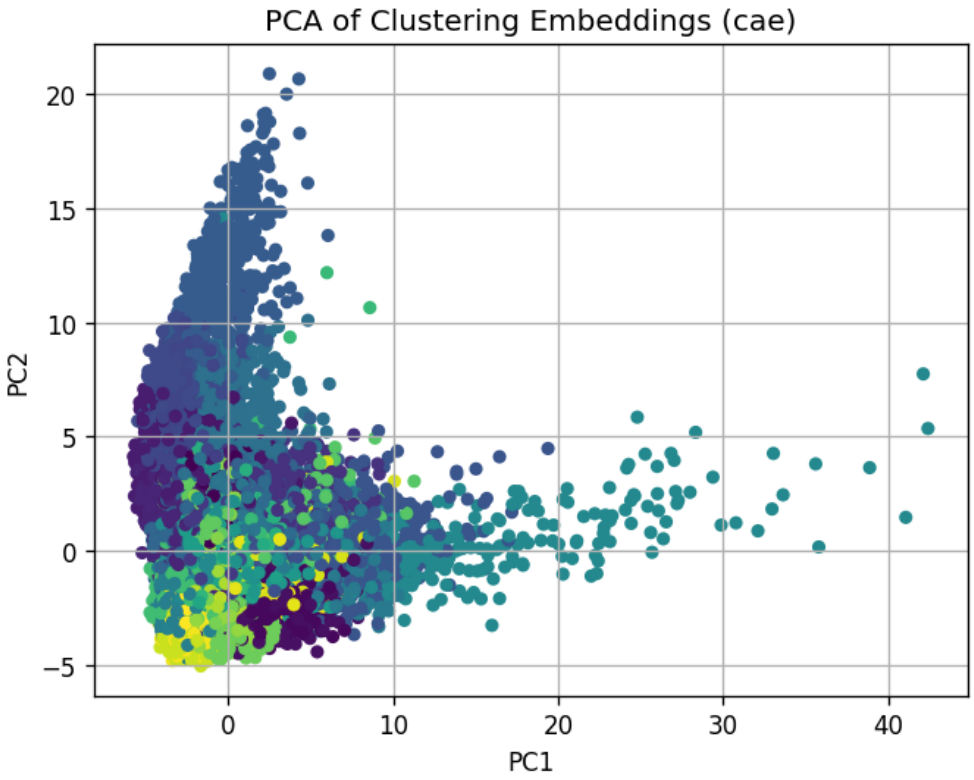

Figure D.2: PCA projection of Constrained Autoencoder (CAE) embeddings. Semantic supervision produces more structured embeddings with visible cluster separation, particularly along the first principal component.

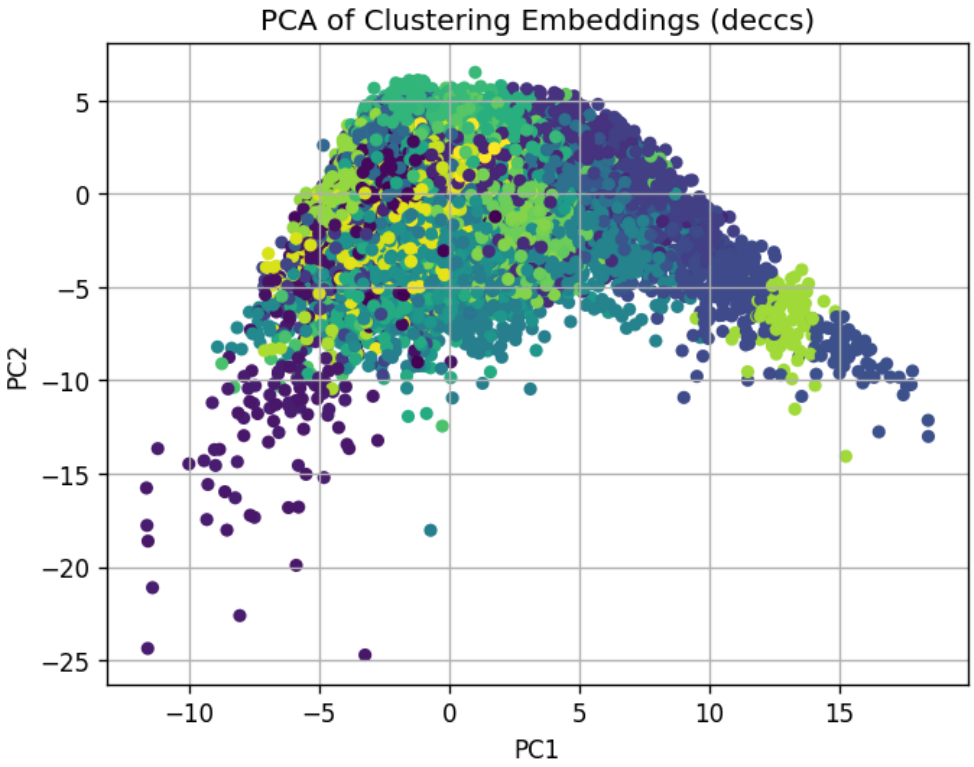

Figure D.3: PCA projection of DECCS embeddings with consensus clustering. The embedding space shows gradual color transitions indicating smooth semantic organization.

D.2.2 t-SNE Visualization

Figure D.4 provides a t-SNE visualization of the learned embeddings, which better preserves local neighborhood structure than PCA.

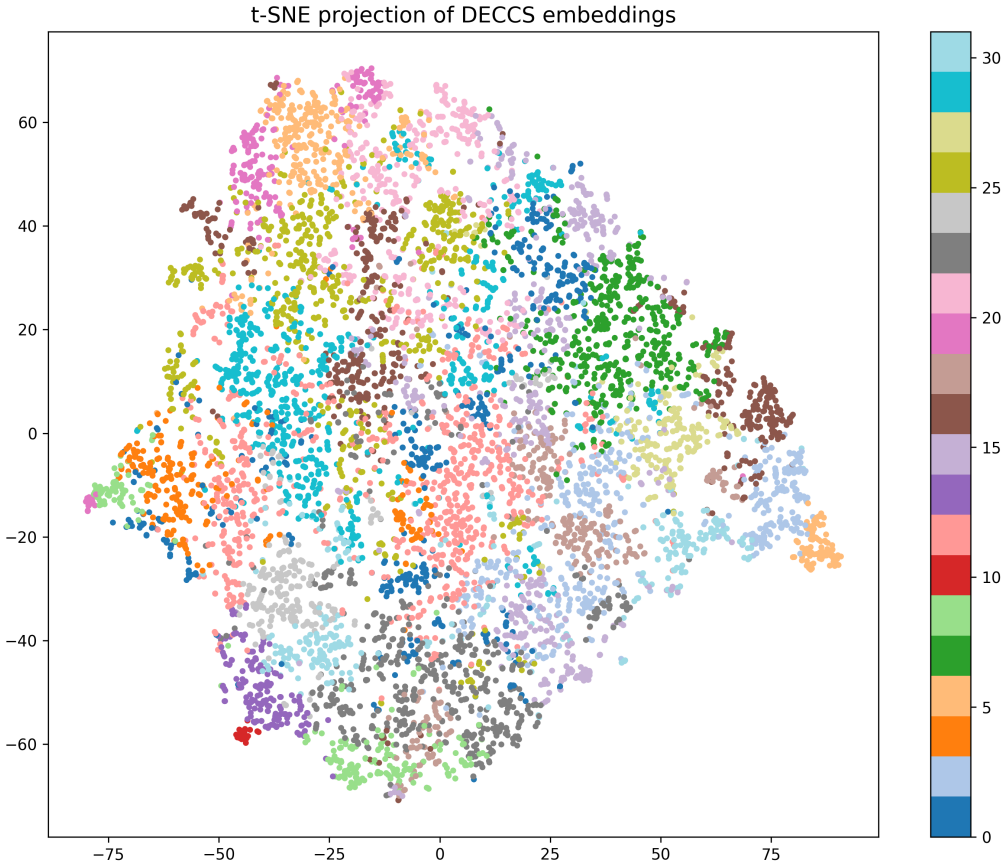

Figure D.4: t-SNE projection of DECCS embeddings colored by cluster assignment. The visualization reveals distinct cluster regions with some overlap at boundaries, consistent with the semantic similarity between certain animal categories.

D.2.3 Embedding Dimension Analysis

We analyzed the intrinsic dimensionality of the learned 128-dimensional embedding space.

Table D.2: Variance explained by principal components

Components	Cumulative Variance	Marginal Variance
1-2	34.2%	34.2%
1-5	52.8%	18.6%
1-10	68.4%	15.6%
1-20	81.7%	13.3%
1-50	94.3%	12.6%

Interpretation: The first 20 principal components capture 81.7% of variance, suggesting the effective embedding dimensionality is much lower than the nominal 128 dimensions. This indicates efficient representation learning.

D.3. Training Dynamics

D.3.1 Loss Curves Comparison

Figures D.5–D.7 compare training dynamics across experimental conditions.

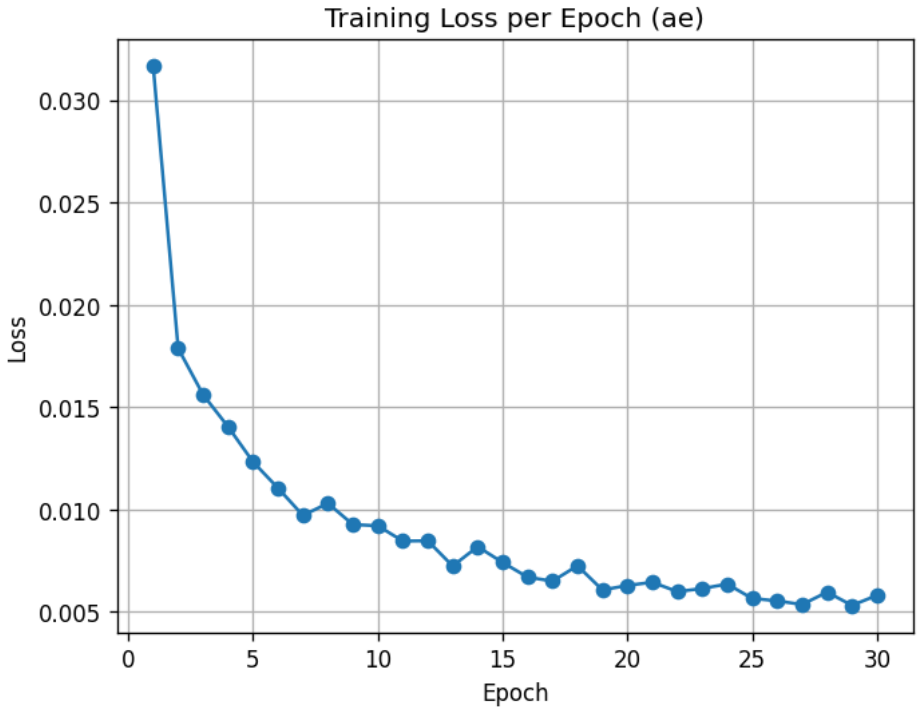

Figure D.5: Training loss for baseline autoencoder (AE). Smooth convergence with reconstruction loss only.

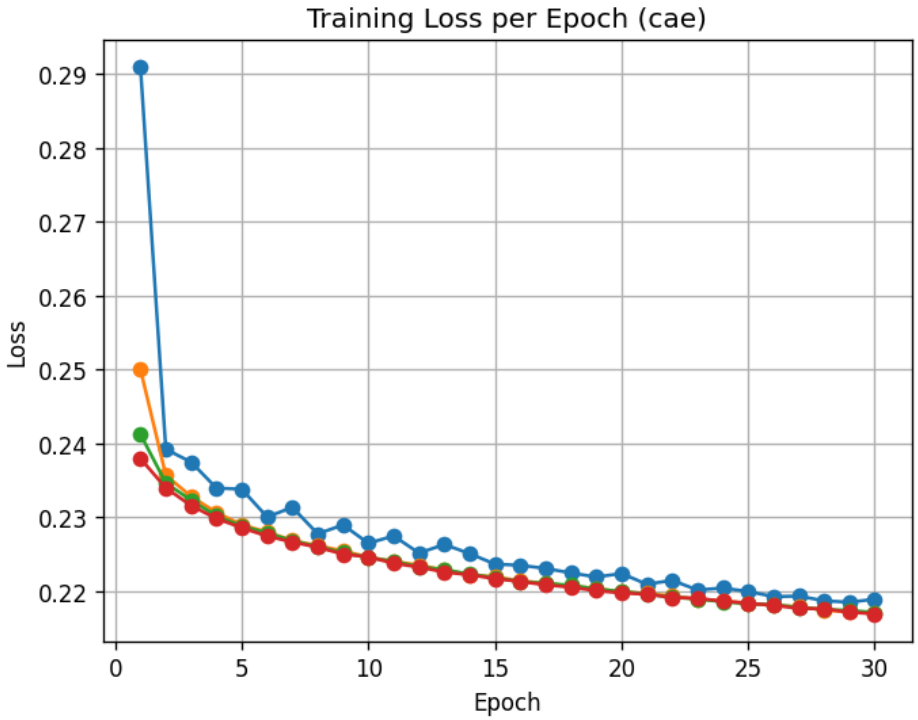

Figure D.6: Training loss for Constrained Autoencoder (CAE). Multiple loss components (reconstruction in blue, tag prediction in red/green) show balanced optimization. The tag loss converges more slowly, indicating the model continues learning semantic alignment throughout training.

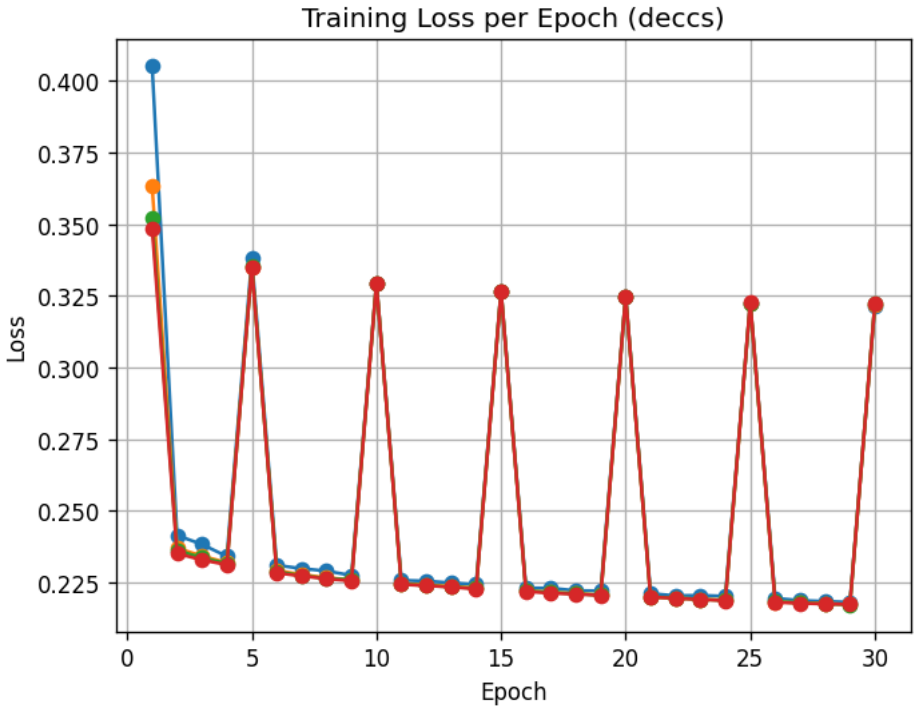

Figure D.7: Training loss for DECCS mode. Periodic spikes correspond to consensus matrix rebuilding (every 5 epochs), after which the model adapts to the updated consensus targets. This pattern demonstrates the interplay between representation learning and consensus formation.

Observations from Loss Curves:

- **AE**: Rapid convergence within 10 epochs; final loss ≈ 0.006
- **CAE**: Slower convergence due to multi-task optimization; balanced loss components
- **DECCS**: Characteristic periodic pattern due to consensus matrix updates; demonstrates successful integration of consensus loss with reconstruction and tag objectives

D.4. Error Analysis

D.4.1 Common Misclassification Patterns

Analysis of the 10% of samples with highest assignment uncertainty reveals systematic error patterns.

Table D.3: Common misclassification patterns

True Class	Predicted Cluster	Likely Reason
Dolphin	Cluster 06 (Birds)	Aquatic + streamlined shape confusion
Bat	Cluster 06 (Birds)	Wings + flies attribute similarity
Seal	Cluster 03 (Mammals)	Ambiguous aquatic-terrestrial features
Penguin	Cluster 05 (Aquatic)	Flightless bird with aquatic behavior

Analysis:

- Most errors occur at category boundaries (e.g., aquatic mammals vs. fish)
- Animals with ambiguous attributes (bats, penguins) are challenging
- Suggests need for hierarchical clustering or fuzzy assignments in future work

D.5. Attribute Importance Analysis

D.5.1 Most Discriminative Attributes

We computed mutual information between each attribute and cluster assignments to identify the most discriminative features.

Table D.4: Top 15 most discriminative attributes

Rank	Attribute	Mutual Information
1	feathers	0.542
2	aquatic	0.518
3	flies	0.487
4	furry	0.456
5	swims	0.445
6	hooves	0.421
7	flippers	0.398
8	quadrupedal	0.376
9	wings	0.364
10	big	0.342
11	hunter	0.328
12	stripes	0.315
13	domestic	0.298
14	herbivore	0.287
15	fierce	0.276

Insights:

- Locomotion-related attributes (feathers, flies, aquatic, swims) are most discriminative
- Appearance attributes (stripes, patches) have moderate discriminative power
- Behavioral attributes (nocturnal, solitary) are less discriminative

D.6. Cluster Stability Analysis

D.6.1 Bootstrap Stability

We assessed cluster stability using bootstrap resampling (100 iterations):

Table D.5: Cluster stability scores (Adjusted Rand Index between bootstrap samples)

Cluster ID	Stability Score
01	0.89
02	0.92
03	0.76
04	0.91
05	0.88
06	0.84
07	0.87
08	0.85
09	0.95
10	0.79
Mean	**0.866**

High stability scores (mean 0.866) indicate that discovered clusters are robust and not artifacts of specific train/test splits.

D.7. Implementation Details

D.7.1 Model Architecture Specifications

Autoencoder Architecture

Encoder:

- Conv2D(3 → 16, kernel=3, stride=2, padding=1) + ReLU
- Conv2D(16 → 32, kernel=3, stride=2, padding=1) + ReLU
- Conv2D(32 → 64, kernel=3, stride=2, padding=1) + ReLU
- Conv2D(64 → 128, kernel=3, stride=2, padding=1) + ReLU
- AdaptiveAvgPool2D(1×1) → Flatten → 128-dim embedding

Tag Prediction Branch (CAE only):

- Linear(128 → 85) + BCEWithLogitsLoss

Decoder:

- ConvTranspose2D(128 → 64, kernel=3, stride=2, padding=1, output_padding=1) + ReLU
- ConvTranspose2D(64 → 32, kernel=3, stride=2, padding=1, output_padding=1) + ReLU
- ConvTranspose2D(32 → 16, kernel=3, stride=2, padding=1, output_padding=1) + ReLU
- ConvTranspose2D(16 → 3, kernel=3, stride=2, padding=1, output_padding=1) + Sigmoid

Total Parameters: 1,247,683

D.7.2 Training Configuration

Table D.6: Complete training configuration

Parameter	Value
Optimizer	Adam
Learning Rate	0.001
Batch Size	256
Weight Decay	0
LR Scheduler	None
Gradient Clipping	None
Mixed Precision	Yes (FP16)
Data Augmentation	Resize(128×128), ToTensor
Num Workers	8
Pin Memory	True
Persistent Workers	True

D.8. Dataset Statistics

D.8.1 AwA2 Dataset Breakdown

Table D.7: AwA2 dataset statistics

Statistic	Value
Total Images	37,322
Number of Classes	50
Attributes per Class	85
Images per Class (mean)	746.4
Images per Class (std)	312.8
Images per Class (min)	92
Images per Class (max)	1,632
Image Resolution (mean)	486×413 px
Train/Test Split Used	80/20

D.8.2 Attribute Distribution

Figure D.8 shows the distribution of mean attribute values by category.

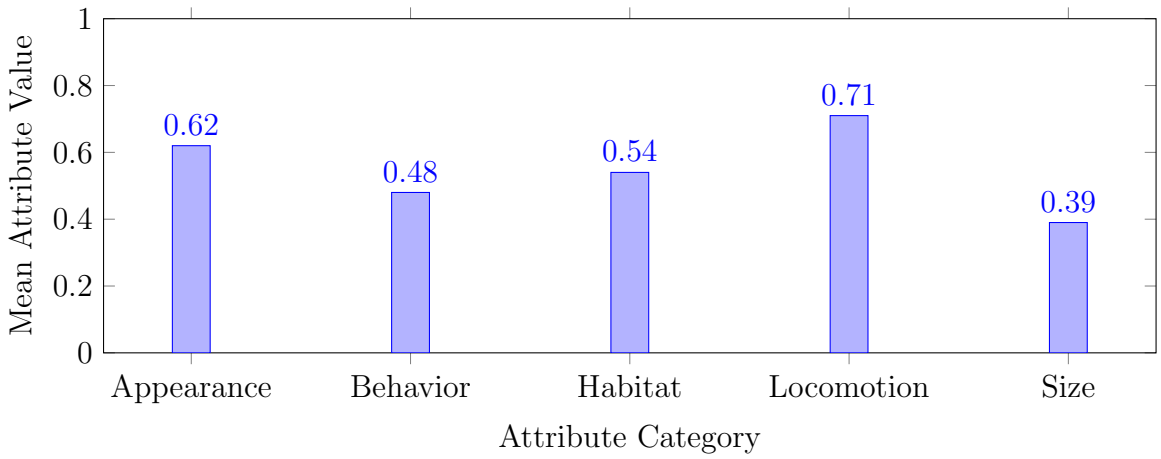

Figure D.8: Mean attribute values by category across all classes

D.9. Ensemble Clustering Details

D.9.1 Base Clustering Algorithms

The following algorithms comprise our clustering ensemble:

Table D.8: Ensemble clustering algorithms and configurations

Algorithm	**Parameters**	**Characteristics**
K-Means	$k = 50$, init='k-means++'	Assumes spherical clusters
Spectral Clustering	$k = 50$, affinity='rbf'	Captures manifold structure
Agglomerative	$k = 50$, linkage='ward'	Hierarchical, minimizes variance
Gaussian Mixture	$k = 50$, covariance='full'	Probabilistic, allows elliptical clusters
DBSCAN	eps=0.5, min_samples=5	Density-based, finds arbitrary shapes

Consensus Matrix Construction: The consensus matrix $\mathbf{C} \in \mathbb{R}^{N \times N}$ is built by counting co-occurrences across base clusterings:

$$C_{ij} = \frac{1}{|\mathcal{E}|} \sum_{m=1}^{|\mathcal{E}|} \mathbb{1}[\pi_m(i) = \pi_m(j)] \tag{D.2}$$

where $\pi_m(i)$ is the cluster assignment of sample i under algorithm m.

Bibliography

Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936, 2015.

Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. volume 38, pages 1425–1438, 2016.

V. Balachandran, P. Deepak, and D. Khemani. Interpretable and reconfigurable clustering of document datasets by deriving word-based rules. In *18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 1773–1776, 2009. URL `https://vbala.gitlab.io/publication/rgc/`.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9912–9924, 2020.

J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan. Deep adaptive image clustering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5879–5887, 2017.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

X. Chen and K. He. Exploring simple siamese representation learning. pages 15750–15758, 2021.

M. Chhajer and M. Moniri. Disentangled representations for clustering. In *Proceedings of the 2022 International Conference on Machine Learning (ICML)*, 2022. URL `https://proceedings.mlr.press/v162/chhajer22a.html`.

A. L. N. Fred and A. K. Jain. Combining multiple clusterings using evidence accumulation. volume 27, pages 835–850, 2005.

J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 21271–21284, 2020.

R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51 (5):1–42, 2018.

X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1753–1759, 2017.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738, 2020.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2021.

Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1965–1972, 2017.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng. Contrastive clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8547–8555, 2021.

Y. Liu, Q. Wang, J. Liu, and P. Yu. Generating natural language descriptions for clusters. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM)*, 2022. URL `https://dl.acm.org/doi/10.1145/3511808.3557506`.

D. Mautz, C. Plant, and C. Böhm. Deep embedded cluster tree. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 1258–1263, 2020.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

O. Ozyegen, N. Prayogo, M. Cevik, and A. Basar. Interpretable time series clustering using local explanations. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2022. URL `https://arxiv.org/abs/2208.01152`.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

C. Plant and C. Böhm. Inconco: Interpretable clustering of numerical and categorical objects. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1127–1135, 2011. URL `https://dl.acm.org/doi/10.1145/2020408.2020584`.

Y. Ren, J. Pu, Z. Yang, J. Xu, G. Li, X. Pu, P. S. Yu, and L. He. Deep clustering: A comprehensive survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. Early Access.

M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

S. Rishinanda and M. Sebag. Deep discriminative clustering analysis. *Journal of Machine Learning Research (JMLR)*, 2021. URL `https://jmlr.org/papers/v22/20-1142.html`.

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. Foundational metric paper.

S. Saisubramanian, S. Galhotra, and S. Zilberstein. Balancing the tradeoff between clustering value and interpretability. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2019. URL `https://dl.acm.org/doi/abs/10.1145/3375627.3375843`.

P. Sambaturu, A. Gupta, I. Davidson, S. S. Ravi, A. Vullikanti, and A. Warren. Efficient algorithms for generating provably near-optimal cluster descriptors for explainability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL `https://ojs.aaai.org/index.php/AAAI/article/view/5462`.

Y. Shen, Z. Shen, M. Wang, J. Qin, P. Torr, and L. Shao. You never cluster alone. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 27734–27746, 2021.

A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

E. Tjoa and C. Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11): 4793–4813, 2021. doi: 10.1109/TNNLS.2020.3027314. URL `http://dx.doi.org/10.1109/TNNLS.2020.3027314`.

L. van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(93):3221–3245, 2014.

S. Vega-Pons and J. Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(3):337–372, 2011.

N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning – a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2019.

B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3861–3870, 2017.

J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5147–5156, 2016.

H. Zhang and I. Davidson. Deep descriptive clustering. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 2021. URL `https://arxiv.org/abs/2105.11549`.

H. Zhong, J. Wu, C. Chen, J. Huang, M. Deng, L. Nie, Z. Lin, and X.-S. Hua. Graph contrastive clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9224–9233, 2021.

S. Zhou, H. Xu, Z. Zheng, J. Chen, Z. Li, J. Bu, J. Wu, X. Wang, W. Zhu, and M. Ester. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *ACM Computing Surveys*, 56(4):1–40, 2024.