Title
Modernizing Deep Unsupervised Learning with Human Experience

Author
Zhang, Hongjing

Publication Date
2022

Peer reviewed|Thesis/dissertation

Modernizing Deep Unsupervised Learning with Human Experience

By

Hongjing Zhang

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Ian Davidson, Chair

Hamed Pirsiavash

Jiawei Zhang

Committee in Charge

2022

This dissertation is dedicated to my parents.

For their endless love, support and encouragement.

Contents

List of Figures

List of Tables

Abstract

Modernizing Deep Unsupervised Learning with Human Experience

Deep unsupervised learning has emerged as a promising alternative to supervised approaches. However, supervised learning needs a tremendous amount of information in the form of annotations on specific pre-defined tasks. In contrast, human learning requires much fewer annotations and is flexible. Recent research efforts have been motivated to explore different deep unsupervised learning algorithms to leverage the massive unlabeled data for various applications that move beyond the supervised learning setting. While recent deep unsupervised learning works have shown their success in representation learning, clustering, and anomaly detection, many challenges remain unsolved. For example, how to improve the quality of learned representations used for downstream applications (*the quality of learned representations challenge*)? How to interpret and understand the deep unsupervised learning models predictions (*the explainability challenge*)? Is there any risk of bias for deep unsupervised learning applications (*the bias and fairness challenge*)?

To gain insights into the aforementioned challenges, we propose a broad range of novel techniques to address them. Each injects human-level knowledge into deep unsupervised learning. To be specific, this dissertation presents five approaches. The first two address the quality of representation challenge, the third the explainability challenge, and the last two the bias and fairness challenges. Our first formulation introduces a deep constrained clustering framework that enhances clustering performance via various constraints. Our second formulation is a self-supervised representation learning framework that automatically discovers and differentiates different categories. The third formulation simultaneously performs representation learning for clustering and describing the generated clusters with semantic tags associated with the clustered instances. Our fourth formulation proposes a novel deep fair anomaly detection architecture that uses adversarial learning to inject human fairness rules. Finally, our fifth formulation enforces disparate impact rules into deep clustering models via minimal modification learning. These methods are unified in modernizing deep unsupervised learning with different types of human guidance.

Acknowledgments

This dissertation becomes a reality with many individuals' kind support and help. I would like to express my warmest gratitude to all who have encouraged and supported me.

Foremost, I would like to express my special gratitude and thanks to my advisor, Professor Ian Davidson. With his guidance, I learn to do research and develop brilliant ideas to solve challenging problems. Thank him for imparting his knowledge and expertise in this dissertation.

Thanks to all my collaborators, Dr. Sugato Basu, Professor S.S. Ravi, and Tianyang Zhan. I would like to express my gratitude for their valuable input. Thanks for sharing your knowledge and expertise in this dissertation.

Thanks to my dissertation committee members and Ph.D. qualifying exam committee members: Professor Ian Davidson, Professor Hamed Pirsiavash, Professor Jiawei Zhang, Professor Lemon Akoglu, and Professor Xin Liu. Thank you for the inspiring comments and helpful suggestions.

My thanks also go to my colleague and people who have willingly helped me out with their abilities.

Chapter 1

Introduction

> If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, RL would be the cherry of the cake.
>
> *Yann LeCun*

Unsupervised learning algorithms in machine learning discover hidden patterns or data groupings without the need for human annotation. With the advancement of deep learning, deep unsupervised learning aims to capture rich patterns in raw data with deep networks in a label-free way. Some of the most common deep unsupervised learning models include Auto-Encoders [10], Generative Adversarial Networks [61], Self-supervised Learning [131, 32, 69]. Deep unsupervised learning has achieved huge success in several domains; for example: massive language models like GPT-3 [24] and BERT [46] have been widely used in NLP tasks; deep clustering algorithms [144, 26] have been applied on image tasks for categorization; deep unsupervised anomaly detection algorithms [163, 108] have shown their advantage in detecting abnormal instances over complex image data.

While recent deep unsupervised learning works have shown success in representation learning, clustering, and anomaly detection. Many challenges remain unsolved for deep unsupervised learning. In this dissertation, we mainly discuss three challenges of current

deep unsupervised learning:

- Quality of Learned Representation Challenge: the quality of learned representation directly determines the performance of its downstream tasks, such as classification or regression problems. The representation learned via supervised learning is still much better than unsupervised learning. The performance gap between supervised and unsupervised learning motivates us to improve the current deep unsupervised learning algorithms.

- The Explainability of Deep Unsupervised Learning Challenge: the area of explainable AI (XAI) is motivated to enhance the interpretability of complex machine learning models, especially deep learning. Arguably, XAI is more needed, different, and challenging in deep unsupervised learning. For example, most supervised learning focuses on instance-level explanation, whereas clustering explanations are usually at the model level rather than the instance level.

- Fairness Issues in Deep Unsupervised Learning Challenge: deep learning boosts the performance of unsupervised learning algorithms such as clustering and anomaly detection. Clustering and anomaly detection algorithms are commonly used in market research, social network study, and crime analysis. While deep unsupervised learning achieves good performance on those applications, recent studies on fairness reveal that these technologies can be at risk of bias. This substantial threat could undermine their entire purpose.

To address the challenges mentioned above, we propose to inject human-level knowledge and experience into deep unsupervised learning models. Unlike semi-supervised learning which uses the exact instance-level ground-truth labels for model training, we seek opportunities in learning with human experiences, which are usually more abstract but easier to acquire. The main directions in this dissertation are: (1) novel ways of integrating constraints and human knowledge to improve representation learning and clustering quality; (2) leveraging interpretable tags to enhance the explainability of deep clustering; (3) enforcing fairness rules for deep clustering and deep anomaly detection.

1.1 Our Contributions

Motivated by the challenges mentioned earlier in deep unsupervised learning, we propose leveraging human knowledge to improve the representation quality, model explainability, and fairness of deep unsupervised learning algorithms. The main contributions of this dissertation are:

- We propose a deep constrained clustering formulation that cannot only encode standard together/apart constraints but new triplet constraints, instance difficulty constraints, and cluster-level balancing constraints. We show that our formulation significantly improves the clustering performance and is robust to the random constraints' negative effects. We also show that our approach can generalize for out-of-sample predictions and scale to complex problems (Chapter 2, [154] and [159]).

- We propose a self-supervised representation learning framework that learns to discover new categories and then learns to differentiate them iteratively. We validate our learning representation on two downstream tasks, including clustering and few-shot classification. Our proposed approach achieves state-of-the-art results in both tasks (Chapter 3 [160]).

- We propose a framework to learn clustering and cluster-level explanations simultaneously. We formulate the class-level explanation problem as an Integer Linear Programming (ILP). A pairwise loss function is proposed with self-generated constraints to bridge the clustering and explanation. Empirical results on public data demonstrate that our model consistently achieves better clustering results and high-quality explanations (Chapter 4 [156]).

- We propose a new architecture for the fair anomaly detection approach (Deep Fair SVDD) and train it using an adversarial network to de-correlate the relationships between the sensitive attributes and the learned representations. Further, we propose two effective fairness measures and empirically demonstrate that existing methods are unfair. Finally, we show that our proposed approach can remove the unfairness with minimal loss of anomaly detection performance (Chapter 5 [158]).

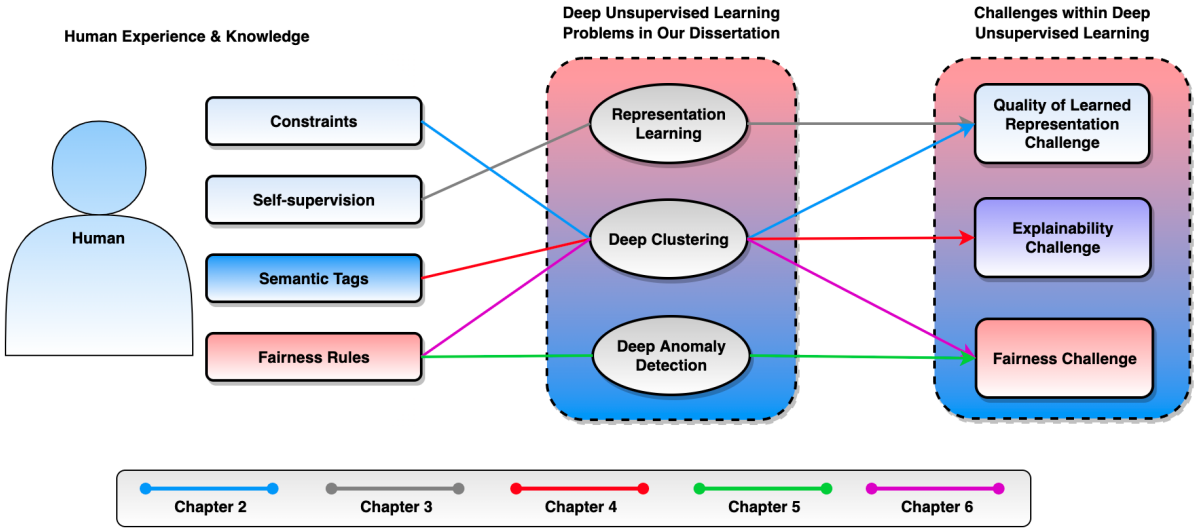


Figure 1.1: Illustration of our efforts of leveraging human knowledge to improve current deep unsupervised learning algorithms. Each chapter is represented by a colored line that connects one type of human knowledge into one deep unsupervised learning model.

- We propose a fair clustering algorithm that studies a general notion of group-level fairness. Our approach formulates the group-level fairness problem as an integer linear programming problem whose totally unimodular constraint matrix means it can be efficiently solved via linear programming. We inject the solver's result as a fairness signal into deep clustering backbones to learn fair clusters adaptively. Our framework shows promising results for various fair clustering tasks (Chapter 6 [157]).

1.2 Summary of the Dissertation

In this section, we briefly introduce the background and contributions of each chapter. The methods presented in this dissertation are unified in how they all improve upon integrating human knowledge in deep unsupervised learning algorithms. This includes three parts: 1) improving the quality of learned representation with various constraints and human-style learning; 2) enhancing the interpretability of the deep clustering model with semantic tags; 3) debiasing the deep clustering and deep anomaly detection models with human-defined fairness rules. Table 1.1 presents a high-level summary of the setting,

Chapter	Human Knowledge	Learning Setting	Main Contributions
Chapter 2 [154], [159]	Constraints	Deep Clustering	Integrating instance-level and global-level constraints into deep clustering model.
Chapter 3 [160]	Self-supervision	Representation Learning	Self-supervised representation learning via deep clustering and few-shot classification.
Chapter 4 [156]	Semantic Tags	Deep Clustering	Simultaneously clustering and explaining with semantic tags.
Chapter 5 [158]	Fairness Rules	Deep Anomaly Detection	Enforce fairness rules into deep anomaly detection via adversarial training.
Chapter 6 [157]	Fairness Rules	Deep Clustering	A general framework to ensure group-level fairness for deep clustering models.

Table 1.1: Outline of the dissertation

human knowledge, and contributions of each chapter within this dissertation. Figure 1.1 visualizes how our proposed works leverage different types of human knowledge to improve the current deep unsupervised learning algorithms.

1.2.1 A Framework for Deep Constrained Clustering

The area of constrained clustering has been extensively explored by researchers and used by practitioners. Constrained clustering formulations exist for popular algorithms such as k-means, mixture models, and spectral clustering but have several limitations. A fundamental strength of deep learning is its flexibility, and here we explore a deep learning framework for constrained clustering and in particular explore how it can extend the field of constrained clustering. We show that our framework can not only handle standard together/apart constraints (without the well-documented negative effects reported earlier)

generated from labeled side information but more complex constraints generated from new types of side information such as continuous values and high-level domain knowledge. Furthermore, we propose an efficient training paradigm that is generally applicable to these four types of constraints. We validate the effectiveness of our approach by empirical results on both image and text datasets. We also study the robustness of our framework when learning with noisy constraints and show how different components of our framework contribute to the final performance. Our source code is available at: http://github.com/blueocean92/deep_constrained_clustering.

1.2.2 A Self-Supervised Deep Learning Framework for Unsupervised Few-Shot Learning and Clustering

The need to learn a good representation is a core problem central to AI. We present a self-supervised representation learning framework and demonstrate its use for few-shot classification and clustering. Our framework can be interpreted as repeatedly discovering new categories from learned embeddings and training a new embedding function with self-supervised signals to differentiate the discovered categories. In our framework, we first discover categories from unlabeled data. Next, we post-process the previous partition results to remove outliers and derive prototypes of each category. We then construct few-shot learning tasks with previously selected data and augmented virtual data. Lastly, we iterative train the network through previous steps to learn the final representation. Our framework can considerably outperform previous baselines in unsupervised few-shot classification tasks on miniImageNet and Omniglot data sets. We also validate our learned representation on clustering tasks and demonstrate that our framework further improves upon the current deep clustering methods.

1.2.3 Deep Descriptive Clustering

Recent work on explainable clustering allows describing clusters when the features are interpretable. However, much modern machine learning focuses on complex data such as images, text, and graphs where deep learning is used, but the data's raw features are not interpretable. This paper explores a novel setting for performing clustering on complex

data while simultaneously generating explanations using interpretable tags: We propose deep descriptive clustering that performs sub-symbolic representation learning on complex data while generating explanations based on symbolic data. We then form good clusters by maximizing the mutual information between empirical distribution on the inputs and the induced clustering labels for clustering objectives. We generate explanations by solving integer linear programming that generates concise and orthogonal descriptions for each cluster. We allow the explanation to inform better clustering by proposing a novel pairwise loss with self-generated constraints to maximize the clustering and explanation module's consistency. Experimental results on public data demonstrate that our model outperforms competitive baselines in clustering performance while offering high-quality cluster-level explanations.

1.2.4 Towards Fair Deep Anomaly Detection

Anomaly detection aims to find instances that are considered unusual and is a fundamental problem of data science. Recently, deep anomaly detection methods were shown to achieve superior results particularly in complex data such as images. Our work focuses on deep one-class classification for anomaly detection which learns a mapping only from the normal samples. However, the non-linear transformation performed by deep learning can potentially find patterns associated with social bias. The challenge of adding fairness to deep anomaly detection is to ensure fair and correct anomaly predictions simultaneously. This paper proposes a new architecture for the fair anomaly detection approach (Deep Fair SVDD) and trains it using an adversarial network to de-correlate the relationships between the sensitive attributes and the learned representations. This differs from fairness typically added as a regularizer or a constraint. Further, we propose two effective fairness measures and empirically demonstrate that existing deep anomaly detection methods are unfair. We also show that our proposed approach can largely remove unfairness with minimal loss on the anomaly detection performance. Lastly, we conduct an in-depth analysis to show the strength and limitations of our proposed model, including parameter analysis, feature visualization, and run-time analysis.

1.2.5 Fair Learning for Deep Clustering

Deep clustering has shown the potential to learn a strong representation and hence better clustering performance than traditional methods such as k-means and spectral clustering. However, this strong representation learning ability may make the clustering unfair by discovering surrogates for protected information which our experiments empirically show. To correct such unfairness, we propose a new fair learning algorithm that studies a general notion of group-level fairness in clustering for both binary and multi-state protected status variables (PSVs). Our approach formulates the group-level fairness problem as an integer linear programming problem whose totally unimodular constraint matrix means it can be efficiently solved via linear programming. Then, we inject the solver's result as a fairness signal into deep clustering backbones to learn fair clusters adaptively. Experimental results on real-world datasets demonstrate that our model consistently outperforms state-of-the-art fair clustering algorithms. Furthermore, our framework shows promising results for novel fair clustering tasks including flexible fairness constraints, multi-state PSVs, and predictive (out of sample) clustering.

Chapter 2

A Framework for Deep Constrained Clustering

2.1 Introduction

Constrained clustering has a long history in machine learning with many standard algorithms being adapted to be constrained [12] including Expectation-maximization (EM) [11], K-Means [136] and spectral methods [140]. The addition of constraints generated from ground truth labels allows a semi-supervised setting to increase accuracy [136] when measured against the ground truth labeling.

However, there are several limitations in these methods, and one purpose of this paper is to explore how deep learning can make advances to the field beyond what other methods have. In particular, we find that existing non-deep formulations of constrained clustering have the following four limitations:

Limited Constraints and Side Information. Constraints are limited to simple together/apart constraints typically generated from labels. However, in some domains, experts may more naturally give guidance at the cluster level, generate constraints from continuous side-information or even complex sources such as ontologies. To address these deficiencies fundamentally new types of constraints are required.

Negative Effect of Constraints. For some algorithms though constraints improve performance when *averaged* over many constraint sets, *individual* constraint sets produce results worse than using no constraints [43] as reported in our earlier paper. However, as

a practitioner typically has only one constraint set, constrained-clustering use can be "hit or miss".

Intractability and Scalability Issues. Iterative algorithms that directly solve for clustering assignments run into problems of intractability [40]. Relaxed formulations (i.e. spectral methods [95, 140]) require solving a full rank eigendecomposition problem which takes $O(n^3)$. The deep learning paradigm has shown to be scalable for large data sets and we explore if this is the case for deep constrained clustering.

Assumption of Good Features. A critical requirement for existing constrained clustering algorithms is the need for good features or a similarity function. The end-to-end learning benefits of deep learning will be explored to determine if they are useful for constrained clustering.

Though deep clustering with constraints has many potential benefits to overcome these limitations, it is not without its challenges. Our major contributions in this paper are summarized as follows:

- We propose a deep constrained clustering formulation that can not only encode standard together/apart constraints but a range of new constraint types. Example types include triplet constraints, instance difficulty constraints, and cluster-level balancing constraints (see section 2.3).

- In addition to generating constraints from instances' labels, we show how our framework can take advantage of continuous side information and an ontology graph to generate triplet constraints and how to learn from multiple constraints simultaneously (see Section 2.5).

- Deep constrained clustering overcomes a long term issue we reported earlier [43] with constrained clustering of significant practical implications: overcoming the negative effects of individual constraint sets.

- We show how the benefits of deep learning such as scalability and end-to-end learning translate to our deep constrained clustering formulation.

- Our method outperforms standard non-deep constrained clustering methods even though these methods are given the auto-encoder embedding provided to our approach (see Table 2.2).

- We show the robustness of our proposed framework (see Section 2.6.4.6) and demonstrate the scalability of our framework on large-scale data set (see Section 2.6.4.8).

- We conduct ablation study and analyze the contributions of each component within our algorithm (see Section 2.6.4.7).

This paper is an extension of our previous work [155] with the following additions: 1) we show our framework can not only work with constraints generated from ground truth labels but also work with constraints that are generated from an ontology graph which is a weaker form of guidance; 2) whereas previously we conducted deep constrained clustering with only one type of constraints at each run in previous work, in this paper we extend our algorithm to learn with pairwise and triplet constraints simultaneously; 3) we experimentally visualize the learning process of our framework and show how our framework overcomes the negative effects of constraints; 4) we analyze the effects of noisy constraints on our framework and show the robustness of our model; 5) we analyze each component's contributions within our framework (i.e., initialization, clustering module, constraints learning module) using an ablation study.

The rest of the paper is organized as follows: First, we introduce the related work in section 2.2. We then propose four forms of constraints in section 2.3 and introduce how to train the clustering network with these constraints in section 2.4. We then discuss the new way of generating constraints and how to learn multiple types of constraints together in section 2.5. In section 2.6, we compare our approach to previous baselines and demonstrate the effectiveness of new types of constraints and also perform a detailed analysis of our proposed framework. Next, we discuss the limitations of current work and conclude in section 2.7.

2.2 Related Work

Constrained clustering. Constrained clustering is an important area, and there is a large body of work that shows how *side information* can improve the clustering performance [135, 136, 145, 19, 140]. Here the side information is typically labeled data which is used to generate *pairwise* together/apart constraints used to partially reveal the ground truth clustering to help the clustering algorithm. Such constraints are easy to encode in matrices and enforce in procedural algorithms though not with its challenges. In particular, we showed [43] clustering performance improves with larger constraint sets when **averaged** over many constraint sets generated from the ground truth labeling. However, for a significant fraction (just not the majority) of these constraint sets, the clustering performance is *worse* than using no constraint set. We recreated some of these results in Table 2.2.

Moreover, side information can exist in different forms beyond labels (i.e., continuous data), and domain experts can provide guidance beyond pairwise constraints. Some work in the supervised classification setting [82, 118, 117, 62] seek alternatives such as relative/triplet guidance, but to our knowledge, such information has not been explored in the non-hierarchical clustering setting. Complex constraints for hierarchical clustering have been explored [9, 30] but these are tightly limited to the hierarchical structure (i.e., x must be joined with y before z) and not directly translated to non-hierarchical (partitional) clustering.

Deep Clustering. Motivated by the success of deep neural networks in supervised learning, unsupervised deep learning approaches are now being explored [144, 81, 149, 64, 76, 57, 26, 65, 3, 120, 80, 66]. There are approaches [149, 76, 26, 120] which learn an encoding that is suitable for a clustering objective first and then applied an external clustering method. Our work builds upon the most direct setting [144, 64] which encodes one self-training objective and finds the clustering allocations for all instances within one neural network.

Deep Clustering with Pairwise Constraints. Most recently, the semi-supervised clustering networks with pairwise constraints have been explored: [73] uses pairwise con-

straints to enforce small divergence between similar pairs while increasing the divergence between dissimilar pairs assignment probability distributions. However, this approach did not leverage the unlabeled data, hence requires lots of labeled data to achieve good results. Fogel et al. proposed an unsupervised clustering network [55] by self-generating pairwise constraints from the mutual KNN graph and extends it to semi-supervised clustering by using labeled connections queried from the human. However, this method cannot make out-of-sample predictions and requires user-defined parameters for generating constraints from the mutual KNN graph.

2.3 Our Deep Constrained Clustering Framework

Here we outline our proposed framework for deep constrained clustering. Our method of adding constraints to and training deep learning can be used for deep clustering methods so long as the network has a k unit output indicating the degree of cluster membership. Here we choose the popular deep embedded clustering method (DEC [144]). We sketch this method first for completeness.

2.3.1 Deep Embedded Clustering

We choose to apply our constraints formulation to the deep embedded clustering method DEC [144], which starts with pre-training an autoencoder ($x_i = g(f(x_i))$ but then removes the decoder. The remaining encoder ($z_i = f(x_i)$) is then fine-tuned by optimizing an objective which takes first z_i and converts it to a soft allocation vector of length k which we term $q_{i,j}$ indicating the degree of belief instance i belongs to cluster j. Then q is self-trained to produce p a unimodal "hard" allocation vector which allocates the instance to primarily only one cluster. We now overview each step.

Conversion of z to Soft Cluster Allocation Vector q. Here DEC takes the similarity between an embedded point z_i and the cluster centroid u_j measured by Student's t-distribution [96]. Note that v is a constant as $v = 1$ and q_{ij} is a soft assignment:

$$q_{ij} = \frac{(1 + ||z_i - \mu_j||^2/v)^{-\frac{v+1}{2}}}{\sum_{j'} (1 + ||z_i - \mu_{j'}||^2/v)^{-\frac{v+1}{2}}} \tag{2.1}$$

Conversion of Q To Hard Cluster Assignments P. The above normalized sim-

ilarities between embedded points and centroids can be considered as soft cluster assignments Q. However, we desire a target distribution P that better resembles a hard allocation vector, p_{ij} is defined as:

$$p_{ij} = \frac{{q_{ij}}^2 / \sum_i q_{ij}}{\sum_{j'} ({q_{ij'}}^2 / \sum_i q_{ij'})} \tag{2.2}$$

Loss Function. Then, the algorithm's loss function is to minimize the distance between P and Q as follows. Note this is a form of self-training as we are trying to teach the network to produce unimodal cluster allocation vectors.

$$\ell_C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.3}$$

The DEC method requires the initial centroids given (μ) to calculate Q are "representative". The initial centroids are set using k-means clustering. However, there is no guarantee that the clustering results over an auto-encoders embedding yield a good clustering. We believe that constraints can help overcome this issue which we test later.

2.3.2 Different Types of Constraints

To enhance the clustering performance and allow for more types of interactions between human and clustering models, we propose four types of guidance which are pairwise constraints, instance difficulty constraints, triplet constraints, cardinality, and give examples of each. As traditional constrained clustering methods put constraints on the final clustering assignments, our proposed approach constrains the q vector which is the soft assignment. A core challenge when adding constraints is to allow the resultant loss function to be differentiable so we can derive backpropagation updates.

2.3.2.1 Pairwise Constraints

Pairwise constraints (must-link and cannot-link) are well studied [12] and we showed they are capable of defining any ground truth set partitions [40]. Here we show how these pairwise constraints can be added to a deep learning algorithm. We encode the loss for must-link constraints set ML as:

$$\ell_{ML} = - \sum_{(a,b) \in ML} \log \sum_j q_{aj} * q_{bj} \tag{2.4}$$

Similarly loss for cannot-link constraints set CL is:

$$\ell_{CL} = -\sum_{(a,b)\in CL} \log\left(1 - \sum_{j} q_{aj} * q_{bj}\right) \tag{2.5}$$

Intuitively speaking, the must-link loss prefers instances with same soft assignments and the cannot-link loss prefers the opposite cases.

2.3.2.2 Instance Difficulty Constraints

A challenge with self-learning in deep learning is that if the initial centroids are incorrect, the self-training can lead to poor results. Here we use constraints to overcome this by allowing the user to specify which instances are easier to cluster (i.e., they belong strongly to only one cluster) and ignoring difficult instances (i.e., those that belong to multiple clusters strongly).

We encode user supervision with an $n \times 1$ constraint vector M. Let $M_i \in [-1, 1]$ be an instance difficulty indicator, $M_i > 0$ means the instance i is easy to cluster, $M_i = 0$ means no difficulty information is provided and $M_i < 0$ means instance i is hard to cluster. The loss function is formulated as:

$$\ell_I = \sum_{t\in\{M_t<0\}} -M_t \sum_{j} {q_{tj}}^2 - \sum_{s\in\{M_s>0\}} M_s \sum_{j} {q_{sj}}^2 \tag{2.6}$$

The instance difficulty loss function aims to encourage the easier instances to have sparse clustering assignments but prevents the difficult instances having sparse clustering assignments. The absolute value of M_i indicates the degree of confidence in difficulty estimation. This loss will help the model training process converge faster on easier instances and increase our model's robustness towards difficult instances.

2.3.2.3 Triplet Constraints

Although pairwise constraints are capable of defining any ground truth set partitions from labeled data [40], in many domains, no labeled side information exists or strong pairwise guidance is not available. Thus we seek triplet constraints, which are weaker constraints that indicate the relationship within a triple of instances. Given an anchor instance a, positive instance p and negative instance n we say that instance a is more similar to p

than to n. The loss function for all triplets $(a, p, n) \in T$ can be represented as:

$$\ell_T = \sum_{(a,p,n)\in T} \max(d(q_a, q_n) - d(q_a, q_p) + \theta, 0) \tag{2.7}$$

where $d(q_a, q_b) = \sum_j q_{aj} * q_{bj}$ and $\theta > 0$. The larger value of $d(q_a, q_b)$ represents larger similarity between a and b. The variable θ controls the gap distance between positive and negative instances. ℓ_T works by pushing the positive instance's assignment closer to anchor's assignment and preventing negative instance's assignment being closer to anchor's assignment.

2.3.2.4 Global Size Constraints

Experts may more naturally give guidance at a cluster level, previous work [57] explored adding uniform distribution assumption to regularize the clustering model. Here we explore clustering size constraints in our framework, which means each cluster should be approximately the same size. Denote the total number of clusters as k, total training instances number as n, the global size constraints loss function is:

$$\ell_G = \sum_{c\in\{1,..k\}} (\sum_{i=1}^{n} q_{ic}/n - \frac{1}{k})^2 \tag{2.8}$$

Our global constraints loss function works by minimizing the distance between the expected cluster size and the actual cluster size. The actual cluster size is calculated by averaging the soft-assignments. To guarantee the effectiveness of global size constraints, we need to assume that the batch size should be large enough to calculate the cluster sizes during our mini-batch training. A similar loss function can be used (see section 2.3.4) to enforce other cardinality constraints on the cluster composition such as upper and lower bounds on the number of people with a certain property.

2.3.3 Preventing Trivial Solution

In our framework the proposed must-link constraints we mentioned before can lead to trivial solution that all the instances are mapped to the same cluster. Previous deep clustering method [149] have also met this problem. To mitigate this problem, we combine the reconstruction loss with the must-link loss to learn together. Denote the encoding

network as $f(x)$ and decoding network as $g(x)$, the reconstruction loss for instance x_i is:

$$\ell_R = \ell(g(f(x_i)), x_i) \tag{2.9}$$

where ℓ is the least-square loss: $\ell(x, y) = ||x - y||^2$.

2.3.4 Extensions to High-level Domain Knowledge-Based Constraints

The constraints proposed in the previous section are typically generated from instance labels or comparisons. A benefit of our framework is the ability to include more complex constraints and we now describe examples of constraints for higher-level domain knowledge.

Cardinality Constraints For Fairness. Cardinality constraints [38] allow expressing requirements on the number of instances that satisfy some conditions in each cluster. Assume we have n people and want to split them into k groups but wish to minimize disparate impact with respect to gender. Then an example cardinality constraint is to enforce each group should have the same number of males and females. We assume each instance has a protected status variable (PSV) which we called P. Then the cardinality constraints can be formulated as:

$$\ell_{Cardinality} = \sum_{c\in\{1,..k\}} (\sum_{P_i=M} q_{ic}/n - \sum_{P_j=F} q_{jc}/n)^2 \tag{2.10}$$

For upper-bound and lower-bound based cardinality constraints [38], we use the same setting as previously described, now the constraint changes as for each party group we need the number of males to range from L to U. Then we can formulate this as:

$$\ell_{CardinalityBound} = \sum_{c\in\{1,..k\}} (\min(0, \sum_{P_i=M} q_{ic} - L)^2 + \max(0, \sum_{P_i=M} q_{ic} - U)^2) \tag{2.11}$$

Logical Combinations of Constraints via Dynamic Addition. Apart from cardinality constraints, complex logic constraints can also be used to enhance the expressive power of existing constraints. For example, if two instances x_a and x_b are in the same cluster then instances x_i and x_j must be in different clusters ($\texttt{Together}(x_a, x_b) \rightarrow$

`Apart`(x_i, x_j)). This can be achieved in our framework as we can dynamically adding cannot-link constraint $CL(x_i, x_j)$ once we check the soft assignment q of x_a and x_b.

Consider a Horn form constraint such as $r \wedge s \wedge t \rightarrow u$. Denote $r = ML(x_a, x_b)$, $s = ML(x_c, x_d)$, $t = ML(x_e, x_f)$ and $u = CL(x_g, x_h)$. By forward passing the instances within r, s, t to our deep constrained clustering model, we can obtain the soft assignment values of these instances. By checking the satisfying results based on $r \wedge s \wedge t$, we can decide whether to enforce cannot-link loss $CL(x_g, x_h)$.

2.4 Putting It All Together - Efficient Training Strategy

Our training strategy consists of two training branches and effectively has two ways of creating mini-batches for training. For instance-difficulty or global-size constraints, we treat their loss functions as addictive losses to the clustering branch so that no new branch needs to be created. For pairwise or triplet constraints, we build another output branch and train the whole network in an alternating fashion. We treat these two groups of constraints differently for a principled reason. For pairwise and triplet constraints, we have explicit constraints on instances and the composition of the clusters. This can be (and often is) contradictory (i.e., incompatible) with the clustering loss. This is indeed something we showed in our ECML 2006 paper [43], where we showed that pairwise constraints could hurt clustering performance. However, since the instance level and group level constraints are guidance not explicitly on specific instances assignments, they can be folded into the clustering loss.

Loss Branch for Instance Constraints. In deep learning, it is common to add loss functions defined over the same output units. In the Improved DEC method [64], the clustering loss ℓ_C and reconstruction loss ℓ_R were added together. To this, we add the instance difficulty loss ℓ_I. This effectively adds guidance to speed up training convergence by identifying "easy" instances and increase the model's robustness by ignoring "difficult" instances. Similarly, we treat the global size constraints loss ℓ_G as an additional additive loss. All instances whether or not they are part of triplet or pairwise constraints are

trained through this branch and the mini-batches are created randomly.

Loss Branch For Complex Constraints. Our framework uses more complex loss functions as they define constraints on pairs and even triples of instances. Thus we create another loss branch that contains pairwise loss ℓ_P or triplet loss ℓ_T to help the network tune the embedding which satisfies these stronger constraints. For each constraint *type* we create a mini-batch consisting of only those instances having that type of constraint. For each *example* of a constraint type, we feed the constrained instances through the network, calculate the loss, calculate the change in weights but do not adjust the weights. We sum the weight adjustments for all constraint examples in the mini-batch and then adjust the weights. Hence our method is an example of batch weight updating as is standard in DL for stability reasons. The whole training procedure is summarized in Algorithm 1.

2.5 Generating Constraints from an Ontology Graph and Learning with Multiple Types of Constraints Simultaneously

In most constrained clustering works, the constraints are normally generated from ground truth labels. For example, the pairwise constraints can be generated from labeled instances by random picking each pair of points and checking the labels; the triplet constraints can be generated based on the latent embeddings' distance among the triplet (i.e., the positive instance should be close to the anchor in latent embedding space while the negative instance will be further). However, to get a good latent embedding, we still need a large amount of ground-truth labels to learn the representation. Here we seek different sources to generate constraints to add in richer side information from humans into the clustering framework to derive better clustering results.

Generating Constraints from an Ontology Graph. In this section, we propose a new empirical strategy to generate triplet constraints based on ontology graphs. The class label names (i.e., `Sneaker`) can be used to place an instance in the ontology. The ontology graph (knowledge graph) represents a collection of interlinked descriptions of entities.

Algorithm 1 Deep Constrained Clustering Framework

Input: X: data, m: maximum epochs , k: number of clusters, N: total number of batches and N_C: total number of constraints batches.

Output: latent embeddings Z, cluster assignment S.

Train the stacked denosing autoencoder to obtain Z

Initialize centroids μ via k-means on embedding Z.

for $epoch = 1$ **to** m **do**

  for $batch = 1$ **to** N **do**

    Calculate ℓ_C via Eqn (2.3), ℓ_R via Eqn (2.9).

    Calculate ℓ_I via Eqn (2.6) or ℓ_G via Eqn (2.8).

    Calculate total loss as $\ell_C + \ell_R + \{\ell_I || \ell_G\}$.

    Update network parameters based on total loss.

  end for

  for $batch = 1$ **to** N_C **do**

    Calculate ℓ_P via Eqn (2.4, 2.5) or ℓ_T via Eqn (2.7).

    Update network parameters based on $\{\ell_P || \ell_T\}$.

  end for

  Forward pass to compute Z and $S_i = \text{argmax}_j\, q_{ij}$.

end for

Here we choose to use WordNet [1] as the ontology graph, WordNet groups English words into sets of synonyms called synsets; it also provides short definitions and usage examples and records a number of relations among these synonym sets or their members. WordNet can thus be seen as a combination of dictionary and thesaurus. We have visualized the WordNet hierarchy structure for Fashion MNIST (the data set we will use for experiments in section 2.6, it consists of a training set of 60000 examples and a test set of 10000 examples. Each example is a 28-by-28 grayscale image, associated with a label from 10 classes.) in Figure 2.1.

We measure the similarity between different classes based on the shortest path that

[1] https://wordnet.princeton.edu/

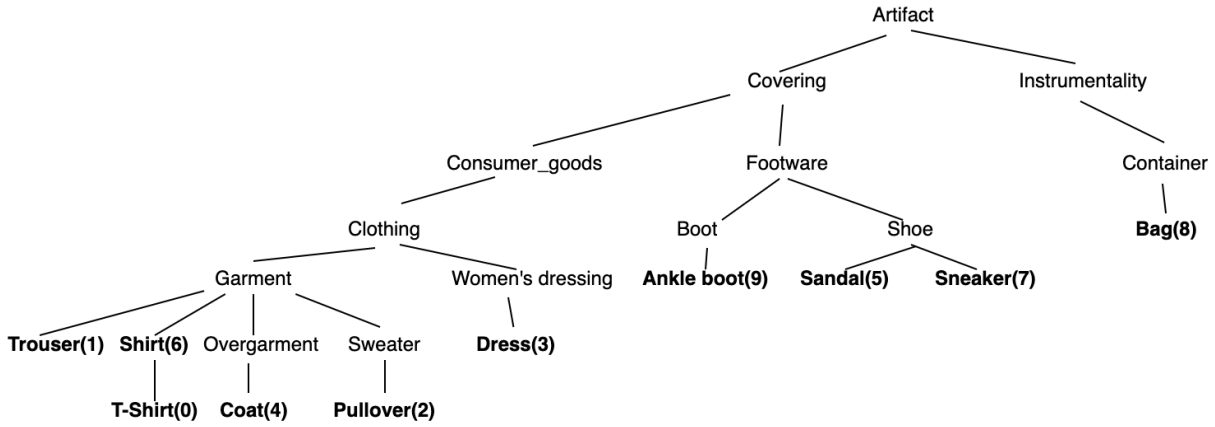


Figure 2.1: WordNet hierarchy of Fashion MNIST data set. The bolded classes are classes we will cluster upon.

connects the two classes. Given class C_i and C_j and the shortest path d_{ij} between C_i and C_j, the similarity is $sim(C_i, C_j) = \frac{1}{d_{ij}+1}$. Note that the similarity value ranges from $(0, 1]$, and the larger value represents a larger similarity. Given an anchor instance a, positive instance p and negative instance n, we wish the class similarity between a and p to be as large as possible and the similarity between a to n to be as small as possible. To satisfy these requirements we can set the positive threshold θ_p to enforce the $sim(a, p) > \theta_p$, similarly we can set negative threshold θ_n to ensure both $sim(a, n) < \theta_n$ and $sim(p, n) < \theta_n$. Further we require $\theta_n < \theta_p$. Note that when θ_p equals 1, the triplet constraints will be equivalent as one Cannot-link constraint and two Must-link constraints since both the anchor and positive instance are from the same class. With proper positive and negative thresholds, we can generate triplet constraints from a set of labeled points with WordNet knowledge.

Learning with Multiple Types of Constraints Simultaneously. It is natural to wish to take advantage of all the generated constraints, even if they are different types. Here we study learning with pairwise constraints and triplet constraints together as they are most common and useful. We motivate the need to learn these two types of constraints together in Figure 2.2.

Given adequate pairwise constraints, the model can find correct clusters as can be seen from the right-hand side in Figure 2.2, but the latent semantic similarity relationships

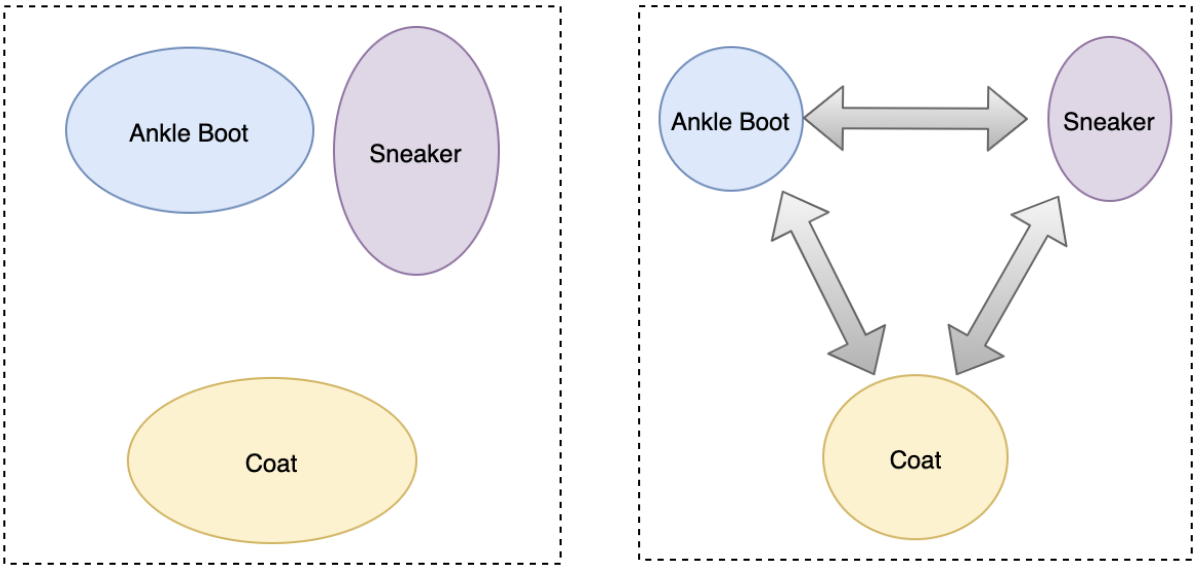


Figure 2.2: Examples of the ideal embedding (left-hand) learnt from an ontology using both triplets and pairwise constraints and the embedding learnt from just pairwise constraints (right-hand). Note in the later the three clusters are far apart from each other because the cannot-links (grey arrows) between these clusters will push them as far as possible which contradicts the ideal data embedding that ankle boots and sneakers are semantically similar.

have been destroyed. Directly applying triplet constraints to this case may end up with a reasonable semantic latent space, but the circles and triangles may be overlapping without cannot-links. To learn a better embedding, we aim to add triplet constraints together with pairwise constraints to the clustering framework. In this case, we have both pairwise constraints between these three classes, as well as the triplet constraints. For example, in Figure 2.2 the anchor and positive classes are ankle boots and sneakers whilst the negative class is coat.

Our algorithm 1 can be naturally extended to learn multiple types of constraints at the same time. Specifically, we prepare the pairwise constraints and triplet constraints with N_p and N_t batches in advance and then optimize the clustering loss ℓ_C, pairwise loss ℓ_P and triplet loss ℓ_T in an iterative way. The entire learning process is detailed in Algorithm 2.

Algorithm 2 Learning with Pairwise and Triplet Constraints

Input: X: data, m: maximum epochs , k: number of clusters, N: total number of batches and N_p: total number of pairwise constraints batches, N_t: total number of triplet constraints batches.

Output: latent embeddings Z, cluster assignment S.

Train the stacked denosing autoencoder to obtain Z

Initialize centroids μ via k-means on embedding Z.

for $epoch = 1$ **to** m **do**

  for $batch = 1$ **to** N **do**

    Calculate ℓ_C via Eqn (2.3), ℓ_R via Eqn (2.9).

    Calculate total loss as $\ell_C + \ell_R$.

    Update network parameters based on total loss.

  end for

  for $batch = 1$ **to** N_p **do**

    Calculate ℓ_P via Eqn (2.4, 2.5).

    Update network parameters based on ℓ_P.

  end for

  for $batch = 1$ **to** N_t **do**

    Calculate ℓ_T via Eqn (2.7).

    Update network parameters based on ℓ_T .

  end for

  Forward pass to compute Z and $S_i = \text{argmax}_j\, q_{ij}$.

end for

2.6 Experiments

All data and code used to perform these experiments are available online (http://github.com/blueocean92/deep_constrained_clustering) to help with reproducibility. In our experiments, we aim to address the following questions:

- How does our end-to-end deep clustering approach using traditional pairwise con-

straints compare with traditional constrained clustering methods? The latter is given the same auto-encoding representation Z used to initialize our method. (see Table 2.2)

- Are the new types of constraints we create for the deep clustering method useful in practice? (see Section 2.6.4.1, 2.6.4.3, 2.6.4.5)
- Is our end-to-end deep constrained clustering method more robust to the well known negative effects of constraints we published earlier [43]? How our learned embedding overcomes the negative effects of constraints? (see Section 2.6.4.2)
- How the model performs with constraints generated from ontologies? (see Section 2.6.4.4)
- How is the proposed model's robustness towards noisy constraints? (see Section 2.6.4.6)
- How do the different components of our approach contribute to our final performance? (see our Ablation study in Section 2.6.4.7)
- How is the scalability of our proposed framework? (see Section 2.6.4.8)

2.6.1 Datasets

To study the performance and generality of different algorithms, we evaluate the proposed method on two image datasets and one test dataset:

MNIST: Consists of 70000 handwritten digits of 28-by-28 pixel size. The digits are centered and size-normalized in our experiments [90].

FASHION-MNIST: A Zalando's article images-consisting of a training set of 60000 examples and a test set of 10000 examples. Each example is a 28-by-28 grayscale image, associated with a label from 10 classes.

REUTERS-10K: This dataset contains English news stories labeled with a category tree [91]. To be comparable with the previous baselines, we used 4 root categories:

`corporate/industrial`, `government/social`, `markets` and `economics` as labels and excluded all documents with multiple labels. We randomly sampled a subset of 10000 examples and computed TF-IDF features on the 2000 most common words.

2.6.2 Evaluation Metric

We adopt standard metrics for evaluating clustering performance which measure how close the clustering found is to the ground truth result. Specifically, we employ the following two metrics: normalized mutual information (**NMI**) [125, 148] and clustering accuracy (**Acc**) [148]. For data point x_i, let l_i and c_i denote its true label and predicted cluster respectively. Let $l = (l_1, ...l_n)$ and similarity $c = (c_1, ...c_n)$. **NMI** is defined as:

$$\mathbf{NMI}(l, c) = \frac{\mathbf{MI}(l, c)}{max\{H(l), H(c)\}}$$

where $\mathbf{MI}(l, c)$ denotes the mutual information between l and c, and H denotes their entropy. The **Acc** is defined as:

$$\mathbf{Acc}(l, c) = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{n}$$

where m ranges over all possible one-to-one mappings between clusters and labels. The optimal assignment of m can be computed using the Kuhn-Munkres algorithm [101]. Both metrics are commonly used in the clustering literature and with higher values indicating better clustering results. By using them together we get a better understanding of the effectiveness of the clustering algorithms.

2.6.3 Implementation Details

Basic Deep Clustering Implementation. To be comparable with deep clustering baselines, we set the encoder network as a fully connected multilayer perceptron with dimensions $d - 500 - 500 - 2000 - 10$ for all datasets, where d is the dimension of input data(features). The decoder network is a mirror of the encoder. All the internal layers are activated by the ReLU [102] nonlinearity function. For a fair comparison with baseline methods, we used the same greedy layer-wise pre-training strategy to calculate the auto-encoders embedding. To initialize clustering centroids, we run k-means with 20 restarts and select the best solution. We choose Adam optimizer with an initial learning rate

of 0.001 for all the experiments. We adopt standard metrics for evaluating clustering performance, which measures how close the clustering found is to the ground truth result. Specifically, we employ the following two metrics: normalized mutual information (**NMI**) [125, 148] and clustering accuracy (**Acc**) [148]. In our baseline comparisons, we use IDEC [64], a non-constrained improved version of DEC published recently.

Pairwise Constraints Experiments. We randomly select pairs of instances and generate the corresponding pairwise constraints between them. To ensure transitivity, we calculate the transitive closure over all must-linked instances and then generate entailed constraints from the cannot-link constraints [40]. Since our loss function for must-link constraints is combined with reconstruction loss, we use grid search and set the penalty weight for must-link as 0.1.

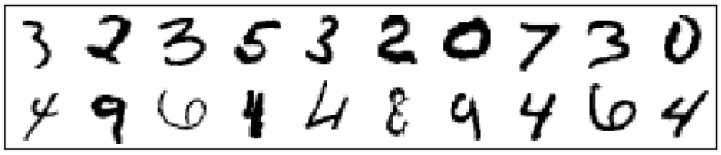


Figure 2.3: Example of instance difficulty constraints. Top row shows the "easy" instances and second row shows the "difficult" instances.

Instance Difficulty Constraints Experiments. To simulate human-guided instance difficulty constraints, we use k-means as a weak base learner and mark all the incorrectly clustered instances as difficult with confidence 0.1, we also mark the correctly classified instances as accessible instances with confidence 1. In Figure 2.3, we give some example difficulty constraints found using this method.

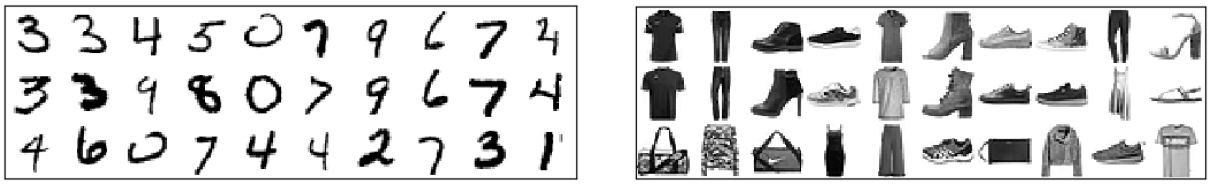


Figure 2.4: Examples of the generated triplet constraints for MNIST and Fashion. The three rows for each plot shows the anchor instances, positive instances and negative instances correspondingly.

Triplet Constraints Experiments. Triplet constraints can state that instance i

is more similar to instance j than instance k. To simulate human guidance on triplet constraints, we randomly select n instances as anchors (i); for each anchor, we randomly select two instances (j and k) based on the similarity between the anchor. The similarity is calculated as the euclidian distance d between two instances pre-trained embedding. The pre-trained embedding is extracted from our deep clustering network trained with 100000 pairwise constraints. Figure 2.4 shows the generated triplets constraints. Through grid search we set the triplet loss margin $\theta = 0.1$.

Global Size Constraints Experiments. We apply global size constraints to MNIST and Fashion datasets since they satisfy the balanced size assumptions. The total number of clusters is set to 10, and each class has the same number of instances.

2.6.4 Experimental Results

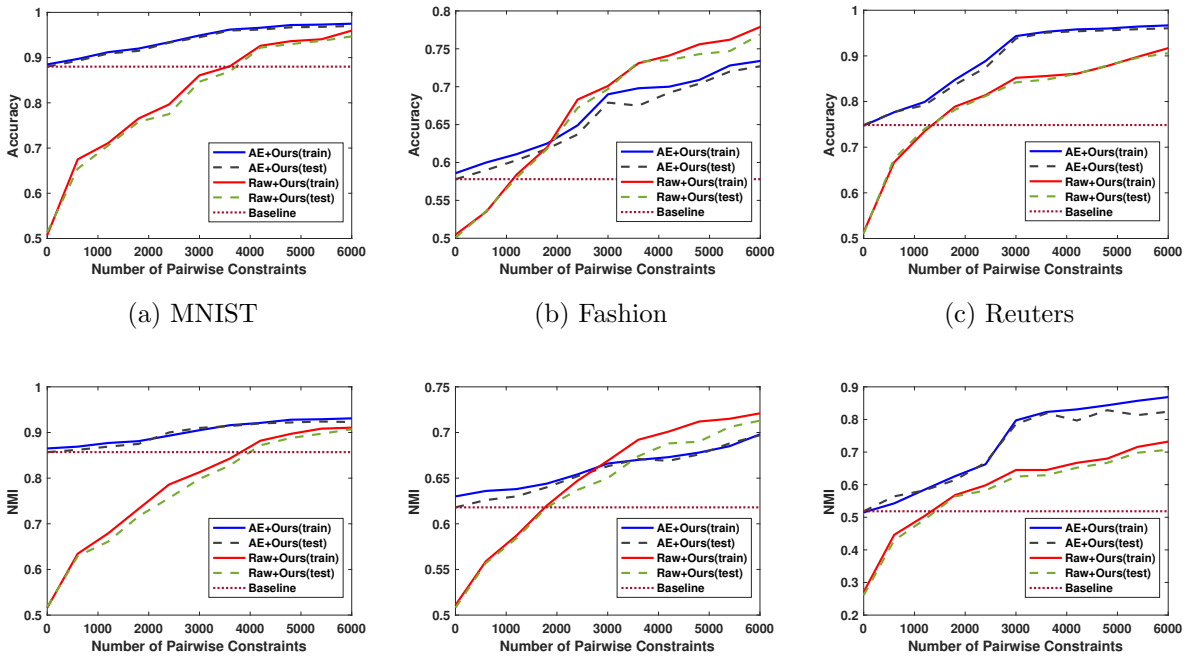
2.6.4.1 Experiments on instance difficulty.

	MNIST	Fashion	Reuters
Acc(%)	88.29 ± 0.05	58.74 ± 0.08	75.20 ± 0.07
NMI(%)	86.12 ± 0.09	63.27 ± 0.11	54.16 ± 1.73
Epoch	87.60 ± 12.53	77.20 ± 11.28	12.90 ± 2.03

	MNIST	Fashion	Reuters
Acc(%)	91.02 ± 0.34	62.17 ± 0.06	78.01 ± 0.13
NMI(%)	88.08 ± 0.14	64.95 ± 0.04	56.02 ± 0.21
Epoch	29.70 ± 4.25	47.60 ± 6.98	9.50 ± 1.80

Table 2.1: Left table shows baseline results for Improved DEC [64] averaged over 20 trials. Right table lists experiments using instance difficulty constraints (mean $\pm$ std) averaged over 20 trials.

In Table 2.1, we report the average test performance of the deep clustering framework without any constraints on the left. In comparison, we report the average test performance of deep clustering framework with instance difficulty constraints on the right, and we find the model learned with instance difficulty constraints outperforms the baseline method in all datasets. This is to be expected as we have given the algorithm more information than the baseline method, but it demonstrates our method can make good use of this extra information. What is unexpected is the effectiveness of speeding up the learning process and will be the focus of future work.



(a) MNIST (b) Fashion (c) Reuters

(d) MNIST (e) Fashion (f) Reuters

Figure 2.5: Clustering accuracy and NMI on training test sets for different number of pairwise constraints. AE means an autoencoder was used to seed our method. The horizontal maroon colored baseline shows the IDEC's [64] test set performance.

2.6.4.2 Experiments on pairwise constraints

We randomly generate 6000 pairs of constraints which are small fractions of possible pairwise constraints for MNIST (0.0002%), Fashion (0.0002%), and Reuters (0.006%). Recall the DEC method is initialized with auto-encoder features. To better understand the contribution of pairwise constraints, we have tested our method with both auto-encoders features and raw data. As can be seen from Figure 2.5: the clustering performance improves consistently as the number of constraints increases in both settings. Moreover, with just 6000 pairwise constraints, the performance on Reuters and MNIST increased significantly especially for the setup with raw data. We also notice that learning with raw data in Fashion achieves a better result than using autoencoder's features. This shows that the autoencoder's features may not always be suitable for DEC's clustering objective. Overall our results show pairwise constraints can help reshape the representation and improve the clustering results.

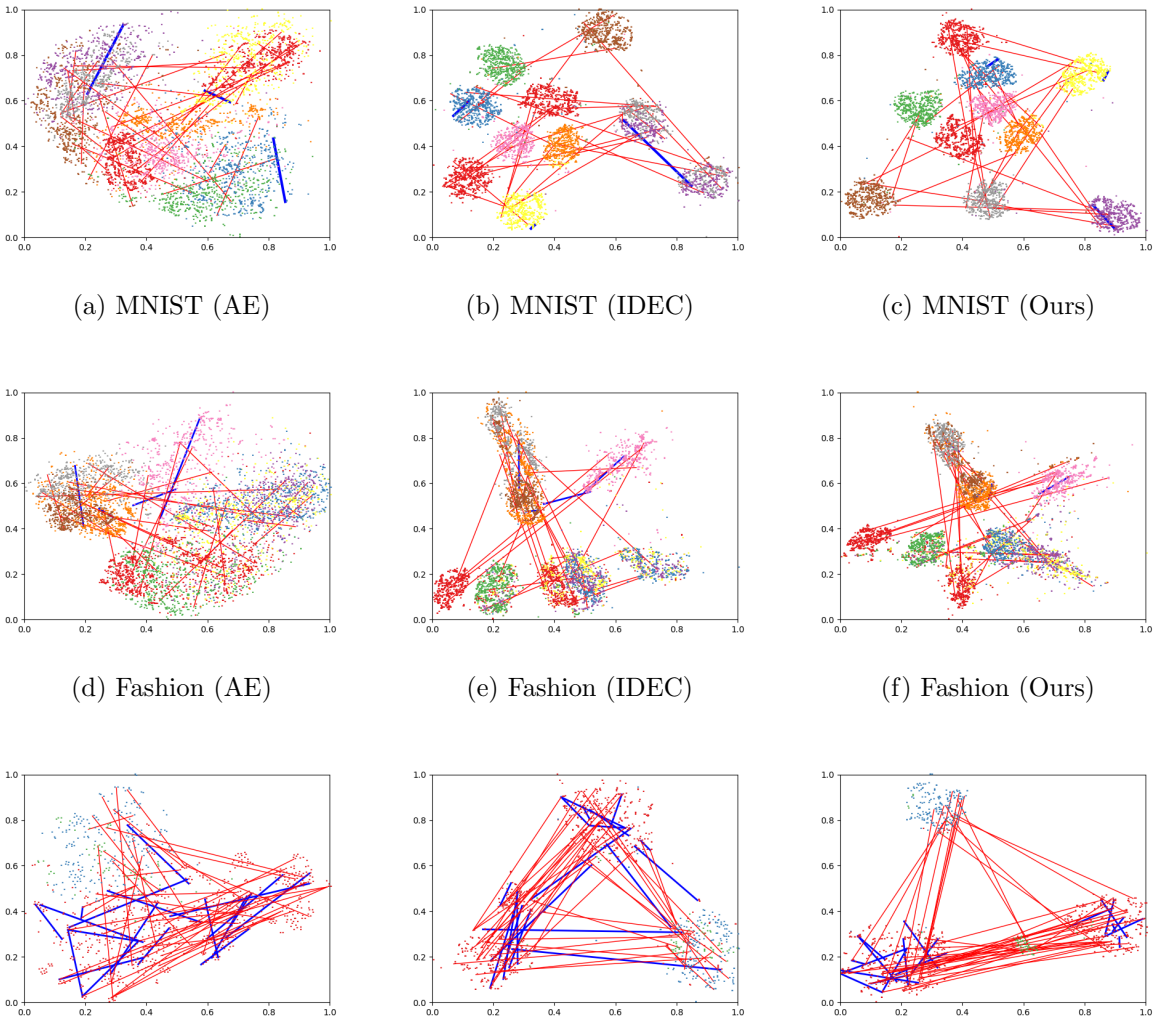
We also compare the results with recent work [73]: our approach(autoencoders features) outperforms the best clustering accuracy reported for MNIST by a margin of 16.08%, 2.16% and 0.13% respectively for 6, 60, and 600 samples/class. Unfortunately, we can't make a comparison with Fogel's algorithm [55] due to an issue in their code repository.

	Flexible CSP*	COP-KMeans	MPCKMeans	Ours
MNIST Acc	0.628 ± 0.07	0.816 ± 0.06	0.846 ± 0.04	$\mathbf{0.963 \pm 0.01}$
MNIST NMI	0.587 ± 0.06	0.773 ± 0.02	0.808 ± 0.04	$\mathbf{0.918 \pm 0.01}$
Negative Ratio	19%	45%	11%	**0 %**
Fashion Acc	0.417 ± 0.05	0.548 ± 0.04	0.589 ± 0.05	$\mathbf{0.681 \pm 0.03}$
Fashion NMI	0.462 ± 0.03	0.589 ± 0.02	0.613 ± 0.04	$\mathbf{0.667 \pm 0.02}$
Negative Ratio	23%	27%	37%	**6 %**
Reuters Acc	0.554 ± 0.07	0.712 ± 0.04	0.763 ± 0.05	$\mathbf{0.950 \pm 0.02}$
Reuters NMI	0.410 ± 0.05	0.478 ± 0.03	0.544 ± 0.04	$\mathbf{0.815 \pm 0.02}$
Negative Ratio	28%	73%	80%	**0 %**

Table 2.2: Pairwise constrained clustering performance (mean ± std) averaged over 100 constraints sets. Due to the scalability issues we apply flexible CSP with downsampled data(3000 instances and 180 constraints). Negative ratio is the fraction of times using constraints resulted in poorer results than not using constraints. See Figure 2.6 and text for an explanation why our method performs well.

Negative Effects of Constraints. Our earlier work [43] showed that for traditional constrained clustering algorithms, that the addition of constraints *on average* helps clustering but many individual constraint sets can hurt performance in that performance is worse than using **no** constraints. Here we recreate these results even when these classic methods use auto-encoded representations. In Table 2.2, we report the average performance with 3600 randomly generated pairwise constraints. For each dataset, we randomly generated 100 sets of constraints to test the negative effects of constraints [43]. In each run, we fixed the random seed and the initial centroids for k-means based methods. For

each method, we compare its performance between the constrained version to the unconstrained version. We calculate the negative ratio, which is the fraction of times that the unconstrained version produced better results than the constrained version. As can be seen from the table, our proposed method achieves significant improvements than traditional non-deep constrained clustering algorithms [136, 19, 140].



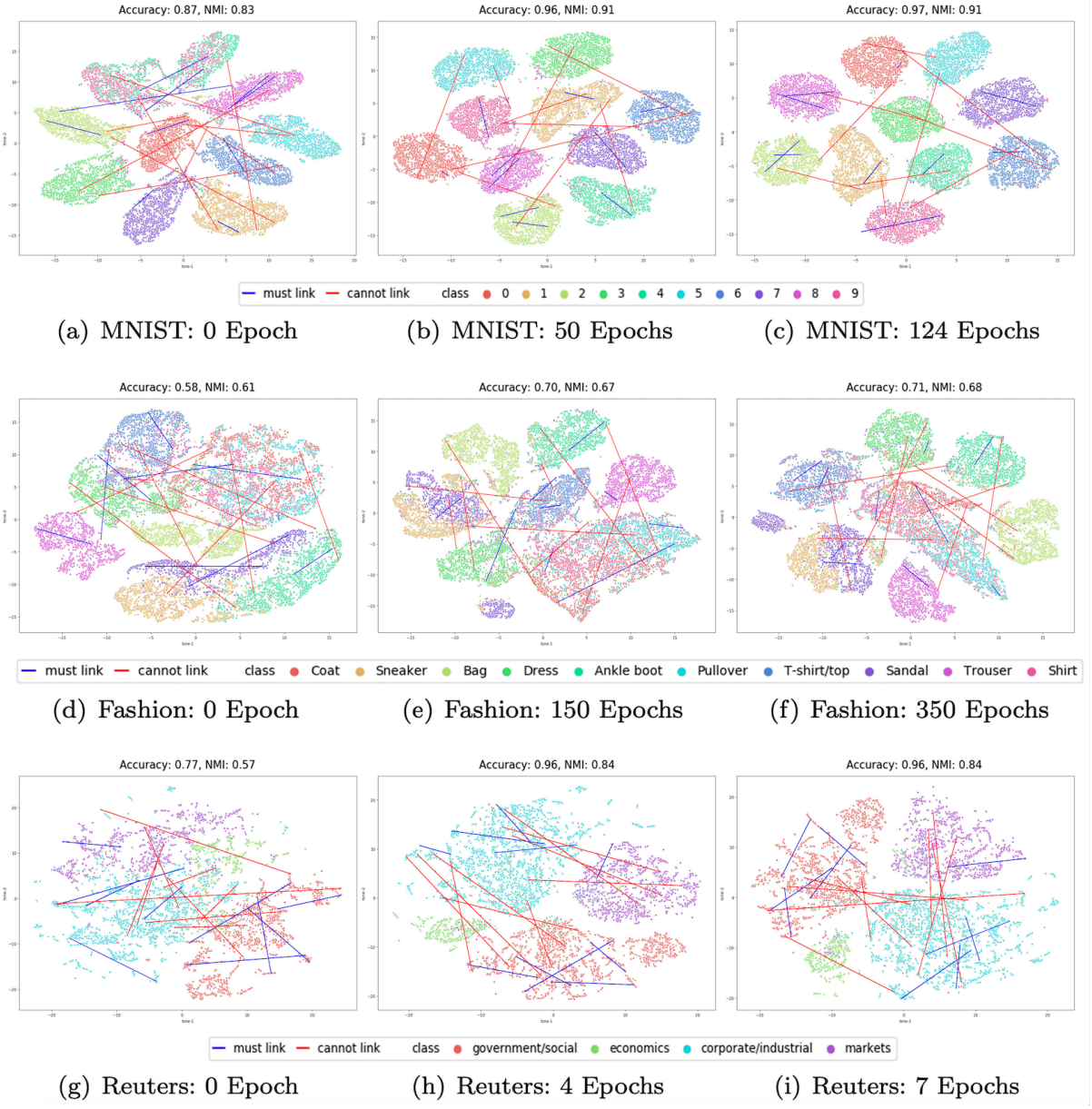
(a) MNIST (AE) (b) MNIST (IDEC) (c) MNIST (Ours)

(d) Fashion (AE) (e) Fashion (IDEC) (f) Fashion (Ours)

(g) Reuters (AE) (h) Reuters (IDEC) (i) Reuters (Ours)

Figure 2.6: We visualize (using t-SNE) the latent representation for a subset of instances and pairwise constraints, we visualize the same instances and constraints for each row. The red lines are cannot-links and blue lines are must-links.

To understand why our method was robust to variations in constraint sets, we vi-



(a) MNIST: 0 Epoch (b) MNIST: 50 Epochs (c) MNIST: 124 Epochs

(d) Fashion: 0 Epoch (e) Fashion: 150 Epochs (f) Fashion: 350 Epochs

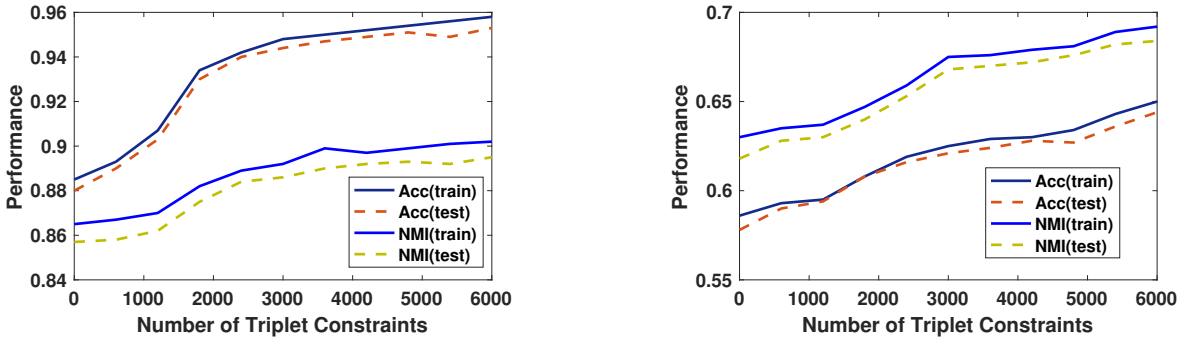
(g) Reuters: 0 Epoch (h) Reuters: 4 Epochs (i) Reuters: 7 Epochs

Figure 2.7: The embedding for a subset of instances and inconsistent pairwise constraints after several training epochs

sualized the embeddings learned. Figure 2.6 shows the embedded representation of a random subset of instances and its corresponding pairwise constraints using t-SNE and the learned embedding z. Based on Figure 2.6, we can see the autoencoders embedding is noisy, and lot's of constraints are inconsistent based on our earlier definition [43]. Further, we visualize the IDEC's latent embedding and find out the clusters are better separated.

However, the inconsistent constraints still exist (blue lines across different clusters and redlines within a cluster); these constraints tend to have negative effects on traditional constrained clustering methods. Finally, for our method's results we can see the clusters are well separated, the must-links are well satisfied (blue lines are within the same cluster), and cannot-links are well satisfied (red lines are across different clusters). Hence we can conclude that end-to-end-learning can address these negative effects of constraints by simultaneously learning a representation that is consistent with the constraints and clustering the data. This result has profound practical significance as practitioners typically only have one constraint set to work with.

To fully understand how our constrained clustering model finds a new representation to satisfy those constraints, we have visualized the latent embeddings during the training process in Figure 2.7.



(a) MNIST(Triplet)

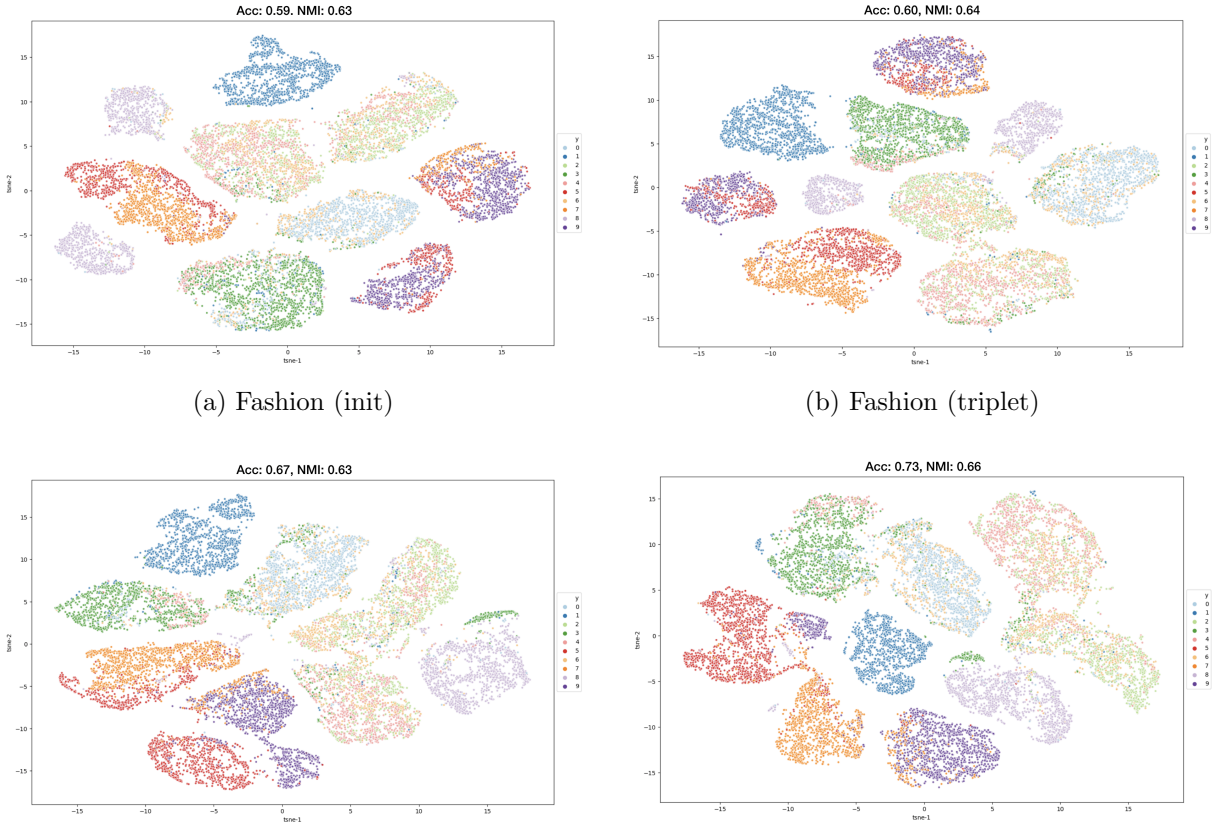
(b) Fashion(Triplet)

Figure 2.8: Evaluation of the effectiveness of triplet constraints in terms of Acc/NMI.

2.6.4.3 Experiments on triplet constraints

We experimented on MNIST and FASHION datasets. Figure 2.4 visualizes example triplet constraints (based on embedding similarity), note the positive instances are closer to anchors than negative instances. In Figure 2.8, we show the clustering Acc/NMI improves consistently as the number of constraints increasing. Comparing with Figure 2.5, we can find the pairwise constraints can bring slightly better improvements. That is because our triplet constraints are generated from a continuous domain and there is no

exact together/apart information encoded in the constraints. Triplet constraints can be seen as a weaker but more general type of constraint.



(a) Fashion (init)

(b) Fashion (triplet)

(c) Fashion (pairwise)

(d) Fashion (together)

Figure 2.9: Experiments with Ontologies. Evaluation of the clustering performance of four settings. (a) No constraints, (b) Triplet constraints from labels, (c) Pairwise constraints from labels and (d) triplet constraints and pairwise constraints generated from WordNet ontology.

2.6.4.4 Experiments on constraints generated from ontologies

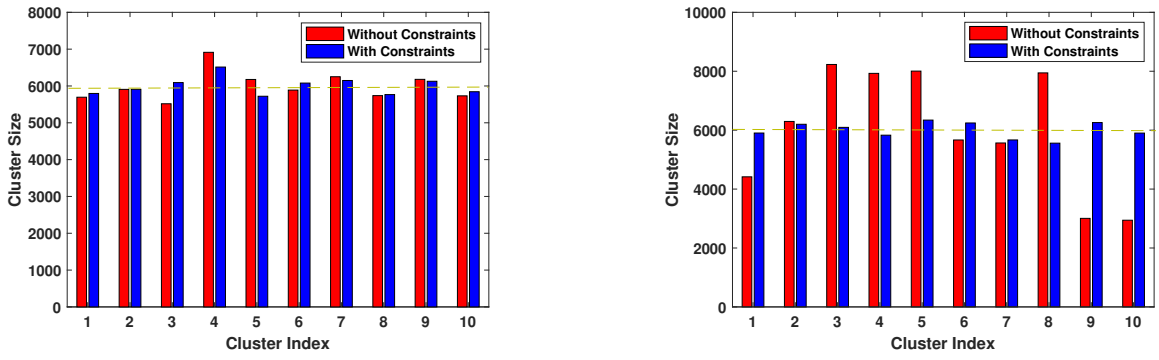
We experimented on the Fashion dataset. To show that pairwise constraints and triplet constraints generated from ontology can boost the performance with minimum supervision, we have randomly chosen 100 training instances as a limited labeled set and generate full pairwise constraints based on these labeled instances. To generate the triplet constraints, we follow the procedure described in section 2.5. Note the threshold for selecting positive pairs θ_p is set to be 0.5 to ensure positive pairs are close and non-trivial (positive points are not all from the same classes), the threshold for negative pairs θ_n is set to be

0.3 to be far away from anchors. We have generated 1000 triplet constraints randomly from the same 100 labeled training instances.

We empirically compare four different settings: i) clustering without any constraints, ii) clustering with just triplet constraints, iii) clustering with just pairwise constraints and iv) clustering with both pairwise and triplet constraints. Figure 2.9 shows the embeddings we learned with four different settings with the corresponding clustering performance. We can see from the plots that both triplet constraints and pairwise constraints can improve the clustering performance when learned individually. Moreover, pairwise constraints can bring more significant improvement. The right bottom plot shows that learning with both these two types of constraints together can improve the clustering accuracy and clustering NMI in a large margin and achieve the highest performance. This shows that the triplet constraints generated from WordNet ontology can help regularize the latent space learned with pairwise constraints and yield a latent space which more similar to the ground truth.

2.6.4.5 Experiments on global size constraints

To test the effectiveness of our proposed global size constraints, we have experimented on MNIST and Fashion training set since they both have balanced cluster sizes (see Figure 2.10). Note that the ideal size for each cluster is 6000 (each data set has 10 classes); we can see that blue bars are more evenly distributed and closer to the ideal size.



(a) MNIST (b) Fashion

Figure 2.10: Evaluation of the global size constraints. This plot shows each cluster's size before/after adding global size constraints.

We also evaluate the clustering performance on MNIST (Acc:0.91, NMI:0.86) and

Fashion (Acc:0.57, NMI:0.59). Comparing to the baselines in table 2.1, interestingly, we find the performance improved slightly on MNIST but dropped slightly on Fashion.

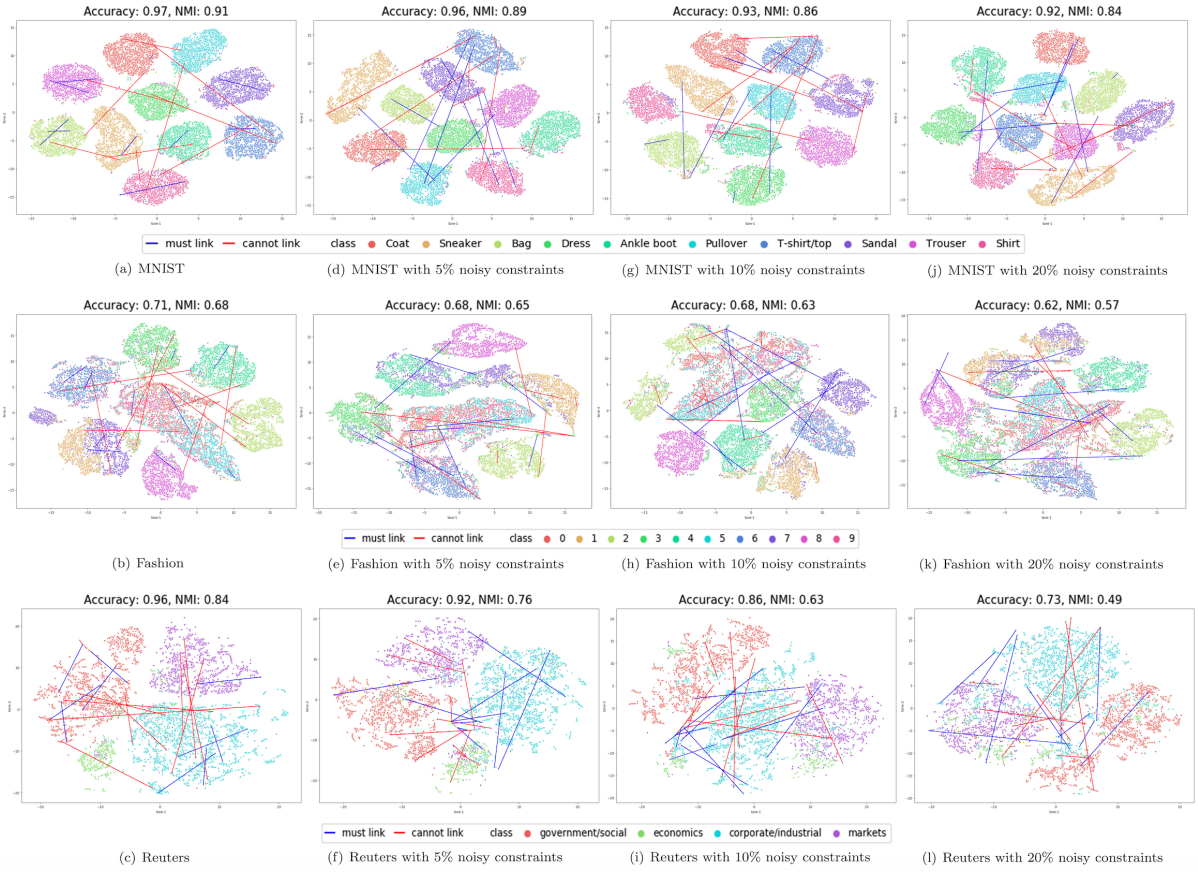


Figure 2.11: Effects of noisy constraints for MNIST, Fashion and Reuters Dataset.

2.6.4.6 Experiments on noisy constraints

Effect of Noisy Constraints. To understand the effect of noisy constraints on our model, we randomly generate 6000 pairs of constraints as described in Section 2.6.4.2. To generate noisy constraints, we first generate ground truth constraints and then flip the labels so that the true cannot-links become noisy must-links and the true must-links become noisy cannot-links. We define the degree of noisy constraints as the ratio of noisy constraints to ground truth constraints for each constraint type. For noisy degrees of 5%, 10%, 20%, we randomly generated 300, 600, 1200 pairs of noisy constraints by flipping the labels of ground truth constraints. We visualized the embedded representation of a random subset of instances and its corresponding pairwise constraints using t-SNE and

the learned embedding z. Figure 2.11 shows the cluster formation in training on MNIST, Fashion, Reuters dataset respectively.

We notice that the noisy constraint has negative effects on model performance. As the number of noisy constraints increases, the negative effect of noisy constraints on the model performance will also increase. For example, in Figure 2.11 the embedding without noisy constraints (plot (a)) has a better clustering result compared to the embedding with 20% noisy constraints (plot (j)). Moreover, we notice that most of the noisy must-links are not satisfied and most of the noisy cannot-links are satisfied. To satisfy noisy cannot-links, the model will move instances from the correct cluster to another cluster that tends to have similar instances, which explains the negative effect noisy cannot-links have on the model performance. Figure 2.11, plot (j) shows the model tries to satisfy some noisy cannot-links and forms a mixed cluster of instances "3", "5", and "8". In the MNIST dataset, the instances with the label "3", "5", and label "8" share some visual similarity. In plot (e, h, k), we observe a similar mixture of instances "Sneaker," "Sandal," "Ankle boot." In the Fashion dataset, these classes all represent shoes and share some similarities.

Noise Degree	0%	5%	10%	20%	Baseline
MNIST Acc	0.962 ± 0.01	0.953 ± 0.01	0.902 ± 0.05	$\mathbf{0.883 \pm 0.05}$	0.883 ± 0.01
MNIST NMI	0.910 ± 0.01	0.894 ± 0.02	0.828 ± 0.04	$\mathbf{0.809 \pm 0.04}$	0.861 ± 0.01
Fashion Acc	0.737 ± 0.04	0.709 ± 0.05	0.695 ± 0.04	$\mathbf{0.681 \pm 0.05}$	0.587 ± 0.01
Fashion NMI	0.694 ± 0.02	0.666 ± 0.03	0.650 ± 0.03	$\mathbf{0.629 \pm 0.03}$	0.632 ± 0.01
Reuters Acc	0.950 ± 0.01	0.856 ± 0.20	0.825 ± 0.10	$\mathbf{0.763 \pm 0.05}$	0.752 ± 0.01
Reuters NMI	0.818 ± 0.01	0.676 ± 0.01	0.578 ± 0.01	$\mathbf{0.503 \pm 0.04}$	0.542 ± 0.02

Table 2.3: Pairwise constrained clustering performance (mean ± std) averaged over 50 random noisy constraints sets. Baseline model is the model without using pairwise constraints.

Robustness Against Noisy Constraints. We define the noisy degree as the ratio of noisy constraints to ground truth constraints for one type of constraint. To test the model robustness against noisy constraints, we randomly generate 6000 pairs constraints.

For the noisy degree of 5%, 10%, 20%, we randomly generate pairs of noisy constraints by flipping the labels of ground truth constraints and test the model performance. In each run, we fix the random seed and the initial centroids for k-means based methods. For each method, we compare its performance to the unconstrained version. In Table 2.3, we show that on average, our model will start to perform worse than the unconstrained baseline model when the noisy degree in constraints reaches 20%.

	Raw & Rand	Raw & KMeans	AE & Rand	AE & KMeans
MNIST Acc	0.880 ± 0.07	0.915 ± 0.06	0.961 ± 0.02	$\mathbf{0.962 \pm 0.01}$
MNIST NMI	0.830 ± 0.06	0.859 ± 0.05	0.910 ± 0.02	$\mathbf{0.910 \pm 0.01}$
Epoch	350*	350*	124.38 ± 66.92	$\mathbf{107.60 \pm 35.62}$
Fashion Acc	0.762 ± 0.03	0.757 ± 0.03	0.721 ± 0.05	$\mathbf{0.737 \pm 0.04}$
Fashion NMI	0.697 ± 0.01	0.695 ± 0.02	0.680 ± 0.03	$\mathbf{0.694 \pm 0.02}$
Epoch	350*	350*	350*	350*
Reuters Acc	0.796 ± 0.06	0.797 ± 0.06	0.945 ± 0.01	$\mathbf{0.950 \pm 0.01}$
Reuters NMI	0.585 ± 0.08	0.588 ± 0.08	0.809 ± 0.02	$\mathbf{0.818 \pm 0.01}$
Epoch	47.73 ± 16.37	46.34 ± 12.36	9.33 ± 4.34	$\mathbf{6.08 \pm 0.79}$

Table 2.4: Pairwise constrained clustering performance (mean $\pm$ std) averaged over 50 random sets. Epoch 350*: model didn't converge after 350 epochs, where convergence is reached when the ratio of changed labels after an epoch < 0.001.

2.6.4.7 Ablation Study

Experiments on Initialization Approaches. To test the effect of different initialization approaches on our proposed deep clustering framework, we evaluate the model results for MNIST, Fashion, and Reuters dataset. Our model initializes both model weights and the cluster centers, so there are four initialization approaches. The "Raw & Rand" approach is to initialize both model weights and cluster centers randomly. "Raw & Kmeans" approach initializes cluster centers with KMeans and randomly initializes weights. The "AE & Rand" approach uses the pre-trained model to initialize weights and randomly initialize centroids. "AE & KMeans" uses Kmeans to initialize cluster centers and the

pre-trained model to initialize model weights.

In Table 2.4, we report the average performance with 6000 randomly generated pairwise constraints. For MNIST and Reuters datasets, we compare the result for "Raw & Rand" with "Raw & Kmeans" and "AE & Rand" with "AE & Kmeans." We find that the cluster center initialization with Kmeans can increase training speed. We also observe that the consistent increase in model performance and training speed by comparing "Raw & Kmeans", "Raw & KMeans," "AE & Rand," and "AE & KMeans." This shows that better weight initialization can help the model learn information from pairwise constraints. However, for the Fashion dataset, the model performance becomes worse when using a pre-trained model to initialize weights. The result agrees with our findings in Section 2.6.4.2. This shows that the autoencoder's features are not always ideal for DEC's clustering objective. To address this issue, we can perform end-to-end deep constrained clustering from raw features.

	600	1200	1800	2400	3000
MNIST Acc	0.33	0.40	0.45	0.47	0.49
MNIST Acc (with ℓ_C)	0.90	0.93	0.95	0.96	0.97
Fashion Acc	0.52	0.56	0.58	0.60	0.62
Fashion Acc (with ℓ_C)	0.59	0.61	0.62	0.63	0.64
Reuters Acc	0.79	0.81	0.83	0.85	0.86
Reuters Acc (with ℓ_C)	0.77	0.79	0.81	0.83	0.85

Table 2.5: Ablation study to evaluate the contribution of clustering loss to pairwise constrained clustering. Note we report the mean clustering accuracy for each data set under two settings which ℓ_C means adding the clustering loss function.

Evaluating the Contribution of Clustering Loss. To measure the contribution of clustering loss ℓ_C to our framework, we choose to study its influence on pairwise constrained clustering. We experiment on MNIST, Fashion, and Reuters data sets and report the average performance with a different number of randomly generated pairwise

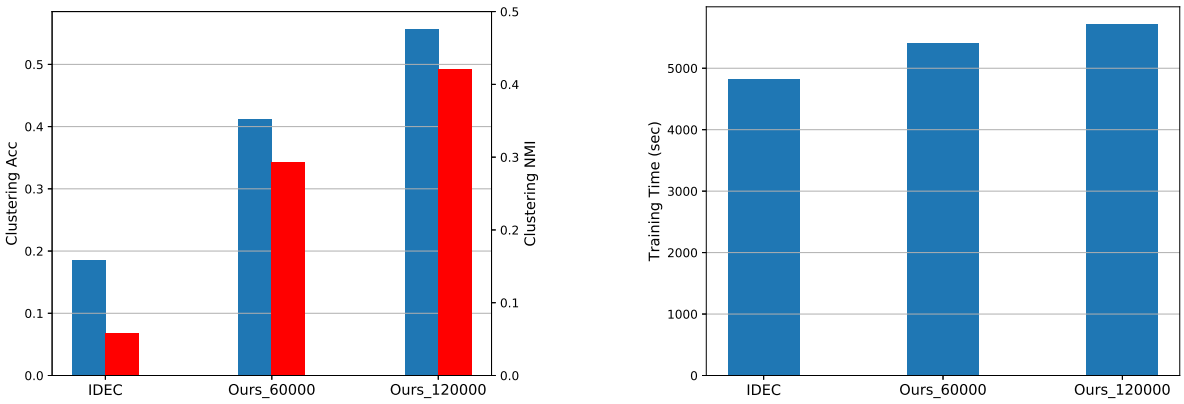
constraints. As shown in Table 2.5, the clustering loss is essential for image data sets, especially for the MNIST data set. The poor performance in MNIST has demonstrated the need to combine clustering loss with constraints learning. Otherwise, the network will overfit for a limited number of constraints. Interestingly the results from the Reuters data set are opposite that adding clustering loss may harm the performance marginally. We hypothesis that the uni-model assumption, which encoded in the clustering loss function is preferred for image data rather than in text data. Another finding from the experimental results is that the performance gap between adding clustering loss or not is shrinking as the number of constraints increases. This is expected because as the number of constraints increasing the contribution of constraints loss is more and more critical.

	IDEC	Pairwise	Instance	Global	Triplet
MNIST	178	197	62	135	230
Fashion	186	246	94	217	310
Reuters	5.15	8.28	3.82	−−	−−

Table 2.6: Runtime analysis for our proposed approach with diffrent types of constraints. We use the same experimental setting for each type of constraints and average the running time (sec) over 10 trails.

2.6.4.8 Experiments on Very Large Data Sets

Our previous experiments were on large data sets but under 100000 instances, here we discuss these results and explore our method on a challenging real-world data set over 600000 instances. A key result of our results shown in Table 2.6 is that our method's increase in run-time over DEC is minimal for the three data sets previously studied. Interesting, our framework with instance-difficulty constraints is actually faster than the IDEC baseline, which speeds up the deep clustering procedure. We believe this is because this extra side information is compatible with the geometry of the data and hence increases converge to the minima. For the remaining three types of constraints, the running time is close to the IDEC's results.



(a) Clustering Performance on SVHN

(b) Running Time on SVHN

Figure 2.12: We report the out-of-sample prediction results of SVHN data in the left figure and the running time analysis in the right figure. Note we report the average clustering performance and running time (sec) over 10 trails.

We now study our method's run time on a very large data set, SVHN [103], which contains 604388 training instances and 26032 test instances. Compared to our previously used data (MNIST), this data set incorporates an order of magnitude more labeled data and comes from a significantly harder, unsolved, real-world problem (recognizing digits and numbers in natural scene images). We use the same experimental setting as our pairwise constrained clustering except that we generate more pairwise constraints (60000 and 120000). We report the clustering performance as well as the time cost in Figure 2.12. The clustering performance result is consistent with our earlier results and improves upon the accuracy of IDEC. Importantly, despite there being hundreds of thousands of constraints, the run time is only slightly more than the baseline IDEC algorithm. These running time results show that our approach is efficient and will not add too much overhead to deep clustering approaches.

2.7 Conclusion, Limitations and Future Work

The area of constrained partitional clustering has a long history and is widely used. Constrained partitional clustering typically is mostly limited to simple pairwise together and apart constraints. In this paper, we show that deep clustering can be extended to a variety of fundamentally different constraint types, including instance-level (specifying hardness),

cluster level (specifying cluster sizes), and triplet-level. We also show that our framework can not only handle standard constraints generated from labeled side information but new constraints generated from an ontology graph. Furthermore, we propose an efficient training paradigm that applies to multiple types of constraints simultaneously.

Our deep learning formulation was shown to advance the general field of constrained clustering in several ways. Firstly, it achieves better experimental performance than well-known k-means, mixture-model, and spectral constrained clustering in both an academic setting and a practical setting (see Table 2.2). Importantly, our approach does not suffer from the negative effects of constraints [43] as it learns a representation that simultaneously satisfies the constraints and finds a good clustering. This result is quite useful as a practitioner typically has just one constraint set, and our method is far more likely to perform better than using no constraints. Moreover, we have visualized our learning process to show how the learned latent representation overcomes inconsistencies and incoherence within the constraints.

Most constrained clustering approaches assume the oracle is perfect, and all the constraints are noise-free. Here we have also studied our model's robustness against noise in constraints, particularly the popular pairwise constraints. The experimental results demonstrate that our model is quite robust (see section 2.6.4.6). We were able to show that our method achieves all of the above but still retains the benefits of deep learning, such as scalability, out-of-sample predictions, and end-to-end learning. We found that even though standard non-deep learning methods were given the same representations (the auto-encoder embedding) of the data used to initialize our methods, the deep constrained clustering was able to adapt these representations even further.

Our current work limitations are two-folded: limitations inherent with the deep clustering backbone we have used (DEC/IDEC) and limitations with how we add constraints. In the first limitation, DEC or IDEC is doing k-means style clustering with limitations such as being partitional (i..e, no hierarchy and partial assignments). Moreover, the cluster number k must be given apriori. As deep clustering evolves to more advanced styles of clustering, using the constraints we have explored in this paper seems reasonable. But

the challenge of also having the deep learning solve for k seems quite challenging given the need for a fixed architecture. As for the second limitation (how we add constraints), the main limitation is that we cannot solve for all constraint types at once without the need for multiple hyper-parameter tuning. This is part of a larger-scale problem in ML, how to tune hyperparameters (including k) efficiently. We leave the current limitations as interesting future works.

Acknowledgement

We acknowledge support for this work from a Google Gift entitled: "Combining Symbolic Reasoning and Deep Learning".

Chapter 3

A Self-Supervised Deep Learning Framework for Unsupervised Few-Shot Learning and Clustering

3.1 Introduction

Supervised visual representation learning has achieved great success in recent years. However, learning a good representation still requires much-labeled data under a supervised learning setting. Learning a useful visual representations without human supervision is a fundamental, understudied and long-standing problem [32]. Two popular learning paradigms for unsupervised representation learning are deep clustering and self-supervised learning.

Some works have been proposed for deep clustering [144, 64, 149]. This work is limited in that it adopts a multi-stage pipeline that pre-trains the auto-encoders with reconstruction loss and then applies a clustering module on it. Though these approaches yield better clustering results than traditional clustering methods on image data (as they learn a new representation for clustering), they are are not as useful for unseen (novel) classes/clusters. An alternative representative learning paradigm is self-supervised learning methods. Self-supervised learning approaches learn representations using objective functions similar to those used for supervised learning, but train networks to perform tasks where both the inputs and labels are derived from an unlabeled dataset. Many

recent works [49, 161, 104, 59] have been proposed, for example, [59] performs data augmentations such as rotating the images with different degrees and then to train the neural networks to predict the corresponding degrees. Although these learned models may not fully match the performance of supervised-learned representations, they have proved to be better than purely unsupervised representation learning approaches. However, these approaches perform self-supervised learning via heuristics with varying assumptions that if not true may harm the generalization ability of the representation.

In this work, we address learning a representation without supervision and demonstrate its use on two popular learning tasks. Our approach (visualized in Figure 3.1) to representation learning takes several intuitive steps as follows.

Category Discovery. Here we perform unsupervised representation along with clustering to take the unlabeled instances and place them into different categories/groups $(C_1 \dots C_k)$.

Post-Processing for Representative Data. Next, using an integer linear programming (ILP) formulation, we post-process the clustering partitions to get representative instances and balance discovered clusters in the previous step.

Construct Tasks for Category Differentiation. Now we use previously selected representative instances and then create augmented versions as our training set. Inspired by the recent success of meta-learning (which aims to learn from limited data to generate to unseen classes), we create few-shot learning tasks which *minimize* the intra-category variation whilst *maximizing* the inter-category embedding distance for better category separation.

Iterative Training. We iterative the above three steps to discover categories based on learned embedding and refining the embedding.

We demonstrate our representation learning scheme on two classic minimal supervision problems: clustering and few-shot classification. The few-shot classification here is a paradigm where the model has been learned for the *base* classes and then is transferred to learn to predict *novel* classes of which there are only a few examples available. We argue that a good visual representation should not only naturally separate images that

belong to different semantic groups within the training set but also be applicable for unseen classes. We evaluate our learned embedding functions in two settings: we first study the unsupervised few-shot classification problem with benchmark data sets such as Omniglot and *mini*ImageNet; we then test our learned embedding function on clustering tasks to show whether our pipeline improves upon different initial unsupervised representations. Based on experimental results, we show our approach achieves state-of-the-art performance comparing to recent unsupervised few-shot classification baselines. Moreover, the clustering results demonstrate that our method consistently improves the embedding learned by unsupervised representation learning for clustering.

We summarize the contributions of our work as follows:

- We propose a framework to improve the unsupervised representation learning (see section 3.3).
- We validate our proposed model in unsupervised-few-shot learning settings and show our proposed model achieves state-of-the-art results for benchmark datasets. Experimental results on clustering analysis also show that our work can further improve the embedding generated via popular deep clustering baselines (see section 3.4).
- Our ablation study shows the contributions of different technical components. We find out both the integer linear programming based post-processing algorithm and iterative training strategy improves the quality of our learned representations.

We begin this paper by next briefly overviewing related research, after which we discuss our approach in section 3.3. We experimentally compare our methods to many different baselines in section 3.4 and finally conclude.

3.2 Related Work

Image Representation Learning. Deep unsupervised feature learning has been explored to learn informative representations of images. One line of this direction is learning various auto-encoders that learn a reduced feature representation that can reconstruct

the inputs. For example, the auto-encoder [21], denoising auto-encoder [133], variational auto-encoder [85], and adversarial auto-encoder [14]. Additionally, deep generative models also learn the underlying latent information about images and many generative adversarial networks [61, 33, 47] have been proposed to encode visual information from an adversarial learning strategy.

Recent deep clustering works take the advantages of these unsupervised visual representation techniques to learn clustering favored representations. For example, [144] (DEC) fine-tune the embedding learned from stacked-denoising auto-encoder via a self-supervised signal to form tight clusters. Unlike DEC, [57] guides agglomerative clustering and feature learning jointly based on the over-clustering initialized by KNN.[26] does discriminative clustering by alternating between clustering the features of a convolutional neural network and using the clusters as labels to optimize the network weights via back-propagating a standard classification loss. Our work is similar in style to previous deep clustering works, which learn a representation that is general for clustering is useful. Differently, our framework further improves the representation learned by these works via clustering and self-supervised few-shot learning.

Unsupervised Meta-Learning. Recent work studies unsupervised meta-learning, which focuses on how to generate tasks for meta-learning approaches and pre-train the meta learner to achieve comparable performance with supervised meta-learning. For example, [72] constructed tasks from unlabeled data via clustering from different views and runs meta-learning models over the constructed tasks. This paper's core idea is to leverage the unsupervised embeddings to propose tasks for a meta-learning algorithm to pre-train the model for potential supervised few-shot classification tasks. [83] allows unsupervised, model-agnostic meta-learning for few-shot classification problems by generating synthetic data with artificial labels. [7] uses a similar way to generate tasks with synthetic data and pseudo labels but train on a more advanced learner as [6]. Our work differs from these works as our model gets trained over unsupervised few-shot classification tasks as an intermediate learning step. Moreover, our work aims to learn a good representation that works for clustering and few-shot learning by iteratively fine-tune the embedding

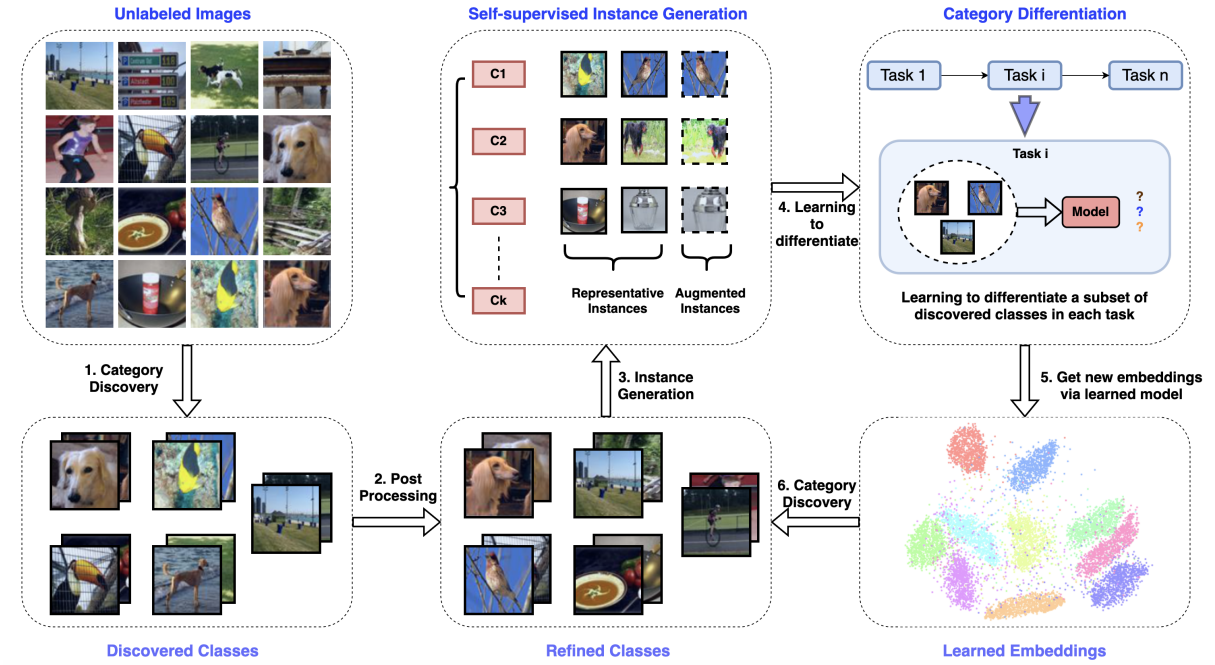


Figure 3.1: Pipeline of the proposed unsupervised representation learning framework. The inputs are unlabeled images and the output is the learned embedding function. The learning process mainly contains fours steps: 1) category discovery, 2) category post-processing, 3) generate virtual instances to construct learning tasks, 4) learning to differentiate created categories.

function learned with carefully designed few-shot learning tasks.

3.3 Methodology

In this section, we present details of the proposed representation learning scheme in which the model discovers different concepts first and then gradually learns to discriminate both the representative instances and augmented instances within different concepts. We introduce a new method to learn an unsupervised embedding function that is useful for both unsupervised few-shot classification tasks and image clustering.

3.3.1 Overview

The pipeline of our proposed framework is illustrated in Figure 3.1. Firstly, we leverage the existing image representation methods and embed all training data into one latent space and conduct clustering. Secondly, given the clustered instances, we propose an integer linear programming based formulation to identify some unlabeled instances for

representative and balanced clusters. Thirdly, we augment the selected representative instances of each discovered category to enrich the training data. Finally, we train our embedding function with Prototype-based networks using self-generated few-shot learning tasks. We repeat the previous learning steps until we learn a stable representation.

3.3.2 Step 1: Category Discovery

Clustering is a natural way to gather data that has similar features or close semantic meanings. Recent works on deep clustering that simultaneously conduct representation learning and clustering have been shown to outperform traditional clustering approaches. We take advantage of new unsupervised representation learning methods and choose one of them as our initialization function $\phi(x)$. Given all the unlabeled points $\{x_1, \dots x_n\}$ we can calculate their embeddings as $\{\phi(x_1), \dots \phi(x_n)\}$.

We assume that each discovered category/concept should have one centroid, and all the instances belonging to this category/concept should be closely clustered around the centroid. Thus we use the K-means objective below to look for concepts:

$$\arg\min_{\mathcal{C}} \sum_{i=1}^{n} [\arg\min_{z_i} ||\mathcal{C}z_i - \phi(x_i)||] \tag{3.1}$$

$\mathcal{C}$ is a $d \times k$ matrix where each column corresponds to a centroid, k is the number of centroids, and z_i is a binary assignment vector with length d. By solving this objective function, we will get k clusters of unlabeled instances, which will be used as discovered categories for the next learning stage.

3.3.3 Step 2: Post-Processing for Representative Data

We wish to use the discovered categories to generate pseudo-labels and create pre-tasks for our next step's self-supervised learning. However, directly using these partitions to generate supervised learning tasks will incur several challenges. For example, 1) the size of each cluster may vary greatly so that large clusters can dominate; 2) some unlabeled instances are hard assigned to a category so that the instances lying on the cluster decision boundaries are prone to be falsely clustered. To counter the previous issues, we propose selecting a group of instances that best resembles the centroids of each discovered category and form balanced clusters.

Given the n unlabeled instances we aim to find a $n \times 1$ binary allocation vector $T = \{t_1, \dots t_n\}$ that selects valuable unlabeled instances. $t_i = 1$ means instance i is selected and $t_i = 0$ means we ignore instance i. Given the learned latent embeddings $\{\phi(x_1), \dots \phi(x_n)\}$ and the centroids for k clusters as $\mathcal{C} = \{c_1, \dots c_k\}$, we calculate the cluster probability distribution of instance x_i belonging to cluster j as w_{ij}:

$$w_{ij} = \frac{\exp\{-||\phi(x_i) - c_j||^2\}}{\sum_p \exp\{-||\phi(x_i) - c_p||^2\}} \tag{3.2}$$

We look for instances which lie close to their category centroids, to measure the closeness we calculate the entropy of each instance as $Q_i = H(w_i) = -\sum_{j=1}^{k} w_{ij} \log w_{ij}$ and use $n \times 1$ vector Q to represent them. Now we solve for the $n \times 1$ binary vector T. One objective function then is simply find the most valuable instances:

$$\arg\min_t \sum_i t_i Q_i \tag{3.3}$$

The aim of the constraints are two-folded: to balance the size of each discovered category whilst also keep the most valuable instances. Our first basic constraint includes the total number of selected instances to be $m < n$ so that uncertain unlabeled instances will be removed. Our next constraint requires that each cluster should have a reasonable number of instances so that some large clusters won't dominate. Thus we have

$$s.t. \sum_i t_i = m \tag{3.4}$$

$$s.t. \sum_i t_i z_{ij} \geq L \quad \forall j \tag{3.5}$$

$$s.t. \sum_i t_i z_{ij} \leq U \quad \forall j \tag{3.6}$$

Note that U and L serves as the upper bound and lower bound of each cluster's size. Details of how we set up the constraint hyper-parameters will be described in experimental section 3.4.

3.3.4 Virtual Instance Generation

Given the m selected instances which belong to k categories, the instances belonging to the same category tend to be similar to each other and are not diverse enough. Training a supervised few-shot learning model over these instances is prone to overfitting due to the low-diversity and cannot generalize well. Moreover, the learned embedding will largely resemble the embedding which we initialized.

Increasing the number of training examples by data augmentation is a popular approach in supervised learning and semi-supervised learning to improve generalization. Before splitting the selected instances into support sets and query sets for meta training, we apply traditional image augmentation approaches to augment the m chosen instances. We pick the standard image augmentation strategies to enrich the diversity while maintaining the class membership of the augmented instances. To be specific, we compose random sized crop, image jitter, and random horizontal flip as our augmentation strategy.

3.3.5 Iterative Training

In this section, we introduce how we train our representation learning framework thoroughly. Our work aims to learn an embedding function f_θ that naturally splits the discovered categories of all the unlabeled instances and form tight and diverse clusters. To achieve this goal, we propose to use a prototype-based network as f_θ and train it via few-shot classification tasks based on discovered labeled instances.

We first introduce notations and recap the few-shot learning. In few-shot learning tasks, we divide the dataset into three disjoint sets which are meta-training set $\mathcal{S}^{tr}$, meta-validation set $\mathcal{S}^{val}$ and meta-testing set $\mathcal{S}^{test}$. $\mathcal{S}^{tr}$ contains all the base classes, $\mathcal{S}^{val}$ and $\mathcal{S}^{test}$ contains disjoint novel classes. The few-shot learning models are trained in an episodic paradigm [134] that each episode contains one support set $\mathcal{D}^S$ and one query set $\mathcal{D}^Q$. Here, the support set $\mathcal{D}^S$ serves as the labeled training set on which the model is trained to minimize the loss of its predictions for query set $\mathcal{D}^Q$.

Secondly, we introduce how to train our few-shot classification learner. Denote the augmented representative training set as $\mathcal{S}^{tr}$, now we randomly split $\mathcal{S}^{tr}$ into support set $\mathcal{D}^S$ and query set $\mathcal{D}^Q$. We choose Prototypical Networks as our learner to classify differ-

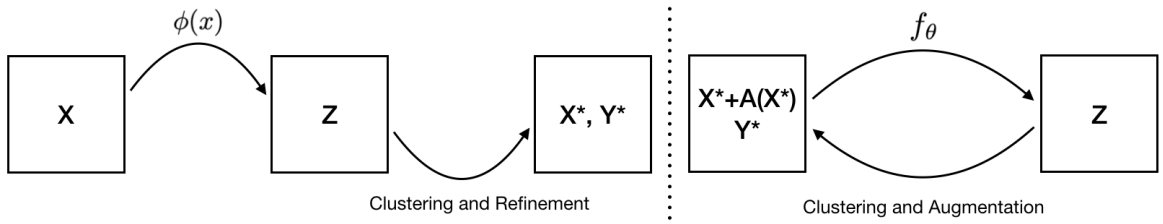


Figure 3.2: The whole learning process of our framework (left hand side shows model initialization and right hand side shows iterative learning). Left: The unlabeled points set X is first encoded to Z via existing unsupervised representation learning function $\phi(x)$. After that we conduct clustering on Z and post-process the clustering assignments to get pseudo-labeled data set X^*, Y^*. Right: Given the pseudo-labeled data and corresponding composed augmentations we train ProtoNet f_θ via episodic learning and get learned embeddings Z, clustering on Z and re-label X^* and generate new augmented instances.

ent discovered classes. Prototypical Networks compute an M dimensional representation $c_k \in \mathcal{R}^M$, or prototype, of each class through an embedding function f_θ with learnable parameters θ. Each prototype c_k is the mean vector of the embedded support points belonging to class k:

$$c_k = \frac{1}{\mathcal{D}_k^S} \sum_{(x_i, y_i) \in \mathcal{D}_k^S} f_\theta(x_i) \tag{3.7}$$

Given a Euclidean distance function d, Prototypical Networks produce a distribution over classes for a query point x based on a softmax over Euclidean distances to the prototypes in the embedding space:

$$p_\theta(y = k|x) = \frac{\exp\left(-d(f_\theta(x), c_k)\right)}{\sum_{k'} \exp\left(-d(f_\theta(x), c_{k'})\right)} \tag{3.8}$$

Learning proceeds by minimizing the negative log probability of the true class k via SGD: $J(\theta) = -\log p_\theta(y = k|x)$. Training episodes are generated from meta-training set $\mathcal{S}^{tr}$.

Now we connect all the technical components and visualize the training process in Figure 3.2. Note the embedding function f_θ is initialized in each iteration to learn a new embedding of unlabeled instances that can discriminate against the previously discovered classes and augmented instances. We repeat the whole training process for f_θ until the clustering results remain stable. We empirically observe that the clustering results of the learned embeddings Z tend to converge after a few rounds of iterative learning loop.

Moreover, the iterative learning consistently improves both the clustering results and few-shot classification performance which can be seen in the experimental section 3.4.

3.4 Experiments

The proposed framework is evaluated in two learning settings, which are image clustering and unsupervised few-shot classification. Our experimental results aim to answer the following questions:

- How does the proposed framework perform on unsupervised few-shot classification tasks compared to recent baselines?
- Compared to the original features we used to discover the categories, can our proposed framework provide a further enhancement in terms of clustering results?
- How does each step within the framework contributes to the final results?

3.4.1 Datasets

Omniglot [112] is a dataset of handwritten characters frequently used to compare few-shot learning algorithms. It comprises 1623 characters from 50 different alphabets. Every character in Omniglot has 20 different instances. The *mini*ImageNetis a collection of ImageNet for few-shot image recognition. It is composed of 100 classes randomly selected from ImageNet, with each class containing 600 examples.

We also experiment on some traditional image clustering data sets to study whether our framework can improve further on the image clustering tasks: 1) MNIST [90], which consists of 70000 handwritten digits of 28-by-28 pixel size. The digits are centered and size-normalized in our experiments; 2) FASHION-MNIST [143]: a Zalando's article images-consisting of a training set of 60000 examples and a test set of 10000 examples. Each example is a 28-by-28 grayscale image, associated with a label from 10 classes; 3) USPS, which contains 9298 handwritten digit images with the size of 16 by 16 pixels.

3.4.2 Implementation Details

We have explored different unsupervised representation learning algorithms to initialize our pipeline. For the Omniglot data set, we choose adversarially constrained autoencoder

interpolation (ACAI) [14], which is a convolutional autoencoder regularized with a term encouraging meaningful interpolations in the latent space. For *mini*ImageNet, we leverage the recent large-scale unsupervised visual representation work (DeepCluster [26]).

For traditional deep clustering data sets such as MNIST, Fashion-MNIST, and USPS, IDEC [124] is an auto-encoder based deep clustering algorithm that we adopted for testing whether our proposed pipeline improves upon traditional deep clustering approaches.

For k-means clustering, we initialize the clustering methods with 100 random trials and use the best trial for initialization; the number of clusters for unsupervised few-shot learning is set to 100 empirically for both Omniglot and *mini*ImageNet.

To select the valuable instances from the clustering results, we set m to be 80% of the original unlabeled data size. The rationale for selecting a relatively large portion is because we only observe a small group of points lies between different clusters across different data sets. The size of selected instances is not sensitive within the range between 70% to 90% of all unlabeled data. Denote the average number of instances per cluster as $\mathcal{S}$; we set the upper bound U and lower bound L as $2\mathcal{S}$ and $0.5\mathcal{S}$. We solve this integer programming problem by linear programming relaxation and then round the results for faster computation on complex datasets such as *mini*ImageNet and Omniglot.

For fair comparison with other few-shot classification methods, we adopt a widely-used CNN architecture [134, 54, 123, 126] as the feature embedding function f_θ. Following [123], we adopt the episodic training procedure, i.e., we sample a set of N-way k-shot training tasks to mimic the N-way k-shot test problems. In all settings, the query set's size of each novel class is set as 15, and the performance is averaged over 1000 randomly generated episodes from the test set. All our models are trained with Adam optimizer and an initial learning rate of 10^{-3}. We trained 240000 tasks for *mini*ImageNet and 100000 tasks for other data sets. We have included [72, 83, 7] as our unsupervised few-shot learning baselines and also [64] as deep clustering baseline.

3.4.3 Unsupervised Few-shot Classification on Omniglot

We compare our model's unsupervised few-shot classification mean accuracy with recent baselines on the Omniglot dataset in Table 3.1. We can find our approach significantly

	Omniglot			
Model	(5, 1)	(5, 5)	(20, 1)	(20, 5)
CACTUs-MAML	68.84%	87.78%	48.09%	73.36%
CACTUs-ProtoNet	68.12%	83.58%	47.75%	66.27%
UMTRA	77.80%	92.74%	62.20%	77.50%
AAL-MAML++	88.40%	97.96%	70.21%	88.32%
AAL-ProtoNet	84.66%	89.14%	68.79%	74.28%
Ours	**94.09%**	**98.43%**	**87.56%**	**96.15%**
Supervised-MAML	98.70%	98.90%	95.80%	98.90%
Supervised-ProtoNet	98.80%	99.70%	96.00%	98.90%

Table 3.1: Comparison to prior works on *Omniglot*. We report the few-shot classification mean accuracies

improves upon the previous baselines in both 5 way few-shot classification and 20 way few-shot classification settings. This shows that our framework consistently improves the learned embeddings under multiple way classification tasks. We also report the representative supervised few-shot classification results and find out our approach largely minimizes the gap between unsupervised and supervised few-shot classification in Omniglot, with 1.27% difference in 5-way 5-shot learning and 1.75% difference in 20-way 5-shot setting.

3.4.4 Unsupervised Few-shot Classification on *mini*ImageNet

We further evaluate the unsupervised few-shot classification on *mini*ImageNet data set which is much more challenging than Omniglot. Results are reported in Table 3.2. As shown, our approach still achieves the best results for 5 way classification with a different number of shots. This suggests that our framework is general across different data sets and behaves consistently under different few-shot classification settings. Moreover, comparing our results with the supervised baselines, we can find the gap is larger compared to the gap in Omniglot dataset. This is acceptable as images within *mini*ImageNet vary and are harder to recognize.

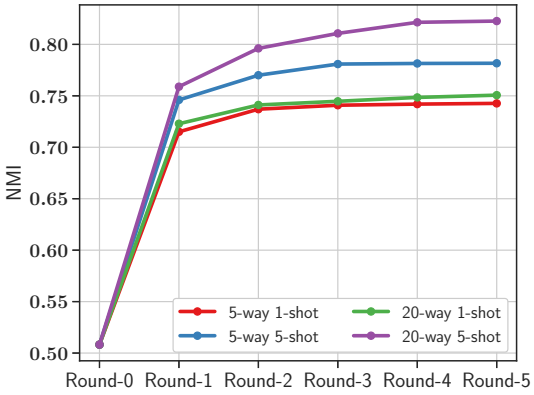
	*mini*ImageNet			
Model	(5, 1)	(5, 5)	(5, 20)	(5, 50)
CACTUs-MAML	39.90%	53.97%	63.84%	69.64%
CACTUs-ProtoNet	39.18%	53.36%	61.54%	63.55%
UMTRA	39.93%	50.73%	61.11%	63.55%
AAL-MAML++	33.30%	49.18%	–	–
AAL-ProtoNet	37.67%	40.29%	–	–
Ours	**41.16%**	**57.03%**	**68.85%**	**70.64%**
Supervised-MAML	46.81%	64.13%	71.03%	75.54%
Supervised-ProtoNet	50.16%	65.56%	70.05%	72.04%

Table 3.2: Comparison to prior works on *mini*ImageNet. We report the few-shot classification mean accuracies. AAL [7] didn't report the results for 20 shot and 50 shot setting so we leave them as blank entries

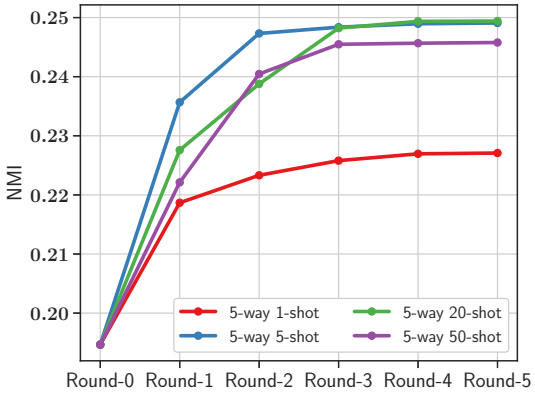
3.4.5 How our framework improves upon the initial embeddings in terms of clustering performance.

Our proposed framework takes advantage of the initialization embeddings learned from recent unsupervised learning methods. To show how our model improves upon them, we test the clustering performance of both initial embeddings and learned embeddings and plot the results in Figure 3.3. Note we have used ACAI's [14] embedding algorithm to initialize Omniglot data and DeepCluster's [26] embedding approach to initialize the *mini*ImageNet data. We adopt standard metrics for evaluating clustering performance which measures how close the clustering found is to the ground truth result. Specifically, we employ the normalized mutual information (NMI) [125, 148].

As can be seen from part (a) of Figure 3.3, we train our framework for five iterations with 5-way and 20-way few-shot classification tasks. The embeddings' clustering performance consistently improves as the number of rounds increasing and tend to converge after five iterations. The first iteration of learning provides the largest improvement, which shows that our framework further improved the initialized embedding. Moreover,



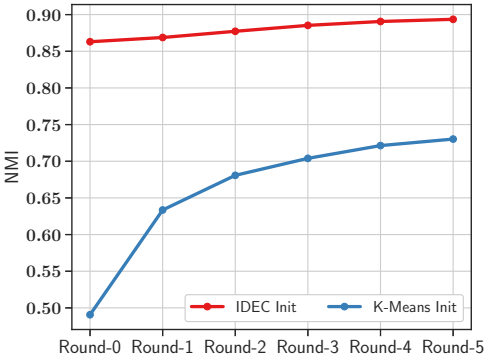
(a) Omniglot



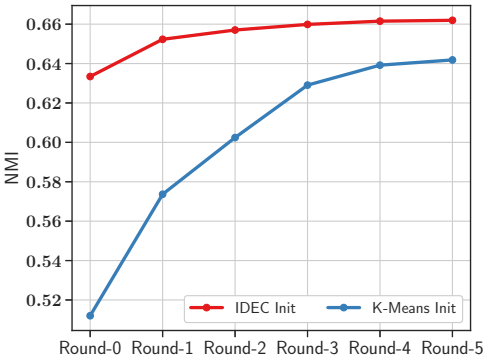
(b) *mini*ImageNet

Figure 3.3: We evaluate the clustering NMI for our learned embeddings on Omniglot and *mini*ImageNet for each training iteration.

we discover that the embeddings learned with five shots are better for clustering than the embeddings learned with only one shot. We have observed the similar results in *mini*ImageNet experiment that shown in part (b) of Figure 3.3. Our framework can provide consistent improvements across different data sets with different initializations.



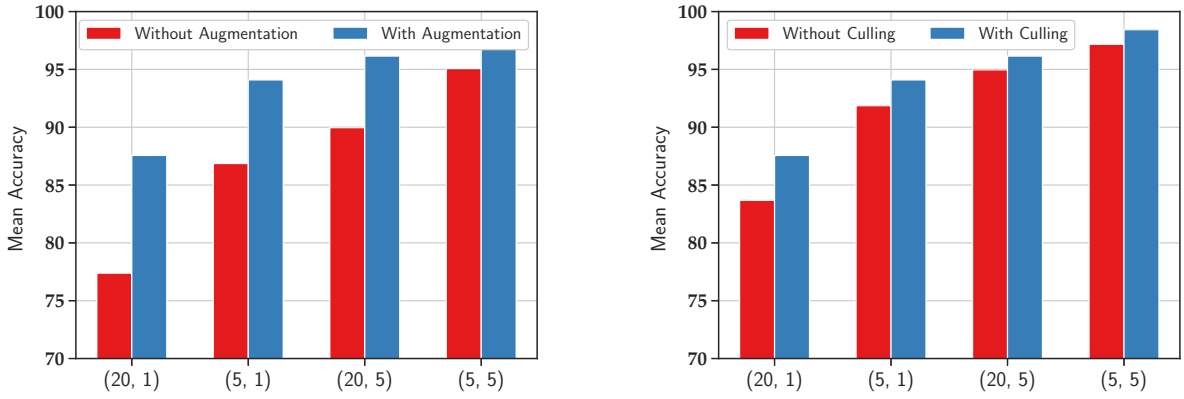
(a) Clustering results on MNIST



(b) Clustering results on Fashion

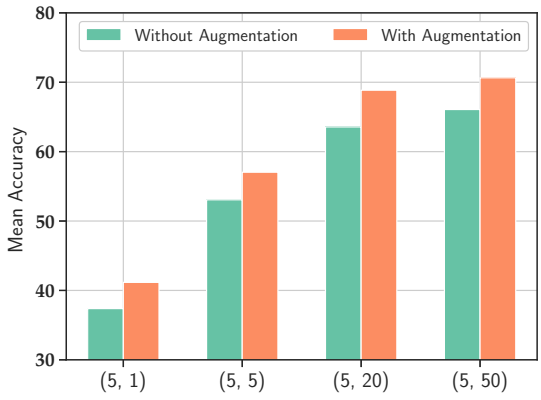
Figure 3.4: We evaluate the clustering NMI for our learned embeddings on MNIST, FASHION-MNIST for each training iteration.

Moreover, we have tested our approach on traditional clustering datasets. We apply two approaches to initialize all these three datasets. We first test on whether our framework can improve upon the existing deep clustering approach (IDEC). We also use

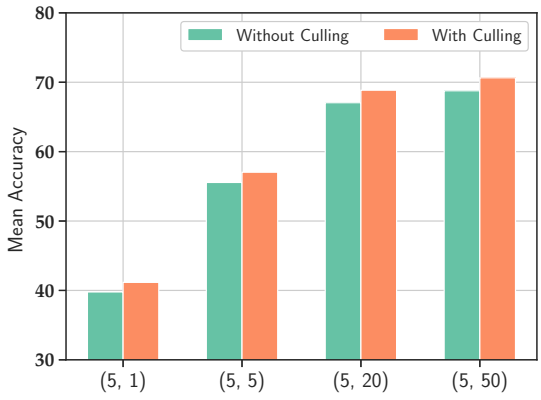


(a) Omniglot

(b) Omniglot



(c) *mini*ImageNet



(d) *mini*ImageNet

Figure 3.5: Plots of ablation study. Note plot (a) and (c) test on how much our framework benefit from composition of data augmentations and plot (b) and (d) test on how much our framework benefit from culling the unlabeled instances.

a naive K-Means clustering on the raw images to test whether our framework is sensitive to the initial partition results. We fix the few-shot classification tasks as 5-way 5 shot classification problem for our training process. The clustering results for these three data sets are shown in Figure 3.4.

Firstly, we can find our framework always improves the clustering performance no matter which initialization strategy we use. Secondly, we observe that even if we start our framework with K-Means partitions over the raw images, our framework can still largely improve the clustering NMI for MNIST and Fashion-MNIST. Thirdly we find that our approach can achieve high NMI over USPS data set by using K-Means initialization.

We hypothesis this could happen when the initial NMI is relatively high. Overall speaking, our proposed approach is general for learning a representation; the initialization approach serves as a performance lower-bound for our framework.

3.4.6 Benefits of data augmentations

Figure 3.5 (a) and (c) shows the impact of adding composed data augmentations when models are trained for different few-shot classification settings. In the Omniglot data set, we find that when we are doing 1-shot learning, data augmentation contributes significantly to the final performance. With more labeled points, the gaps between using or not use augmentation decrease. However, we still observe a large improvement in terms of classification accuracy. In *mini*ImageNet, we find data augmentation consistently improves the final performance with a different number of shots. Overall speaking, the composition of data augmentations plays a critical role in unsupervised few-shot classification.

3.4.7 Benefits of category post-processing

Figure 3.5 (b) and (d) shows that the culling of unlabeled instances via our post-processing objective improves the performance on both data sets. This suggests that directly using the naive K-Means clustering results to design pre-task may not be appropriate, picking instances which are representative for each discovered class is beneficial for creating useful few-shot classification tasks.

3.5 Conclusions

We have presented a self-supervised learning framework to learn representation on images for better downstream tasks such as unsupervised few-shot learning and clustering, where no human annotation is provided. We first discover categories from initialized embeddings and then propose integer linear programming to select valuable instances that form balance and representative concepts. Moreover, we augment the selected valuable instances to create pre-tasks and iterative train our network to fine-tune gradually, which resembles the human learning process, such as discovering concepts and learning to separate them. We show by experiments that the proposed framework has further improved the embedding initialized by existing representation learning approaches in terms of down-

stream tasks. For unsupervised few-shot classification, our proposed approach achieved state-of-the-art performance on the Omniglot and *mini*ImageNet benchmarks. In the image clustering setting, we also find our approach further improves recent deep clustering approaches under traditional clustering data sets. The ablation study demonstrates the importance of each technical component in our framework.

Chapter 4

Deep Descriptive Clustering

4.1 Introduction

As machine learning is applied to more complex data and situations, the need to understand a model's decisions becomes more paramount. The area of explainable AI (XAI) [1] is motivated to enhance the interpretability of complex machine learning models, especially deep learning. Arguably XAI is more needed and more challenging in unsupervised learning such as clustering as the explanations are usually at the model level rather than the instance level. For example, supervised learning explanations mainly focus on why an instance is classified to a specific class [106]; however, with clustering we need to explain the semantics of each discovered cluster.

Existing work on explainable clustering (see Figure 4.1) typically takes one of two directions: i) explanation by design algorithms [15, 100] that use interpretable features to create both a clustering and an explanation (left branch of Figure 4.1). ii) explanation by post-processing [39, 111] which take an existing clustering and generate an explanation using another additional set of features (tags) not given to the clustering algorithm (right branch of Figure 4.1). Each direction has its limitations: the first type of work is not suitable for complex data such as images and graphs as the underlying features are uninterpretable to a human. Post-processing methods are algorithm-agnostic, but they did not fully use the information from additional features to guide the clustering process hence may generate poor explanations as the clustering may be difficult to explain. In-

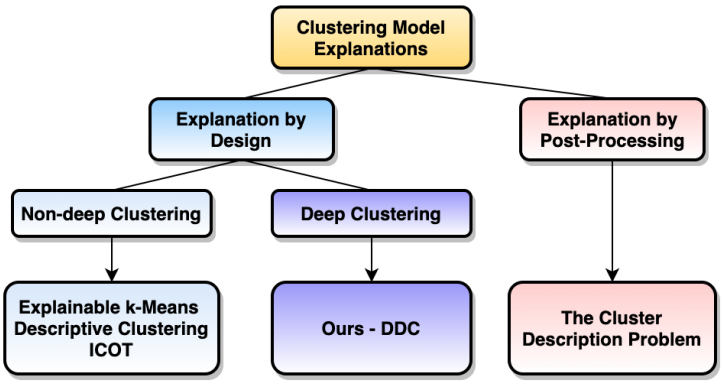


Figure 4.1: Taxonomy of works on clustering explanations.

stead, our proposed method will learn a good clustering that is also interpretable while post-processing approaches only find the best explanation possible for a given clustering.

Explaining deep clustering results using the underlying complex features is a challenging but alternative direction. Instead, we explore the situation that a partial set of instance-level semantic tags are known from which we generate a cluster-level description along with complex features to cluster on. This setting is not uncommon and was previously studied [37, 39]. Example settings include Twitter graphs where the user-user graph is clustered and explained using hashtags and personal images where the tags are descriptors of events in the image.

To address the challenges of simultaneously clustering and explaining efficiently with incomplete semantic features, we propose a novel deep descriptive clustering framework (DDC) that incorporates both deep clustering and cluster-level explanation. The whole framework is trained iteratively to maximize the consistency between the clustering and its explanation. To be specific, we formulate the cluster-level description problem as an Integer Linear Programming (ILP) which is solved for concise and orthogonal descriptions for each cluster. Inspired by the success of the discriminative clustering model [88] which has fewer assumptions of the data, our main clustering objective maximizes the mutual information between empirical distribution on the inputs and the induced clustering labels. Finally, we propose a pairwise loss with self-generated constraints to penalize the inconsistency between the clustering feature space and discovered descriptive tag space to improve the clustering quality. The major contributions of this paper are listed as follows:

- We propose a novel framework to learn clustering and cluster-level explanations simultaneously. The proposed architecture supports learning from the sub-symbolic level (which clustering is performed on) and symbolic level (which explanations are created from).
- We formulate the class-level explanation problem as an ILP to solve concise and orthogonal explanations. A pairwise loss function is proposed with self-generated constraints to bridge the clustering and explanation.
- Empirical results on public data demonstrate that our model consistently achieves better clustering results and high-quality explanations compared to recent baselines.
- We explore the novel direction of graphical ontology explanations for clustering when the number of clusters is large and a lengthy explanation list is problematic.

The rest of this paper is organized as follows. In section 4.2, we overview related works. We then introduce our learning objectives and optimization algorithms in section 4.3. Experimental results and analysis are reported in section 4.4. Finally, section 4.5 concludes this paper with future research directions.

4.2 Related Work

Explainable clustering models. Many previous works [93, 56, 58, 15] have explored the explainable-by-design algorithms which consider the simultaneous construction of decision trees and cluster discovery for explainable clustering. Typical work such as [100] considered traditional clustering algorithms like k-medians/means. However, one major limitation of these methods is that they require the features used for clustering to be interpretable which may not be the case for complex data such as graphs and images. Another line of research [39, 111] assumes one set of semantic tags for each instance are available to do post-processing explanation. [39] proposed a model-agnostic explanation method that explains any given clustering with tags but does not change the clustering. Perhaps the closest work to our own is [37] but is limited to simple diameter-based clustering objectives and scales to just a few thousand instances whilst making strong

assumptions such as having well-annotated tags for every instance. Our work differs from the above: we learn a *deep clustering* model and cluster explanation *simultaneously* with a *partial* set of semantic tags and scales for *large* data sets.

Multi-view clustering. As our approach uses semantic tags for explanation this can be seen as another view of the data; hence we overview the recent works on multi-view clustering [17, 146, 121, 128, 141, 75] and discuss how our proposed work differentiates from it. The goal of multi-view clustering is getting better clustering results via exploiting complementary and consensus information across multiple views rather than simply concatenating different views. Our descriptive clustering setting is different from multi-view clustering: Firstly, instead of just one goal which maximizes clustering performance, our work has another explanation objective to find meaningful descriptions of clusters. Secondly, most multi-view clustering is for similar views (i.e., all images) whereas our views are more diverse (e.g., continuous image features with categorical semantic tags) than general multi-view clustering settings.

Constrained clustering. Unlike most multi-view clustering algorithms which leverages the knowledge from different views to maximize the clustering performance, constrained clustering assumes the users have access to partial pre-existing knowledge about the desired partition of the data. The constraints are usually expressed via pairwise constraints [136, 19, 12] such as *together* and *apart* which indicates whether two instances belong to the same cluster or different clusters. Recent works [55, 155] have also extended constrained clustering to deep learning models. Our work shares one common attribute with these works in using a constrained optimization objective for better clustering. However, in this work our constraints are dynamically self-generated in that they cannot be known a priori as generating those constraints require both the feature representation and the clustering explanations.

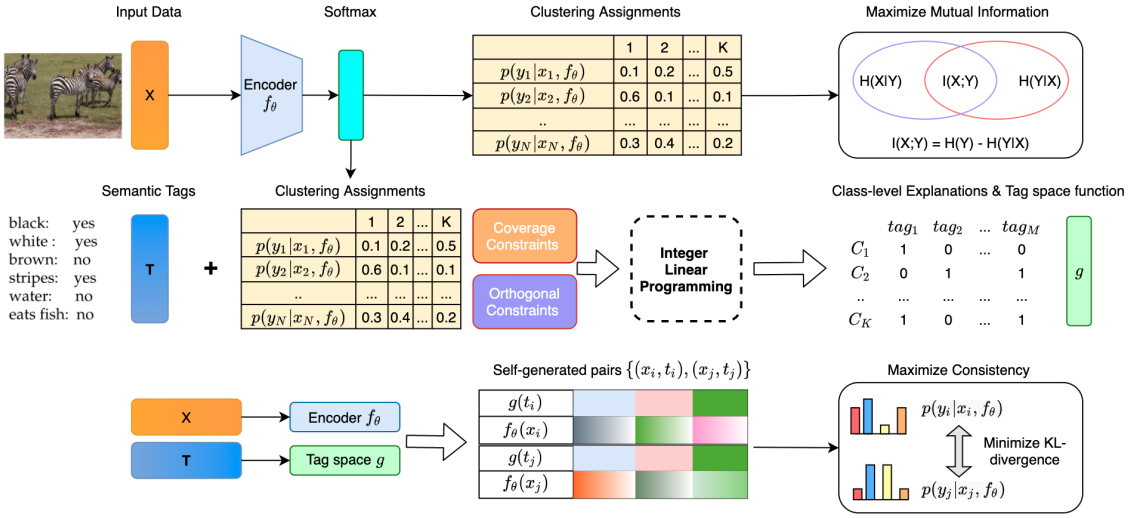


Figure 4.2: The framework of proposed deep descriptive clustering (DDC). DDC consists of one clustering objective, one sub-symbolic explanation objective, and one self-generated objective to maximize the consistency between clustering and explanation modules.

4.3 Approach

4.3.1 Overall Framework

The framework of our proposed Deep Descriptive Clustering (DDC) is shown in Figure 4.2. It can be divided into three learning objectives: i) *clustering objective* which maximizes the mutual information between the empirical distribution on the inputs and the induced label distribution; ii) *class-level explanation objective* which finds the shortest and different explanations for each cluster and creates a tag space mask function g to filter out uninformative tags; iii) *an instance pairwise loss term* with self-generated constraints to maximize the consistency between the clustering feature space and the descriptive tag space induced by mask function g.

4.3.2 Information Maximization for Clustering

Given unlabeled dataset of N data points as $X = \{x_1, ..., x_N\}$ where $x_i = (x_{i1}, ..., x_{iD}) \in \mathbb{R}^D$ are D dimensional feature vectors, the goal of our proposed model is to predict the clustering assignments $y \in \{1, ..., K\}$ given input x, encoding network f_θ and cluster size K. Inspired by RIM [88] which learns a probabilistic clustering model $p(y|x, \mathcal{W})$ with parameters $\mathcal{W}$ to maximize the mutual information between x and y, we represent the

estimated mutual information [23] between x and y with network f_θ as the difference between marginal entropy $H(Y)$ and conditional entropy $H(Y|X)$. Our clustering objective maximizes the mutual information $I(X;Y)$ via minimizing loss $\mathcal{L}_{MI}$:

$$\begin{aligned}\mathcal{L}_{MI} &= -I(X;Y) = H(Y|X) - H(Y) \\ &= \frac{1}{N}\sum_{i=1}^{N} h(p(y_i|x_i, f_\theta)) - h(\frac{1}{N}\sum_{i=1}^{N} p(y_i|x_i, f_\theta))\end{aligned} \tag{4.1}$$

where h is the entropy function and $p(y_i|x_i, f_\theta)$ is calculated through f_θ and K-way softmax function. Intuitively minimizing conditional entropy $H(Y|X)$ will map similar x to have similar clustering predictions y and maximizing the entropy $H(Y)$ will incorporate the notion of class balance to avoid degenerate solution such as all the points map to one class.

4.3.3 The Cluster-level Explanation Objective

In addition to the unlabeled data X, to provide high-level explanations we assume a set of partial annotated tags $T = \{t_1, ...t_N\}$ where $t_i = (t_{i1}, ..., t_{iM}) \in \mathbb{R}^M$ is a binary vector. In real world applications assuming each instance has a complete set of annotated tags is unlikely, thus we assume each instance's tag can be missing with a specified probability r. With the predicted clustering assignments Y we can partition both X and T into K clusters.

We formulate the cluster-level description problem as an Integer Linear Programming (ILP) to learn short and orthogonal descriptors for each cluster. Specifically, we solve for the $K \times M$ binary allocation matrix W where $W_{i,j} = 1$ iff cluster C_i is described by tag j. The main objective function is to find the most concise overall cluster description:

$$\arg\min_{W} \sum_{i,j} W_{ij} \tag{4.2}$$

Our first constraint set includes the explanation length requirement for each cluster explicitly and set coverage constraints implicitly. Given a fixed number of tags as explanations for discovered cluster C_i, a high coverage explanation indicates that most instances within cluster C_i contain the selected tags. Now we define the fraction of cluster i having

tag j as $Q_{ij} = \frac{1}{|C_i|}\sum_{t_k \in C_i} t_{kj}$. Note we use mean imputation for missing tags. Formally we expect at least α tags being selected to explain each cluster:

$$\sum_{j=1}^{M} W_{ij} Q_{ij} \geq \alpha \quad \forall i \in \{1, ..K\} \tag{4.3}$$

Combining Eq (4.3) with our main objective in Eq (4.2), the constraint set in Eq (4.3) will naturally require the ILP solver to select tags that have higher coverage within each cluster.

Our next orthogonal constraint requires that the tags chosen to represent each cluster have minimum overlap which encourages informative explanations. Denote the hyper-parameter β as the upper-bound of expected number of overlapping tags per cluster, the orthogonal constraint can be encoded as follow:

$$\sum_{i=1}^{K} W_{ij} Q_{ij} \leq \beta \quad \forall j \in \{1, ..M\} \tag{4.4}$$

Lastly we have the regular constraints to make W a valid binary matrix as $W_{ij} \in \{0, 1\}$. There are KM variables to solve and $K + M$ constraints for set coverage and orthogonal requirements. Our proposed ILP objective can be solved efficiently due to the cluster-level explanation design ($K \ll N$). Empirical results have shown that our ILP module's running time only takes 1% of the whole framework's training time.

Now we define the tag space mapping function g which is used in our next objective. Let the solution for our proposed cluster-level explanation problem be W^*. We define function g for all the data as $g(t_i) = t_i * G$ where $G \in \mathbb{R}^{M \times M}$ is a diagonal matrix such that $G_{jj} = 1$ iff tag j is used in explanation allocation matrix W^*. Note function g can be treated as a mask function to filter out less informative semantic tags solved by our proposed cluster-level explanation objectives.

4.3.4 Self-generated Pairwise Loss Term

Our first proposed objective trains network f_θ for clustering; the second objective solves for explanations and a tag space function g. We propose a pairwise loss objective to reconcile inconsistencies between the two by finding instances that share similar informative descriptors but from different clusters, that is $g(t_i) \approx g(t_j)$ but $f_\theta(x_i) \neq f_\theta(x_j)$. To

achieve this we introduce a pairwise loss objective to bridge the explanation and clustering module. This part is important because our goal is to use semantic tags to generate explanations and reshape the clustering features for better clustering. Previous works on constrained clustering [136, 12] have shown that adding pairwise guidance such as *together* and *apart* constraints to clustering modules can largely improve clustering performance. However, these algorithms assume the pairwise guidance is given as ground-truth. In our setting we propose to add self-generated pairwise constraints with the assumption that instances which are close in tag space should be close in clustering feature space. Formally for each instance x_i we search for top l instances which minimize the objective J for self-generated *together* constraints:

$$J_i = \min_{j \in \{1,\dots,N\}} \gamma * |g(t_i) - g(t_j)| - |f_\theta(x_i) - f_\theta(x_j)| \tag{4.5}$$

where γ is the penalizing weight for tag space's difference. Minimizing J requires accessing the whole training set which is inefficient for mini-batch training. Instead we replace N with batch size N_B and solve an approximated version of Eq (4.5) in each mini-batch. We generate l pairs of together constraints for each instance x_i and then directly minimize the KL divergence between the clustering predictions y_i and y_j:

$$\mathcal{L}_P = \frac{1}{Nl} \sum_{i=1}^{N} \sum_{j=1}^{l} KL(p(y_i|x_i, f_\theta), p(y_j|x_j, f_\theta)) \tag{4.6}$$

Eq (4.6) minimizes the inconsistency between the clustering feature space and the semantic tag space and reshapes the clustering feature space for better clustering and explanation.

4.3.5 Overall Training Algorithm

Algorithm 3 presents our training algorithm for the deep descriptive clustering. Firstly we initialize the clustering network f_θ with random weights and initialize the weight matrix G of function g as identity matrix. Then we minimize the overall loss $\mathcal{L}$ by combining the clustering objective $\mathcal{L}_{MI}$ and pairwise loss term $\mathcal{L}_P$ with weight λ:

$$\begin{aligned} \mathcal{L} = & \frac{\lambda}{Nl} \sum_{i=1}^{N} \sum_{j=1}^{l} KL(p(y_i|x_i, f_\theta), p(y_j|x_j, f_\theta)) + \\ & \frac{1}{N} \sum_{i=1}^{N} h(p(y_i|x_i, f_\theta)) - h(\frac{1}{N} \sum_{i=1}^{N} p(y_i|x_i, f_\theta)) \end{aligned} \tag{4.7}$$

Algorithm 3 Algorithm for Deep Descriptive Clustering

Input: Data $X = \{x_1, ..., x_N\}$, tags $T = \{t_1, ..., t_N\}$, number of clusters K, hyper-parameters $\alpha, \beta, \gamma, \lambda$.

Output: Clustering partitions $\{C_1, ...C_K\}$, well-trained f_θ and g, explanation allocation matrix W^*.

Initialize network f_θ and tag space function g.
Pre-train f_θ via back-propagating overall loss in Eq (4.7).
repeat
    Construct cluster-level explanation problem as ILP defined in Eq (4.2,4.3,4.4). Initialize $\beta = 0$, $W^* = \emptyset$.
    while ILP solution W^* is not feasible **do**
        Increase hyper-parameter β by the fixed step size 1.
        Solve the ILP for W^* and tag space function g.
    end while
    for each mini-batch **do**
        Generate pairwise constraints based on the objective J in Eq (4.5) within each batch.
        Calculate the pairwise loss $\mathcal{L}_P$ via Eq (4.6) and the clustering loss $\mathcal{L}_{MI}$ via Eq (4.1).
        Update network parameters f_θ by minimizing overall loss $\mathcal{L}$ in Eq (4.7).
    end for
until Network f_θ and explanation results converge

Given the clustering predictions we construct the cluster-level explanation problem with binary variable W and calculate Q values for all the discovered clusters $\{C_1, ...C_K\}$. Note given the expected number of tags used for each cluster as α, we run our ILP solver iteratively with linear search for the smallest feasible hyper-parameter β to ensure tightest orthogonal constraints. Once the binary explanation allocation matrix W^* is solved, we update the tag space function g and regenerate the pairwise constraints via objective J to maximize the consistency between clustering features and tag space. The whole model

is trained repetitively until convergence.

4.4 Experiments

In this section, we conduct experiments to evaluate our approach empirically. Based on our experiments, we aim to answer the following questions:

- Can our proposed approach generate better explanations compared to existing methods? (see Sec 4.4.2) Can it generate more complex explanations such as ontologies (see Sec 4.4.3)?
- How does our proposed approach perform in terms of clustering quality? (see Sec 4.4.4)
- How does simultaneously clustering and explaining improve our model's performance? (see Sec 4.4.5)

4.4.1 Experimental Setup

Data. We evaluate the performance of our proposed model on two visual data sets with annotated semantic attributes. We first use Attribute Pascal and Yahoo (aPY) [53], a small-scale coarse-grained dataset with 64 semantic attributes and 5274 instances. We have selected 15 classes for our clustering task. Further, we have studied Animals with Attributes (AwA) [89], which is a medium-scale dataset in terms of the number of images. For AwA we use 85 semantic attributes and 19832 instances, we have set 40 classes for clustering, the total number of animals.

Baselines and Evaluation Metrics. In the experiments, deep descriptive clustering is compared with descriptive clustering [37] in terms of the explanation quality. To evaluate the generated explanations quantitatively and qualitatively, we list all the composition and selected tags for each discovered cluster and report the *Tag Coverage* (TC) and *Inverse Tag Frequency* (ITF). For cluster C_i, let the descriptive tag set be D_i, the TC and ITF for C_i are calculated as:

$$TC(C_i) = \frac{1}{|D_i|} \sum_{d \in D_i} \frac{|\{(x,t) \in C_i : d \in t\}|}{|C_i|} \tag{4.8}$$

$$ITF(C_i) = \frac{1}{|D_i|} \sum_{d \in D_i} \log \frac{K}{\sum_{j=1}^{K} |d \in D_j|} \tag{4.9}$$

The *Tag Coverage* for C_i ranges from $[0, 1]$ and the max value is achieved when each descriptive tag exists in all the instances within C_i. The *Inverse Tag Frequency* for C_i ranges from $[0, \log K]$ and the max value is achieved when each descriptive tag is only used once. For both TC and ITF the *larger* the better. We have also generated a *graphical ontology* as high-level explanation on the clustering results when the number of clusters is large and a long explanation list is problematic. We evaluate its quality by comparing it to human knowledge. Further, we evaluate the clustering performance with a range of tag annotated ratios r as $[10\%, 30\%, 50\%]$ and compare DCC's results against vanilla k-means clustering and competitive incomplete multi-view clustering approaches such as MIC, IMG, and DAIMC [121, 162, 75]. For the clustering evaluation metric, we choose to use both Normalized Mutual Information (NMI) [125] and Clustering Accuracy (ACC) [148] for comprehensive evaluation.

Implementations. For a fair comparison with all the baseline approaches, we use pre-trained ResNet-101 [70] features for all the clustering tasks and the encoder networks of deep descriptive clustering model are stacked by three fully connected layers with size of $[1200, 1200, K]$ where K is the desired number of clusters. We set the expected number of tags for each cluster as 8 and hyper-parameters l, λ, γ as $1, 1, 100$ respectively. The tag annotated ratio r is set as 0.5 by default to simulate a challenging setting. The activation function is ReLU, and the optimizer is Adam [84] with default parameters.

4.4.2 Comparison with Descriptive Clustering

We duplicate the experimental setting in [37] by down-sampling 10 classes from AwA and cluster the data into 5 clusters for a fair comparison. We list the explanation results in Table 4.1 and 4.2. Our model's *Tag Coverage* values for all the clusters are 1; this result shows that our model successfully maximizes the consistency between the discovered tag space and clustering feature space so that similar instances with similar tags are grouped. Moreover, the *Inverse Tag Frequency* values of our model are much higher than

C	Composition by animals	Description by tags	TC ↑	ITF ↑
C1	1 grizzly bear, 2 dalmatian, 1 horse, 2 blue whale	big, fast, strong, muscle, new world, smart	0.94	1.34
C2	5 antelope, 2 grizzly bear, 2 beaver, 5 dalmatian, 5 Persian cat, 5 horse, 6 German shepherd, 3 Siamese cat	furry, chew teeth, fast, quadrupedal, new world, ground	0.98	0.94
C3	69 beaver, 64 dalmatian, 42 Persian cat, 29 blue whale, 42 Siamese cat	tail, fast, timid, smart, new world, solitary	0.98	1.17
C4	100 killer whale, 69 blue whale, 1 Siamese cat	tail, fast, fish, smart	1.00	1.10
C5	95 antelope, 97 grizzly bear, 29 beaver, 29 dalmatian, 53 Persian cat, 94 horse, 94 German shepherd, 54 Siamese cat	furry, chew teeth, fast, quadrupedal, new world, ground	1.00	0.94

Table 4.1: Results generated by descriptive clustering [37], we present the first Pareto point of their result such that the diameter of all the clusters are minimized. ↑ means the larger value the better.

the competing method. This result indicates that our model selects informative tags for each cluster that differentiate from other discovered clusters. We also observe that our proposed model generates high-quality clusters where similar animals are correctly grouped together. Finally, we have found one attribute's annotation error in the AwA data when examining our explanations for C_1; the beavers are annotated with attribute *hibernate* but the truth is the opposite. This finding suggests that the labeled attributes are noisy in the AwA data set.

4.4.3 Novel Explanation as Ontology Extraction

Interpreting the descriptions for each cluster can be problematic when the number of clusters is large and the description list is long. We propose to generate a graphical ontology that not only describes each cluster but shows the relationships between them to better inform people. We have visualized the ontology graph for aPY in Figure 4.3. The nodes represent discovered clusters and the name of the nodes corresponds to the

C	Composition by animals	Description by tags	TC ↑	ITF ↑
C1	100 beaver, 100 grizzly bear	tough skin, bulbous, paws, quadrupedal, nocturnal, hibernate, smart, solitary	1.00	2.32
C2	100 Siamese cat, 100 Persian cat	white, gray, pads, chew teeth, claws, weak, inactive, old world	1.00	2.32
C3	100 antelope, 100 dalmatian	furry, big, long leg, active, ground, timid, group, new world,	1.00	2.32
C4	100 killer whale, 100 blue whale	spots, hairless, flippers, strain teeth, fish, plankton, arctic, ocean	1.00	2.32
C5	100 horse, 100 German shepherd	black, brown, patches, smelly, walks, strong, agility, domestic	1.00	2.32

Table 4.2: Results generated by our proposed DDC. ↑ means the larger value the better.

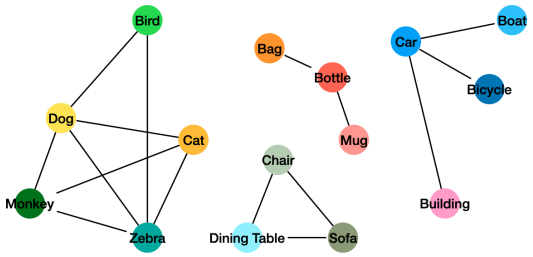


Figure 4.3: The graphical ontology generated for aPY data set.

majority class within each cluster. When two clusters share at least q tags ($q = 3$ in our experiments) we add an edge between these two clusters. This shows a higher level of structure as we can see the ontology plot in Figure 4.3 reflects four distinct communities which are animals, transportation tools, furniture, and small objects. Those ontologies are generally in line with human knowledge and provide a high-level abstraction explanation of our deep descriptive clustering model.

4.4.4 Evaluating Clustering Performance

Here we report if the descriptive clustering problem can increase clustering quality. Since these methods are not deep learning based, to make a fair comparison we use the same pre-trained ResNet-101 features. We report the clustering results of our model under

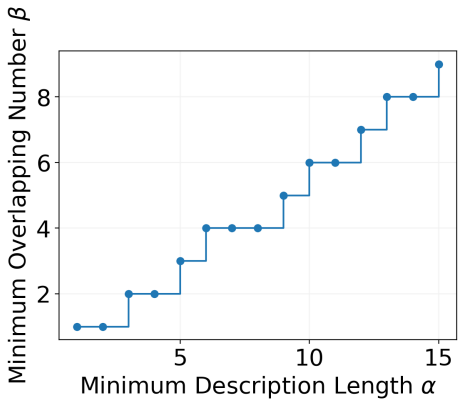
		NMI			ACC		
Datasets	Methods $r\%$	10	30	50	10	30	50
AwA	K-Means	71.67 ± 0.63	73.72 ± 0.66	74.23 ± 0.69	66.21 ± 0.57	67.98 ± 0.60	68.24 ± 0.54
	IMG	71.86 ± 2.41	74.43 ± 2.69	82.16 ± 3.01	66.19 ± 2.05	69.17 ± 2.25	76.24 ± 2.78
	MIC	72.40 ± 1.68	76.85 ± 1.71	83.43 ± 1.89	67.26 ± 1.45	70.52 ± 1.58	77.68 ± 1.84
	DAIMC	72.88 ± 2.38	79.02 ± 2.46	87.10 ± 2.74	67.87 ± 1.97	73.14 ± 2.13	82.34 ± 2.39
	Ours DDC	$\mathbf{75.62 \pm 1.17}$	$\mathbf{83.93 \pm 1.35}$	$\mathbf{89.57 \pm 1.37}$	$\mathbf{71.19 \pm 0.93}$	$\mathbf{78.74 \pm 1.12}$	$\mathbf{84.48 \pm 1.20}$
aPY	K-Means	63.08 ± 0.45	63.89 ± 0.42	64.38 ± 0.48	57.11 ± 0.39	58.13 ± 0.36	58.98 ± 0.37
	IMG	64.75 ± 2.05	70.19 ± 2.19	77.50 ± 2.37	60.18 ± 1.78	65.72 ± 1.90	71.21 ± 1.96
	MIC	65.36 ± 1.49	73.89 ± 1.61	80.38 ± 1.83	62.36 ± 1.28	66.98 ± 1.40	72.42 ± 1.53
	DAIMC	69.29 ± 1.82	80.70 ± 1.91	84.24 ± 1.97	68.21 ± 1.54	73.63 ± 1.63	76.11 ± 1.68
	Ours DDC	$\mathbf{70.54 \pm 0.98}$	$\mathbf{82.41 \pm 1.15}$	$\mathbf{86.30 \pm 1.22}$	$\mathbf{69.30 \pm 0.86}$	$\mathbf{76.34 \pm 0.95}$	$\mathbf{79.87 \pm 1.02}$

Table 4.3: Comparison of clustering performance averaged over 10 trials (mean $\pm$ var) on AwA and aPY under different tag annotated ratio $r\% \in \{10, 30, 50\}$. Bold results are the best mean results among all the algorithms.

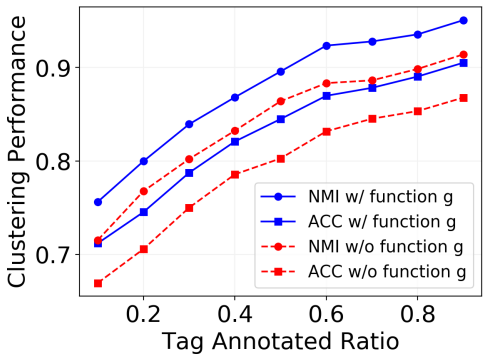
a range of annotated ratios in Table 4.3. We have several observations to highlight: firstly our model consistently outperforms the k-means and multi-view clustering baselines with different tag annotated ratios; secondly with more annotated tags, both multi-view clustering baselines and our model improves largely comparing to the k-means clustering which naively concatenates the images features with semantic attributes. We attribute the good clustering performance for both deep representation learning and our novel way of leveraging semantic tag information for better clustering.

4.4.5 Parameter Analysis and Ablation Test

Given the hyper-parameter α which denotes the minimum expected number of tags for description, we plot the automatically searched parameter β for AwA in Figure 4.4 (a). This result shows our automatic searching procedure's success and suggests that a relatively small α leads to more concise and orthogonal explanations. Meanwhile, we conduct ablation experiments to analyze the impact of mask function g solved via our explanation module. In Figure 4.4 (b), the blue lines indicate clustering results with function g. In red lines we replace function g with an identity function to conduct the ablation study. Comparing red and blue lines we can see that mask function g can remove the noisy information within semantic tag space and consistently improve the clustering performance.



(a) Parameter analysis on α



(b) Ablation study on function g

Figure 4.4: Plots for parameter analysis and ablation study

4.5 Conclusion and Future Work

This paper proposes deep descriptive clustering, which can learn to cluster and generate cluster-level explanations simultaneously. We develop a novel deep learning framework that supports learning from the sub-symbolic level (which clustering is performed on) and symbolic level (which explanations are created from). Empirical results on real-world data demonstrate the high quality of our generated explanations and good clustering performance. Our future work will focus on building an explainable clustering model with noisy semantic features and exploring other novel forms of explanations beyond ontologies on different types of data.

Chapter 5

Towards Fair Deep Anomaly Detection

5.1 Introduction

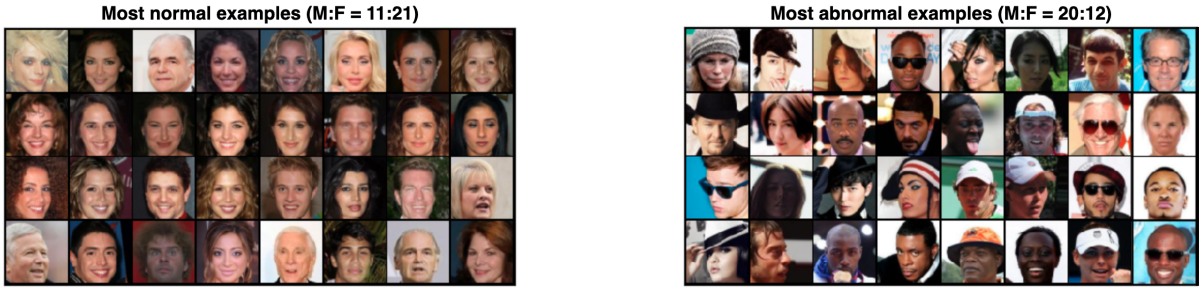
Anomalies are the unusual, unexpected, surprising patterns in the observed world that warrant further investigation. Classic work [67] defines outliers as an observation that deviates so significantly from other observations as to arouse suspicion that a different mechanism generated it. Anomalies and outliers are often used interchangeably though we note that some use the term differently [29] and for this paper we use the term anomalies. The goal of an anomaly detection algorithm is given a set of instances to determine which instances stand out as being dissimilar to other instances. Effective detection of anomalies can be used for various applications, such as stopping malicious intruders, fraud detection, system health monitoring, and medical image analysis [28].

Recent algorithmic developments have proposed many novel deep learning methods for anomaly detection [52, 31, 108, 60, 105, 71]. This previous works on deep anomaly detection are typically unsupervised (e.g., assume all training data are from the normal group) and have demonstrated better anomaly detection performance than traditional anomaly detection approaches. One popular approach to deep anomaly detection is the deep support vector data description (deep SVDD) [108]. This work attempts to transform the input data into a new feature space where all the points are closely clustered into a predetermined center. Hence, by definition, those points that cannot be projected to be

close to the center are deemed anomalies. The anomaly scores are calculated based on the Euclidean distances between the test instances and the predetermined center during the test time. Deep SVDD is a general approach which can be applied to both low dimensional and high dimensional data. In this first paper on the topic we focus on adding fairness to deep SVDD.

Since anomaly detection is often applied to humans who are then suspected of unusual behavior, ensuring fairness becomes paramount. The notion of fairness has recently received much attention in supervised learning [151, 48] and unsupervised learning [35, 114, 8]. Measures of fairness can generally be divided into two categories [36]: (i) group-level fairness and (ii) individual level fairness. In anomaly detection problems, we divide the data into two groups, which are the normal group and the abnormal group. We propose to study the group-level fairness problems which ensure that no one particular group contains a disproportionate number of individuals with protected status. To our best knowledge, there is no prior published work on fairness in the context of deep anomaly detection though work on auditing (i.e., checking) anomaly detection algorithms exist [42].

A Motivating Example For Group-Level Fairness. Consider the example of finding anomalies by applying deep SVDD to facial images. The top 32 normal instances and top 32 abnormal instances are shown in Figure 5.1. These pictures are from the celebA (celebrity) data set (which we introduce in section 5.5.1). The deep SVDD model is trained on attractive celebrity faces (normal group) and used to detect plain celebrity faces (abnormal group) where the labels are given in the data set. The model performs well in terms of the anomaly detection quality as most attractive celebrity faces and plain celebrity faces are separated correctly. However, when we consider the protected status variable gender in this problem, more females are predicted to be attractive (normal group), and more males are predicted as plain (abnormal group). Moreover, if we consider race as a protected status variable, we can see that the most attractive faces are white people and many black people in the abnormal group. Motivated by these observations, we aim to design experiments to examine the fairness of existing deep anomaly detection



(a) Normal Group (b) Abnormal Group

Figure 5.1: Motivating example of the need for group-level fairness in deep anomaly detection problem. We visualize the top 32 normal instances and top 32 abnormal instances discovered by deep SVDD on celebA data set. We see that the normal group is dominated by females while the abnormal group is dominated by males.

methods quantitatively and propose a fair anomaly detection model to balance the number of instances with different sensitive attribute values in the anomaly predictions.

In this paper, we present the Deep Fair Support Vector Data Description (*Deep Fair SVDD*) model which learns a compact and fair description of the normal data via adversarial learning. We summarize the main contributions in this paper as follows:

- We show existing deep anomaly detection approaches are unfair (see section 5.5.4) due to the deep learners' ability to extract out complex features.
- We consider fair anomaly detection in the context of deep representation learning. To the best of our knowledge, this is an under-studied so far and challenging due to the need for fair and high-quality predictions.
- We address these challenges by proposing a novel fair anomaly detection architecture (see Figure 5.3) and use adversarial learning to remove the unfairness. The idea of using adversarial learning contrasts with many recent works on fairness in learning which typically encodes fairness as a regularization term or a constraint.
- We propose two measures of group-level fairness for deep anomaly detection problems: i) A demographic parity motivated fairness measure for the abnormal group (equation 5.3) ii) A parameter-free measure based on Wasserstein distance for calculating the overall fairness (equation 5.4).

- We demonstrate our method on several types of data, including traditional tabular datasets, face data sets, and digit images. We study the fairness problem concerning gender, racism, and the source of the visual objects. (see section 5.5.1). We find that introducing fairness causes a marginal drop in anomaly detection performance measured by the AUC score (see section 5.5.5).

- We conduct an in-depth analysis of our proposed model to show our proposed model's strengths and limitations, including parameter analysis, feature visualization, and run-time analysis. (see section 5.5.6, 5.5.8, 5.5.9).

Our paper structure is as follows. In the next section 5.2, we discuss the related work. Then, we provide background knowledge about deep SVDD and our fairness measures in section 5.3. Next, we propose the deep fair SVDD model and analyze how we use adversarial networks to tackle fair anomaly detection problems (section 5.4). Finally, we perform experiments on real-world data sets to demonstrate the effectiveness of our method in section 5.5 and conclude our proposed approach in section 5.6.

5.2 Related Work

Deep Anomaly Detection. We first outline related works on deep anomaly detection. One of the most common deep anomaly detection approaches is reconstruction-based methods [68, 98, 142, 4, 110, 31, 77] which assume the anomalies possess different features than the normal instances. Hence, given a pre-trained autoencoder over the normal instances it will be hard to compress and reconstruct the anomalies. The anomaly score in this research is defined as the reconstruction loss for each test instance. Inspired by the generative adversarial networks [61], another line of related works [113, 44, 152] score an unseen sample based on the ability of the model to generate a similar one.

More recently, A deep version of support vector data description (Deep SVDD) has been proposed [108]. This work is inspired by kernel-based one-class classification [115] which combines the ability of deep representation learning with the one-class objective to separate normal data from anomalies by concentrating normal data in embedded space while mapping anomalies to distant locations. Another recent progress on deep anomaly

detection uses self-supervised learning on image data sets and achieves excellent performance [59, 60, 71, 139]. For example, [60] uses a composition of image transformations and then trains a neural network to predict which transformation was used. The anomaly scores are computed based on the predictions' confidence over different image transformations given the test samples.

Fairness in Anomaly Detection. With so many works focusing on improving the deep anomaly detection performance, our work differentiates from those previous works as we investigate the fairness of the existing deep anomaly detection problems and propose a novel deep fair anomaly detection model to help humans make fair decisions. To the best of our knowledge, there is no work on deep fair anomaly detection algorithms. We now introduce two related works on non-deep fair anomaly detection problems. Recent work [42] has studied auditing the output of any anomaly detection algorithm. In their work, the anomaly detection algorithm's output fairness with respect to multiple protected status variables (PSVs) is measured by finding PSV combinations in the outlier group which are more common than in the normal group. Their empirical results show that the output of five classic anomaly detection methods is unfair. Another work [45] studies the fairness problem of LOF (Local Outlier Factor) [22] and proposes several heuristics to mitigate the unfairness within LOF on tabular data sets. Differently, our work proposes to examine fairness for the deep anomaly detection problems which work for both tabular data and image data. Moreover, unlike LOF-based approaches that have no training phase and do not learn a model of normality, our proposed model can make out-of-sample predictions.

Adversarial Learning for Fairness. Lastly, we introduce the related works which take the advantages of adversarial networks to remove unfairness. [16] applies an adversarial training method to satisfy parity for salary prediction. This work shows that small amounts of data are needed to train a powerful adversarial model to enforce fairness constraints. The work of [153] uses a predictor and adversary with an additional projection term to remove unfairness in both supervised learning tasks and debiasing word embedding tasks. [51] shows that demographic information leaks into intermediate representations of neural networks trained on text datasets and applies adversarial learning

to mitigate the information leaks. [127] takes the advantages of adversarial networks to reduce word vector sentiment bias for demographic identity terms.

5.3 Preliminary

5.3.1 Deep Support Vector Data Description

Among the recent deep anomaly detection methods we focus on deep SVDD [108] as a base learner because it is not only a popular method but also performs well on both low dimensional (tabular data) and high dimensional data (images). Unlike generative models or compression based anomaly detection models which are adapted for anomaly detection, deep SVDD is directly learned with an anomaly detection based objective. Given the training data of just normal points $\mathcal{X} \in \mathbb{R}^{n \times d}$, the deep SVDD network is trained to map all the n normal points close to a fixed center $\mathbf{c}$ where $\mathbf{c}$ is normally set as the mean of the points. Denote function f as a neural network with parameters θ the simplified objective function is:

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} ||f(\mathbf{x}_i; \theta) - \mathbf{c}||^2 + \frac{\alpha}{2} \sum_{\ell=1}^{L} ||\theta^{\ell}||^2 \tag{5.1}$$

The second term is a network weight decay regularizer with hyper-parameter $\alpha > 0$ which prevents finding a too complex mapping function. The network has L hidden layers and set of weights $\{\theta^1, ..., \theta^L\}$ are the weights of layer $\ell \in \{1, ..., L\}$. Deep SVDD contracts the embedding space enclosing the points by minimizing the mean distance of all data points to the center. During the evaluation/scoring stage, given a test point $\mathbf{x} \in \mathcal{X}^T$ Deep SVDD will calculate the anomaly score $s(\mathbf{x})$ for $\mathbf{x}$ as follows:

$$s(\mathbf{x}) = ||f(\mathbf{x}; \theta) - \mathbf{c}||^2 \tag{5.2}$$

Note this is just the distance the instance is from the center, abnormal points are then those far from the center.

5.3.2 Notion of Fairness

Fairness is measured using protected status variables or sensitive features such as gender and race. In this paper, we study group-level fairness which ensures that no one particular group contains a disproportionate number of instances of a given protected status.

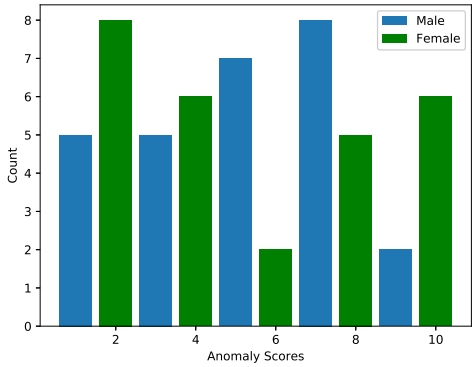
Fairness by $p\%$ -rule. Our first notion of fairness is inspired by [151] which proposed a statistical parity motivated measure for a supervised classification model. Statistical parity is a popular fairness measure used in many unsupervised learning and supervised learning problems [35, 8, 151, 122]. Let t be the anomaly score threshold, then the normal groups are points with $s(\mathbf{x}) \leq t$ and the abnormal groups are points with $s(\mathbf{x}) > t$. Given the protected status variable as $z \in \{0, 1\}$, our definition of fairness measure leverages the 80% rule [18]: a normal / abnormal group partition satisfies the 80% rule if the ratio between the percentage of person with a particular protected status variable value having $s(\mathbf{x}) > t$ and the percentage of person without protected status having $s(\mathbf{x}) > t$ is no less than $80 : 100$. We define the $p\%$ -rule as our fairness measure for the anomaly detection problem:

$$\min\left(\frac{P(s(x) > t|z = 1)}{P(s(x) > t|z = 0)}, \frac{P(s(x) > t|z = 0)}{P(s(x) > t|z = 1)}\right) \geq \frac{p}{100} \tag{5.3}$$

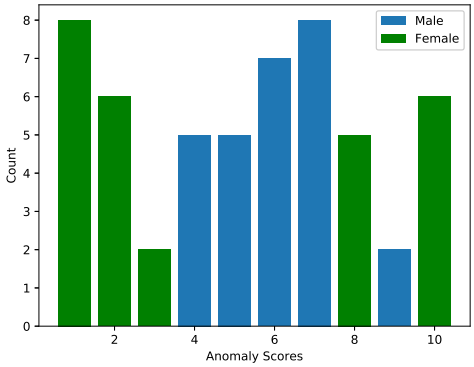
Note the $p\%$ -rule value ranges from 0 to 1 and a larger value indicates the model is fairer. In ideal case we have $P(s(x) > t|z = 1) = P(s(x) > t|z = 0)$. Maximizing $p\%$ -rule means predicting x as an anomaly will be independent of the protected status variable z

The rationale behind using our first fairness measure in equation 5.3 is because it is closely related to the 80% rule advocated by the US Equal Employment Opportunity Commission [18]. We can determine a deep anomaly detection model's fairness using the 80% rule. However, there are some limitations to our first proposed measurement. Firstly, we need to know the exact number of anomalies in the test set to correctly set the anomaly score threshold t to partition the normal and abnormal groups. Secondly, this measure only considers the fairness in the abnormal group.

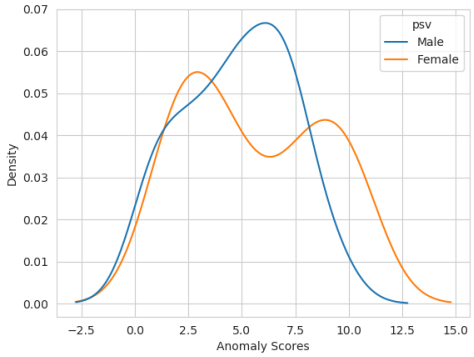
Fairness by distribution distance. Here we propose a new fairness measure for anomaly detection problems which is invariant of the anomaly score threshold t and covers both normal and abnormal groups. We have designed one synthetic anomaly detection problem to show the motivation for our second fairness measure. Assume there are two anomaly detection models named A and B. The test data includes 27 males and 27 females, and the binary sensitive attribute is *gender*. To be specific, the predicted anomaly scores



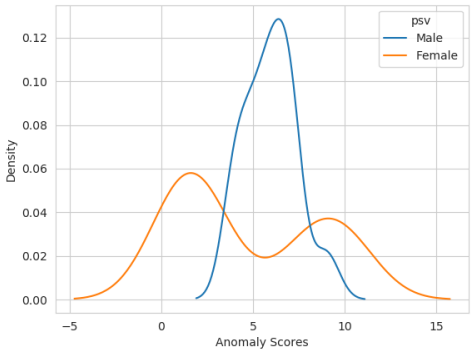
(a) Model A's Predictions



(b) Model B's Predictions



(c) Model A's Prediction Distribution



(d) Model B's Prediction Distribution

Figure 5.2: A toy example to show the difference between our proposed two fairness measures. Figure a, b summaries the statistics of predicted anomaly scores of model A and B. Given the ground-truth anomaly score threshold $t = 8$, model A and B have the same fairness by $p\%$ -rule as $2/6 = 0.33$. Figure c, d shows the anomaly score distributions for model A's predictions (M_A, F_A) and model B's predictions (M_B, F_B). Model B is more unfair as the anomaly scores are highly correlated with the sensitive attribute *gender* (M, F). The fairness by distribution distance for model A and B are $W(M_A, F_A) = 1.37$ and $W(M_B, F_B) = 2.87$.

from Model A and B are shown in Figure 5.2 (a) and (b). Given the ground truth number of anomalies as 8, we can set the anomaly score threshold $t = 8$ to predict anomalies with $s(x) > 8$. Now we can calculate the $p\%$ -rule for Model A and Model B as: $2/6 = \frac{1}{3}$. Although models A and B achieve the same fairness measured by $p\%$ -rule, we can learn from the anomaly score distributions in Figure 5.2 (c) and (d) that model B's predictions are highly correlated with the sensitive attribute *gender* which is less fair.

Now we formulate our second definition of fairness which quantifies the difference between each demographic group's anomaly score distributions: let $\mathbb{P}$ denotes the distribution of the anomaly scores for test instances with sensitive attribute $z = 0$ and $\mathbb{Q}$ for test instances with sensitive attribute $z = 1$. We calculate the Wasserstein-1 (*Earth-Mover Distance*) distance between distribution $\mathbb{P}$ and $\mathbb{Q}$ as fairness by distribution distance measure:

$$W(\mathbb{P}, \mathbb{Q}) = \inf_{\gamma \in \Pi(\mathbb{P}, \mathbb{Q})} \mathrm{E}_{(x,y)\sim\gamma}[||x - y||] \tag{5.4}$$

where $\Pi(\mathbb{P}, \mathbb{Q})$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}$ and $\mathbb{Q}$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distribution $\mathbb{P}$ to $\mathbb{Q}$. For our previous toy example, we calculate the *Distribution distance* for model A and B's predictions as 1.37 and 2.78. These results indicate that model A is overall fairer than model B. From a practitioner's perspective, we can use distribution distance to evaluate the fairness performance for different anomaly detection models and conduct model selection when we have no access to the test set. Lastly, we will use both Fairness by $p\%$ *-rule* and Fairness by distribution distance to evaluate the fairness performance in our experimental section.

5.4 Methods

5.4.1 Learning Overview

In this section, we propose the deep fair SVDD model for deep anomaly detection problems. Following the previous deep anomaly detection works [108, 59], we assume the training data $\mathcal{X}$ contains only normal instances. Moreover, our proposed model requires access to the binary protected status variable Z for each of the training instances $\mathcal{X}$. We learn $f(\theta)$ as an encoder network to learn compact descriptions of $\mathcal{X}$ (i.e. a mapping to a lower-dimensional space), and a classification network $g(\theta_d)$ to predict protected status variable value $z \in Z$ based on the learned embedding $f(\mathcal{X}; \theta)$. We train the encoder f and discriminator g using adversarial training so that we hope the embedding learned via encoder f can fool the discriminator g. Training such a network is challenging and we take advantage of adversarial learning since it has shown promising results for other fair-

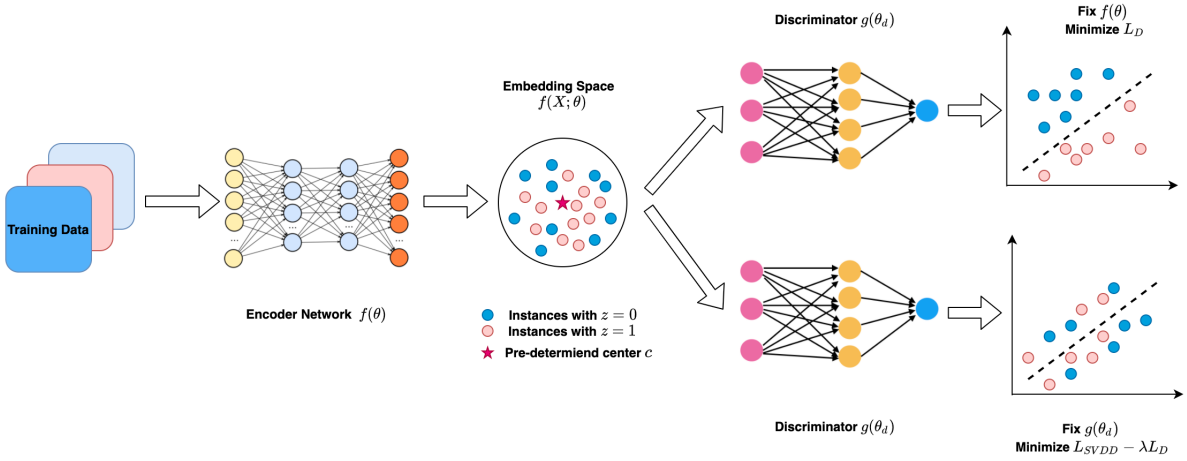


Figure 5.3: Pipeline of the proposed deep fair SVDD learning framework. The inputs are normal training data $\mathcal{X}$ and the outputs are learned embedding $f(\mathcal{X};\theta)$ and a discriminatory function $g(\theta_d)$. The end-to-end learning process contains three steps: 1) train the encoder $f(\theta)$ via minimizing the loss L_{SVDD}, 2) fix the encoder's parameters θ, and train the discriminator $g(\theta_d)$ via minimizing the discriminator's loss L_D, 3) fix the discriminator's parameters θ_d and train encoder $f(\theta)$ to minimize the adversarial loss $L_{SVDD}-\lambda L_D$. Procedure (2) and (3) are trained alternatively until convergence.

ness tasks such as removing unfairness in NLP applications [51, 127]. We use adversarial learning to de-correlate the relationships between protected status variable Z and feature vectors encoded via $f(\theta)$. Note that our fair learning method is fundamentally different from much existing work [151, 27, 74] which uses a regularization term to encode fairness or encodes fairness as a constraint.

5.4.2 Deep Fair SVDD Model

Our proposed deep fair SVDD network aims to learn a fair representation to describe all the training data via adversarial learning. Given the normal training data $\mathcal{X} \in \mathbb{R}^{M\times D}$, encoder network $f(\theta)$ we have the latent encoding of all the normal points as $f(\mathcal{X};\theta)$. Assume the binary protected status variable is $Z \in \mathbb{R}^{M\times 1}$. The fair representation is achieved when the learned embedding are statistically independent of sensitive attribute Z. Given $z \in \{0,1\}$ we hope to optimize the function $f(\theta)$ to have:

$$\mathbf{p}(f(\mathcal{X};\theta)|z=0) = \mathbf{p}(f(\mathcal{X};\theta)|z=1) \tag{5.5}$$

To achieve the goal in equation (5.5) we propose to use adversarial networks with a min-max game strategy to constrain the embedding function $f(\theta)$. Firstly, the encoder network is trained with normal points $\mathcal{X}$ to generate compact embedding around a pre-determined center $\mathbf{c}$. To regularize the encoder we add a weight decay regularizer with positive hyper-parameter α for all the L hidden layers. We use term L_{SVDD} to represent the encoder's loss function:

$$L_{SVDD} = \frac{1}{M}\sum_{i=1}^{M}||f(\mathbf{x}_i;\theta) - \mathbf{c}||^2 + \frac{\alpha}{2}\sum_{\ell=1}^{L}||\theta^{\ell}||^2 \tag{5.6}$$

Secondly we concatenate the encoding network $f(\theta)$ with a discriminator $g(\theta_d)$ to learn to classify the sensitive attributes Z based on learned embedding $f(\mathcal{X};\theta)$. Since Z is a binary variable we use sigmoid function to get the probabilistic prediction as $\hat{z}_i$:

$$\hat{z}_i = \frac{1}{1+\exp^{-g(f(\mathbf{x}_i;\theta)|\theta_d)}} \tag{5.7}$$

We choose cross entropy loss to train discriminator $g(\theta_d)$ as:

$$L_D = -\frac{1}{M}\sum_{i=1}^{M}(z_i * \log(\hat{z}_i) + (1-z_i) * \log(1-\hat{z}_i)) \tag{5.8}$$

To make the learned embedding $f(\mathcal{X};\theta)$ invariant with sensitive attributes Z we hope to tune the embedding function $f(\theta)$ to fool the discriminator $g(\theta_d)$. Meanwhile, we hope the normal points are still closely clustered together so that we design the adversarial loss L_{Adv} as follows:

$$L_{Adv} = L_{SVDD} - \lambda L_D \tag{5.9}$$

where the hyper-parameter λ is a positive constant number. Minimizing the adversarial loss $L_{Adv} = L_{SVDD} - \lambda L_D$ is actually maximizing the discriminator's loss L_D. Note the discriminator's parameters θ_d are fixed when we back-propagate the adversarial loss. Similar as the generated adversarial networks [61], we propose to train the $f(\theta)$ and $g(\theta_d)$ in an alternative way until we find the min-max solution. The training procedure tries to jointly optimize both quantities:

$$\arg\min_{\theta_d} L_D \tag{5.10}$$

$$\arg\min_{\theta} L_{SVDD} - \lambda L_D \tag{5.11}$$

Once the joint training converges, the anomaly scores for all the instances are calculated as:

$$\mathcal{S} = ||f(\mathcal{X}; \theta) - \mathbf{c}||^2 \tag{5.12}$$

Note the instances with larger anomaly scores have larger probability to be predicted as anomalies. The pseudo-code for the learning algorithm is summarized in Algorithm 4. We also visualize the learning pipeline of deep fair SVDD model in Figure 5.3.

Algorithm 4 Algorithm for deep fair SVDD

Input: $\mathcal{X}$: training data, $\mathcal{X}^T$: test data, Z: Protected status variable, $f(\theta)$: encoder network, $\mathbf{c}$: pre-determined data center, $g(\theta_d)$: discriminator, K: initial training epochs, T: adversarial training epochs.

Output: $\mathcal{S}$: predicted anomaly score.

Train the encoder network $f(\theta)$ via minimizing L_{SVDD} in equation (5.6) for K epochs.

Fix the encoder network $f(\theta)$, train the discriminator $g(\theta_d)$ via minimizing L_D in equation (5.8) for K epochs.

for epoch from 1 to T **do**

    Fix the parameters θ for encoder network $f(\theta)$. Calculate L_D in equation (5.8) for each mini-batch.

    Back-propagate the discriminator loss L_D and update the parameters θ_d.

    Fix the parameters θ_d for discriminator $g(\theta_d)$. Calculate the loss L_{Adv} in equation (5.9) for each mini-batch.

    Back-propagate the adversarial loss L_{Adv} and update θ.

end for

Output the anomaly scores for test set $\mathcal{S} = ||f(\mathcal{X}^T; \theta) - \mathbf{c}||^2$.

5.4.3 Potential Extensions of Deep Fair SVDD

In this subsection, we analyze the design of our proposed deep fair SVDD and provide several potential extensions of our proposed learning framework that we intend to study:

5.4.3.1 Extensions to Fairness Problems with Multi-State PSV

Note we study the fairness problem with binary protected status variable $z \in \{0, 1\}$ in this work. However, our deep fair SVDD learning framework can be extended to solve fairness problems with multi-state protected state variable (e.g., education level, nationality) by changing the current binary discriminator g into a multi-class classification network.

5.4.3.2 Extensions to Fairness Problems with Multiple PSVs

Our framework can also support multiple protected status variables if we substitute the binary classification discriminator with a multi-class classification network. Given the fairness requirements on multiple protected state variables (say gender and race together), we can enumerate all the combinations via a Cartesian product of these two variables and transform them into a multi-state protected state variable to feed in our extended framework. This is an important property lacking in many fair classification methods as clearly making a model fairer with respect to say gender could make it unfair with respect to say race.

5.4.3.3 Extensions to Semi-supervised Anomaly Detection

The current encoder $f(\theta)$ is trained via an unsupervised loss function (5.6) to force all the normal data to be close to the pre-determined center c. Recently, some works on general semi-supervised anomaly detection [63, 109] have demonstrated superior performance. In general semi-supervised anomaly detection settings, we assume the learners have access to a small subset of labeled normal and abnormal instances. Our current learning framework can be modified to accommodate semi-supervised anomaly detection settings by combining the current loss function (5.6) with a new supervised classification loss for labeled anomalies in the training set.

5.5 Experiments

In this section, we conduct experiments to empirically evaluate our proposed approach. From our experiments, we aim to address following questions:

- Do existing deep anomaly detection algorithms produce unfair results? (see Section 5.5.4)
- How does our proposed algorithms work in two types of datasets: involving low dimensional data (COMPAS Recidivism) and high dimensional data (Facial images and digits)? (see Section 5.5.5)
- What is the sensitivity of the hyper-parameter λ in our proposed deep fair SVDD model (see Section 5.5.6)?
- How do our proposed algorithm change the latent feature space whilst making anomaly detection fairer? (see Section 5.5.7, 5.5.8)
- How efficient are our proposed algorithms? (see Section 5.5.9)

Dataset	Type	# Instances	# Dimension	Protected Status Variable	Normal Group	Abnormal Group
COMPAS Recidivism [5]	Tabular	3878	11	Race	Not reoffending	Reoffending
celebA [94]	Face	24000	64 x 64 x 3	Gender	Attractive faces	Plain faces
MNIST-USPS	Digits	7435	28 x 28 x 1	Source of digits	Digit 3	Digit 5
MNIST-Invert	Digits	15804	28 x 28 x 1	Color of the digits	Digit 3	Digit 5

Table 5.1: Characteristics of four datasets used in our experiments. Our methods requires the protected status variables such as Gender (Male and Female) and Race (African-American and non African-American) to be binary variables.

5.5.1 Data Sets

We propose to experiment on four public datasets which include visual data and tabular data. We list the characteristics of our selected datasets in Table 5.1 and introduce the details of how we construct those datasets as below. For each data set, only normal instances are in the training data set, but there are both normal and abnormal instances in the test data.

- COMPAS Recidivism [5]: The COMPAS recidivism data set consists of data from criminal defendants from Broward County, Florida. We create a binary protected status variable for whether the defendant is African American. Given the ProPublica collected label of whether the defendant was rearrested within two years, we set the normal group for people who were not reoffending and the abnormal group for reoffending. We select this data set to demonstrate our approach's performance on low-dimension tabular data.

- celebA [94]: This is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. We sample a subset of this data set and treat gender as a binary protected status variable in this data set. The normal group contains celebrity faces labeled as attractive, and the abnormal group contains the celebrity faces labeled as plain. We choose celebA data set to test our approach on high-dimension images.

- MNIST-USPS: This dataset consists of MNIST and USPS images, which include different style's hand-written digits. We set the sample source as a binary protected attribute. The normal group contains digits from class 3, and the abnormal group includes digits from class 5.

- MNIST-Invert: We take the images from MNIST and create a duplicate which is inverted to build this dataset. The binary protected attribute is then original or inverted. The normal group contains digits from class 3 and the abnormal group contains digits from class 5.

5.5.2 Implementation

Due to the different characteristics of our selected data sets, we have implemented different networks for them. For the SVDD based encoder network $f(\theta)$: we use a convolutional neural network with two modules, 8 (5×5) filters followed by 4 (5×5) filters, and a final fully connected layer of 32 units for MNIST-USPS and MNIST-Invert data sets; we use a convolutional neural network with three modules, 32 $(5 \times 5 \times 3)$ filters, 64 $(5 \times 5 \times 3)$ filters, and 128 $(5 \times 5 \times 3)$ filters, followed by a fully connected layer of 128 units for

celebA data set; we use a fully connected neural network with two hidden layers with 32 and 16 units for the COMPAS Recidivism data set. We use batch normalization [78] and ReLU activations in these networks. Note for the fair deep SVDD model we have another classification branch $g(\theta_d)$. We employ a fully connected neural network with three hidden layers $(500 - 2000 - 500)$ as the sensitive attribute discriminator for all the data sets. We set the trade-off hyper-parameter λ default to 1 and the center c as the mean of all the instances embeddings. We set the learning rate as $1e^{-3}$ for Adam optimizer and conduct mini-batch training with batch size as 128. The weight decay hyper-parameter α is set to $5 * 10^{-6}$.

5.5.3 Evaluation Metrics and Baselines

In our experiments, we evaluate two aspects of the proposed approaches and the baseline methods. The first aspect is the ability to detect anomalies. We evaluate the anomaly detection performance using the common Area Under the ROC Curve (AUC). The AUC measure can be thought of as the probability that an anomalous example is given a higher anomaly score than a normal example. In this way, the higher the AUC score is better. The benefit of using AUC is because it represents the anomaly detection performance across various anomaly score thresholds t. The second aspect is the ability to be fair in terms of protected status variables. We use aforementioned $p\%$ -rule (equation 5.3) and distribution distance (equation 5.4) measures as our evaluation metrics.

We compare deep fair SVDD with two popular deep anomaly detection methods: deep SVDD [108] and deep convolutional auto-encoders (DCAE) [98]. We duplicate the deep fair SVDD's encoder network architecture for those two deep anomaly detection baselines to make a fair comparison. We use the default parameters suggested in their original papers.

5.5.4 The Unfairness of Deep Anomaly Detection

We first study the problem of whether existing deep anomaly detection methods can generate fair predictions. We study this under two settings one where we balance the PSV one where we do not. An imbalanced data set can very easily lead to unfair results whilst

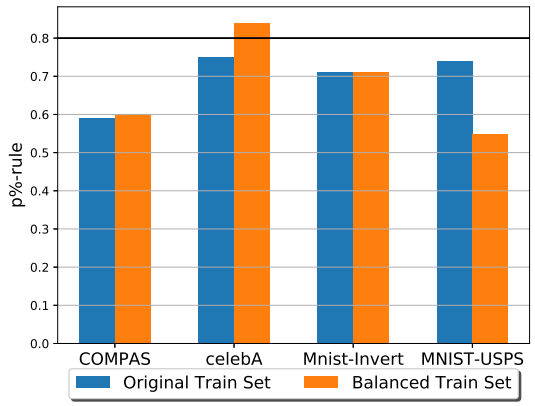
	COMPAS Recidivism	celebA	MNIST-Invert	MNIST-USPS
Original ($z = 0$)	1480	16000	6000	6131
Original ($z = 1$)	1210	4000	6000	658
Balanced ($z = 0$)	1210	4000	6000	658
Balanced ($z = 1$)	1210	4000	6000	658

Table 5.2: Characteristics of original training set and balanced training set used in experiments. We reduce the number of over-represented group in original training set to generate balanced training set.

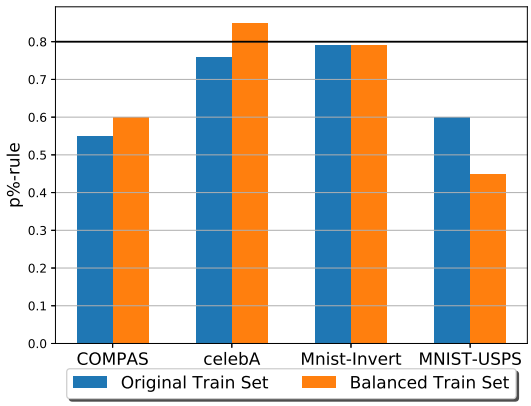
a balanced data set is easier to find fair anomalies. To demonstrate that deep anomaly detection models are unfair, we have prepared two versions of the training set: the original training set and the balanced training set. We have listed the detailed information in table 5.2. If the deep anomaly detection models can't generate fair predictions with both original and balanced training set, then we can conclude that our selected deep anomaly detection methods are unfair.

Thus, we conduct anomaly detection experiments and report both deep SVDD and DCAE's fairness performance on both versions of training sets in figure 5.4. We select these two methods because they represent the two popular types of deep anomaly detection methods. Observing Figure 5.4 (a) and (b), We can see for both COMPAS and celebA data set the deep SVDD and DCAE achieves higher fairness by $p\%$ -rule with a balanced training set. However, the improvements are not ideal because both approaches only satisfied the 80% rule on one data set (celebA). Moreover, for the MNIST-USPS data set, both deep SVDD and DCAE become more unfair with a balanced training set.

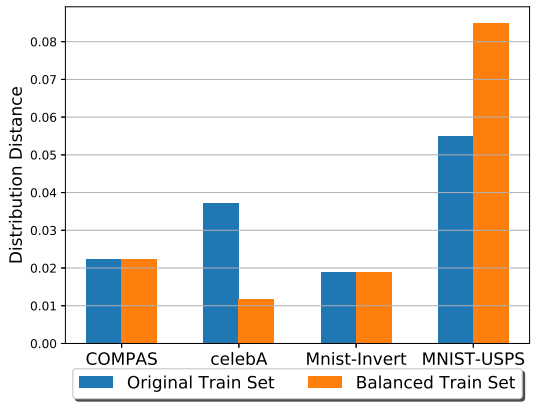
Figure 5.4 (c) and (d) shows the distribution distance which reflects the overall fairness of each model. The smaller distances indicate the model's predictions are more likely to be independent with the sensitive attribute. We can observe a similar trend as we have seen in Figure 5.5 (a) and (b) that learning on a balanced training set can only provide marginal improvements. We learn from these results that a fair anomaly detection approach is needed to mitigate deep anomaly detection algorithms' unfairness.



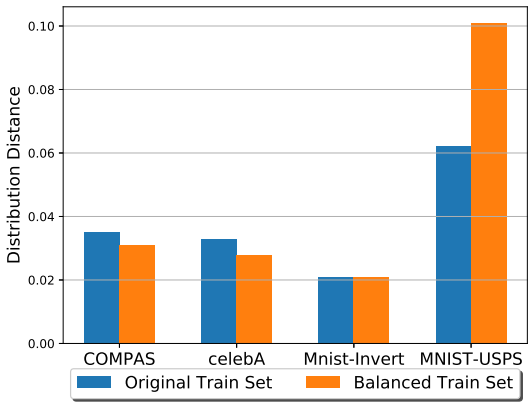
(a) Deep SVDD (p% -rule)



(b) DCAE (p% -rule)



(c) Deep SVDD (distribution distance)

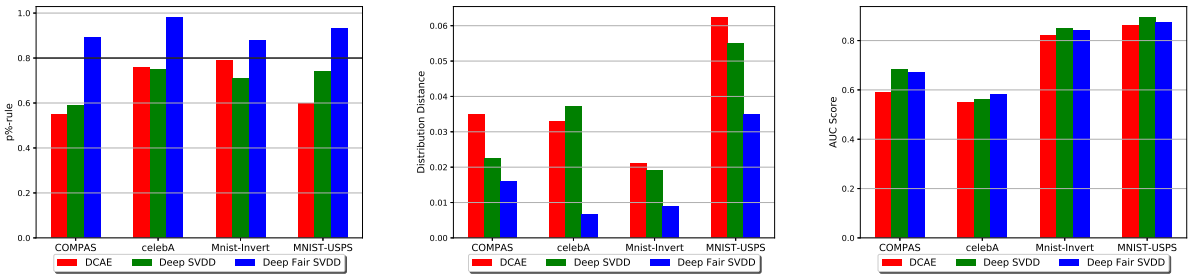


(d) DCAE (distribution distance)

Figure 5.4: Two methods of evaluating the unfairness for existing deep anomaly detection methods on both the original training sets (blue bars) and balanced training sets (orange bars). Note the larger fairness by p% -rule and smaller distribution distances means the model is fairer. Observed from these figures we can see that training deep anomaly detection models with a balanced training set can slightly improves the fairness in most cases. However, in most cases the fairness by p% -rule do not satisfy the 80% rule (black horizontal line) advocated by the US Equal Employment Opportunity Commission [18].

5.5.5 Evaluating Deep Fair SVDD

We now evaluate our proposed deep fair SVDD networks' performance and make a comparison with deep SVDD and DCAE. Figure 5.5 (a) shows the fairness by p% -rule on abnormal groups. We can see that deep fair SVDD outperforms both deep SVDD and DCAE in all four data sets. Moreover, deep fair SVDD's fairness by p% -rule are greater than 80% which satisfies the 80% rule [18] advocated by the US Equal Employment Op-



(a) Fairness by $p\%$ -rule

(b) Fairness by distribution distance

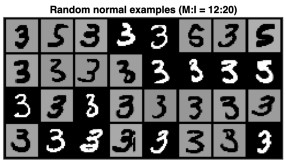
(c) AUC Scores

Figure 5.5: Comparison of deep fair SVDD with deep anomaly detection baseline methods on all four selected data sets. We evaluate the fairness performance for all the models trained on original data sets and plot the fairness by $p\%$ -rule and distribution distances in Figure (a), (b). We also evaluate the anomaly detection performance and show the AUC scores in Figure (c). Note deep fair SVDD achieves better fairness results with a slightly loss in terms of the AUC score.

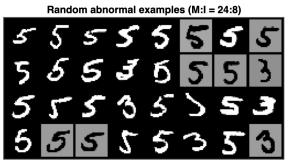
portunity Commission. The distribution distance results are shown in Figure 5.5 (b). We can see that deep fair SVDD achieves better overall fairness performance, especially for the celebA data set. Lastly, we show the test set AUC scores for four data sets in Figure 5.5 (c); we notice that in COMPAS, MNIST-Invert, and MNIST-USPS data sets, the deep SVDD performs slightly better than the other two approaches, while in the celebA data set the deep fair SVDD performs slightly better than other two approaches. Overall speaking, deep fair SVDD achieves much better fairness with a minimal loss in anomaly detection performance. Further, we analyze the interesting result on the celebA data set. In the celebA test set, both the normal and abnormal groups have a balanced number of males and females. Thus optimizing fairness in the celebA data set may also improve the anomaly detection performance. We have observed similar results in the following experiments on the trade-off analysis of deep fair SVDD (section 5.5.6).

Figures 5.6 shows examples of the random selected normal and anomalous examples according to deep SVDD and deep fair SVDD's predictions. For the MNIST-Invert data set, we can see that both the MNIST instances and Inverted MNIST instances are distributed evenly in the normal/abnormal groups determined by deep fair SVDD. On the contrary, there are more MNIST instances in the abnormal group and fewer MNIST in-

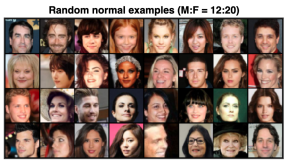
stances in the normal group determined by deep SVDD. As for the anomaly detection quality, both approaches have made few mistakes and achieved similar results, as shown in Figure 5.5.



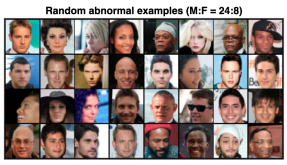
(a) Deep SVDD Results (MNIST-Invert)



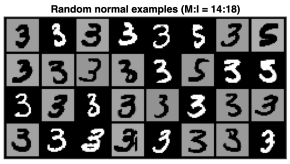
(b) Deep SVDD Results (MNIST-Invert)



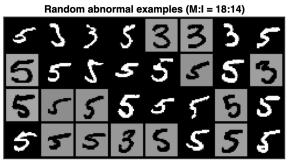
(c) Deep SVDD Results (celebA)



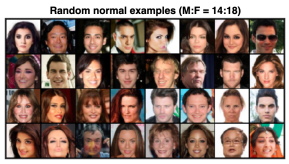
(d) Deep SVDD Results (celebA)



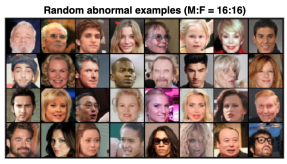
(e) Deep Fair SVDD Results (MNIST-Invert)



(f) Deep Fair SVDD Results (MNIST-Invert)



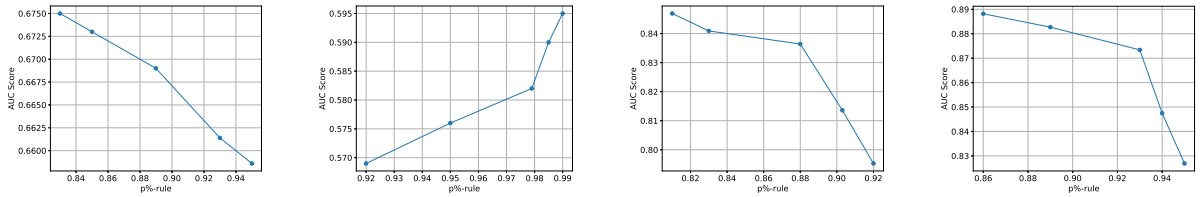
(g) Deep Fair SVDD Results (celebA)



(h) Deep Fair SVDD Results (celebA)

Figure 5.6: The visualization of the random selected normal and abnormal examples determined by deep SVDD (top row) and deep fair SVDD (bottom row) for MNIST-Invert data set and celebA data set. Comparing to the deep SVDD's prediction results, the size of instances with different protected status variable values are more balanced in fair SVDD's predictions.

The right-hand side of the Figure 5.6 shows the results for the celebA data set. Observing the deep SVDD's results on the top row shows that more males are predicted as plain faces and more females are predicted as attractive faces. These unfair results are mitigated with deep fair SVDD and we can see a nearly balanced number of males and females in both groups predicted via fair SVDD. As for the anomaly detection quality, both approaches made some mistakes and these are in line with the AUC scores we have reported in Figure 5.5 (c). This is reasonable as human faces contain far more information than digits. The anomaly detection tasks over human faces are more challenging than recognizing digits. Our main goal is to demonstrate how deep fair SVDD mitigates the unfair problems caused by deep anomaly detection baselines.



(a) COMPAS (b) celebA (c) MNIST-Invert (d) MNIST-USPS

Figure 5.7: The trade-off between fairness and anomaly detection performance. We tune the hyper-parameter λ to demonstrate the trade-off between fairness by $p\%$ -rule and anomaly detection performance in all the data sets. Note the λ ranges from 10^{-2} to 10^{2} and it is visualized in each plot with the order from left to right respectively. In all four datasets the fairness by $p\%$ -rule value increases as λ increases. The AUC scores decrease in most data sets as λ increases.

5.5.6 The Trade-off between Fairness and Anomaly Detection Performance

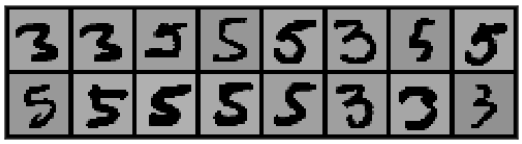
This section analyzes the trade-off between fairness performance and anomaly detection performance of deep fair SVDD. We re-train and test the deep fair SVDD under different values of hyper-parameter λ (range from 10^{-2} to 10^{2}) within equation (5.9). The hyper-parameter λ controls the weight of the discriminator's loss term within the adversarial loss function and directly determines the trade-off between the fairness performance and anomaly detection performance. Figure 5.7 shows the results: in all four selected data sets, the fairness by $p\%$ -rule increases as λ increases. The AUC score drops as the fairness by $p\%$ -rule value goes up for COMPAS, MNIST-Invert, and MNIST-USPS data sets. We have also noticed one different result in the celebA data set, both fairness by $p\%$ -rule and AUC score increase as the λ increases. We have analyzed this case before when comparing deep fair SVDD to deep anomaly detection baselines in plot 5.5 (c). Here fairness constraint is extra information that could help the algorithm improve anomaly detection performance. Generally speaking, training the deep fair SVDD with a larger λ will lead to fairer results and usually a slight loss in terms of the anomaly detection performance (AUC score).

	COMPAS	celebA	MNIST-Invert	MNIST-USPS
SVDD (Z_0 : Z_1)	198:336	854:1146	743:1041	186:137
Ours (Z_0 : Z_1)	263:271	980:1020	832:952	164:159
Overlap ratio	0.78	0.70	0.81	0.82

Table 5.3: Anomaly prediction results for deep SVDD and deep fair SVDD. Z_0 and Z_1 represent the number of predicted anomalies with protected status variable value as 0 and 1 respectively. There is a large overlap between these two model's anomaly predictions.

5.5.7 Anomaly Predictions Analysis

This section will conduct experiments to study how deep fair SVDD's predictions differ from deep SVDD's predictions. We have stored the anomaly prediction results for both approaches and summarized their overlapped anomaly predictions in Table 5.3. We use the number of overlapped anomaly predictions to divide the total number of anomalies as the overlap ratio. We can see that the overlap ratios are pretty high across all the data sets. We hypothesize the reason is that fair SVDD is also optimized with SVDD loss function. Furthermore, this high overlapping can also explain why fair SVDD only performs slightly worse than SVDD in terms of the AUC scores as we demonstrated in Figure 5.5 (c).



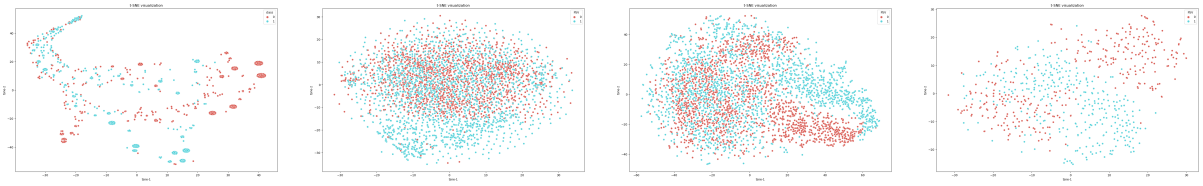
(a) Instances "moved" from normal to abnormal group

(b) Instances "moved" from abnormal to normal group

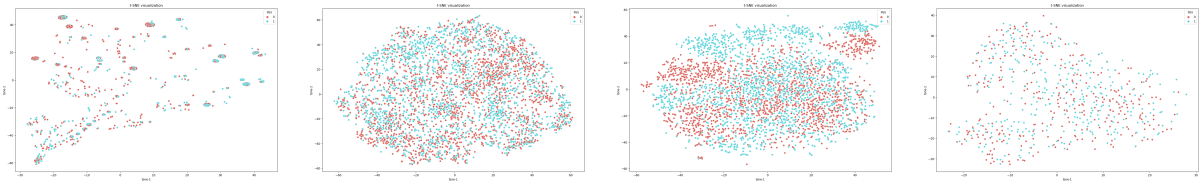
Figure 5.8: Illustration of how deep fair SVDD makes the anomaly detection results fairer. We visualize the sampled non-overlapping predictions between deep SVDD and deep fair SVDD. The instances in (a) can be seen as moved from deep SVDD's predicted normal group to deep fair SVDD's predicted abnormal group and vice versa for (b).

We also visualize the non-overlapping predictions between deep SVDD and deep fair SVDD in Figure 5.8. Take the MNIST-Invert data set for example; we randomly sample 16 non-overlapping anomalies with $z = 0$ from fair SVDD's predictions. We can view

these instances as moved from deep SVDD's predicted normal group to deep fair SVDD's predicted abnormal group to make the results fairer. Observing the digits from Figure 5.8 (a), we can see that deep fair SVDD is improving the fairness by moving instances that are "prone to be anomalies" to the abnormal group. One common feature of those instances is that they are dissimilar to a regular style of digit 3 and many of them are digits 5. It is important to show that these non-overlapping instances are not randomly distributed but are all prone to be anomalies. This interesting finding demonstrates that our proposed model is optimized to make fair and accurate anomaly predictions instead of random altering predictions to satisfy group-level fairness. We can observe the similar results from Figure 5.8 (b) that instances moved from deep SVDD's abnormal group to deep fair SVDD's normal group are "prone to be normal points."



(a) COMPAS (Non-fair) (b) celebA (Non-fair) (c) MNIST-Invert (Non-fair) (d) MNIST-USPS (Non-fair)



(e) COMPAS (Fair) (f) celebA (Fair) (g) MNIST-Invert (Fair) (h) MNIST-USPS (Fair)

Figure 5.9: The t-SNE [96] visualization of the feature embeddings for test instances. Red and blue points represent test instances with different sensitive attribute values. Comparing to deep SVDD's results (top row), the deep fair SVDD's learned embeddings (bottom row) are more fair as blue and red points are always blended together which are hard to separate.

5.5.8 Embedding Visualization

We visualize and compare the learned embeddings for both deep SVDD and deep fair SVDD to show why deep fair SVDD make fairer anomaly predictions. This analysis

is important as deep fair SVDD's objective is to learn a fair representation which is independent on the protected status variable z: $\mathbf{p}(f(\mathcal{X};\theta)|z=0) = \mathbf{p}(f(\mathcal{X};\theta)|z=1)$.

As shown in Figure 5.9, the red and blue points represent the test instances with the sensitive attribute value as $z = 0$ and $z = 1$ respectively. We first analyze the visualization results from deep SVDD; in each plot we can find some regions dominated by one particular color which indicates the correlation between feature embeddings and the protected status variable. On the contrary, observing from the deep fair SVDD's result we can see that the red and blue points are almost uniformly distributed in the feature space especially in the celebA data set. Deep fair SVDD is demonstrated to learn a fair representation that is independent of sensitive attributes.

	COMPAS	celebA	MNIST-Invert	MNIST-USPS
Deep SVDD	0.97	285.10	25.73	13.12
Ours	8.54	1703.49	167.53	231.78

Table 5.4: Training time results measured by seconds. Training deep fair SVDD takes longer time due to the min-max optimization of the adversarial learning.

5.5.9 Running Time Analysis

We have also reported the training time for deep fair SVDD and compared it against the deep SVDD approach in Table 5.4. Training deep fair SVDD takes longer time because we have a new fairness objective and it is learned through adversarial training. We leave how to speed up the training process as an interesting future work.

5.6 Conclusions and Future Work

This paper studied the fairness problem of deep anomaly detection methods and proposed a novel deep fair anomaly detection approach (deep fair SVDD). Deep fair SVDD is a method that uses deep neural networks to embed the data into a feature space where the normal data are closely clustered to the centroid. Adversarial training is used so that a discriminatory network cannot predict the protected status. Further, we propose two measures of the group-level fairness for deep anomaly detection problems. Given the

ground truth labels, we can directly measure the $p\%$ *-rule* (equation 5.3) for the abnormal group. We also propose *distribution distance* (equation 5.4), which can measure the overall fairness without knowing the labels of anomaly instances. We have conducted extensive empirical studies to evaluate the usefulness of our proposed approach. Firstly, our experiments show that deep anomaly detection methods will generate unfair predictions, even if the training data is balanced with respect to the binary protected state variables. Secondly, we evaluate our proposed deep fair SVDD and compare it to the deep anomaly detection baselines in various data sets. We demonstrate that our proposed work can achieve satisfying fairness results with minimal loss of anomaly detection performance. Next, we analyze the hyper-parameter λ which controls the trade-off between fairness and anomaly detection performance within our model and analyze the learned embeddings to study how our proposed model makes fair decisions.

In this paper, we limited ourselves to studying group-level fairness for deep anomaly detection problems with a single binary protected state variable. We leave for future works to study more complex fair anomaly detection problems such as considering multiple protected state variables, extending to semi-supervised anomaly detection settings (see section 5.4.3), and improving the training efficiency and scalability.

Chapter 6

Fair Learning for Deep Clustering

6.1 Introduction

Clustering is an essential data mining method which has been widely used in real-world applications involving humans [79] such as market research, social network analysis, and crime analysis. However, as intelligent tools augment and even replace humans in decision-making, the need to ensure clustering is fair becomes paramount. Here fairness is measured using protected status variables [1] (PSVs) such as gender, race, or education level.

Fairness takes two primary forms [20]: i) group-level and ii) individual-level. In this paper, we study the former which ensures that no one cluster contains a disproportionately small/large number of individuals with protected status compared to the population.

Recent works [107, 114, 87, 8, 13] have been proposed for non-deep fair clustering algorithms. To ensure group-level fairness, many of these works use the notion of the *disparate impact doctrine* encoded as a constraint, namely that instances from different protected groups must have approximately (within a tolerance) equal representation in a cluster compared to the population. Different geographic regions place this tolerance at different levels [35]. These existing algorithms optimize the clustering quality by minimizing some well-known clustering objectives while satisfying the group-level fairness constraints. Previous examples of adding fairness to clustering algorithms include k-median based approaches [35, 8, 13] and spectral clustering based algorithm [87]. However, all these works

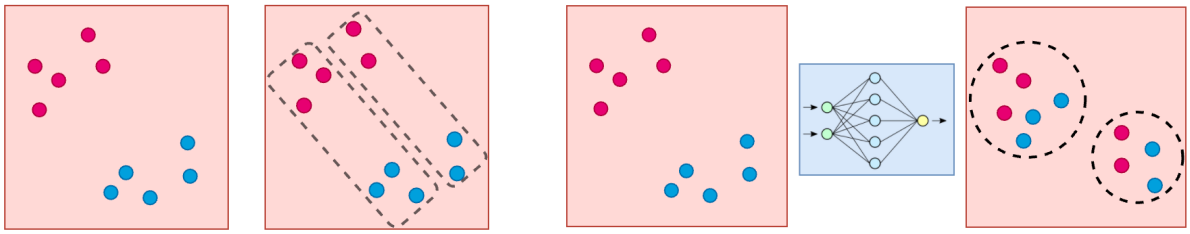
[1] In the paper, we use the term "protected status variable" and "sensitive attribute" interchangeably.

evaluate their performance on low-dimensional tabular data and [35, 87, 8] study the problems only with binary PSV.

Deep clustering [144, 76, 64, 138] has the ability to simultaneously cluster and learn a representation for problems with large amounts of complex data (i.e., images, texts, graphs). However, this creates the opportunity for the learner to become biased. For example, clustering of portraits may create clusters based on features which are surrogates for racial and other protected status information. One way to overcome this is by adding group-level fairness to deep clustering which is a challenging and understudied problem. A significant challenge is it is hard to translate the current fair clustering algorithms into an end-to-end deep clustering setting. For example, geometric pre-processing steps such as computing fairlets [35] to ensure fairness will not work as the end-to-end learning of deep learners means the underlying features that clustering is performed on are unknown apriori. Similarly, another line of work that adds constraints into deep learning models such as [147, 154] are not appropriate either as these constraints are at the *instance* level, whereas we require to apply fairness rules at a cluster level.

The work on fair deep clustering is relatively new. The first work on fair deep clustering [137] studies deep fair clustering problem from a geometric perspective which aims to learn a fair representation with multi-state PSV. The most recent work [92] proposes a deep fair visual clustering model with adversarial learning to encourage the clustering partition to be statistically independent of each sensitive attribute (PSV). Although these deep clustering approaches demonstrate better clustering performance compared to the non-deep fair clustering algorithms (Table 6.3), their fairness results are relatively poor compared to those fair clusterings with fairness guarantees [35, 8]. Our work can be seen as combining the benefits of deep learning and discrete optimization to produce *guaranteed fair predictions* on clustered data with PSVs while making *out-of-sample fair predictions* for data without PSVs.

In this paper, we propose a novel deep fair clustering framework and implement the Deep Fair Discriminative Clustering (DFDC) method to address the above issues. We adopt a probabilistic discriminative clustering network and learn a representation that



(a) Fair Non-Deep Clustering (b) Deep Fair Clustering

Figure 6.1: Note the red and blue points are instances with different PSV values. Fair Non-Deep Clustering (left) aims to find a fair partition of the data while minimizing some classic clustering objectives. Deep Fair Clustering (right) aims to learn a general fair representation and to simultaneously cluster the data.

naturally yields compact clusters. To incorporate the group-level fairness rules in the deep learner, we first formulate our fairness objective as an integer linear programming (ILP) problem that guarantees group-level fairness. This ILP takes an existing clustering and makes it fair. We show this ILP is efficient to solve as its constraint matrix is totally unimodular. The output of the ILP is then used as a fairness signal to learn from. Besides DFDC, our proposed general fair learning algorithm is applicable for different deep clustering backbones which we demonstrate on Deep Embedded Clustering (DEC)[144]. The major contributions of this paper are summarized as follows:

- We address the newly-emerging deep fair clustering problem and propose a novel fair learning framework for deep clustering. We propose DFDC as one representative example of our proposed fair deep clustering algorithms.

- DFDC optimizes a general notion of fairness for multi-state PSVs which we prove (see Theorem 6.3.2) is equivalent to optimizing the balance measure [35] for *disparate impact*.

- We propose a novel two-step approach that: i) uses an efficiently solvable (see Theorem 6.4.2 proof of total unimodularity) ILP problem to generate a fairer variant of the current clustering the DL produces and ii) uses an end-to-end refinement learning algorithm to learn from the ILP's solution (See Algorithm 5). This fair learning algorithm can be applied to different clustering backbones which have fractional

clustering assignments as outputs.

- Extensive experimental results show that our proposed DFDC can achieve guaranteed fairness with competitive clustering performance (See Figure 6.2 and Table 6.3).

- We demonstrate novel extensions for new fair clustering tasks such as predictive (out of sample) clustering, multi-state PSVs and flexible fairness rules. (See Section 6.5.2).

In the next section 6.2 we discuss the related work. Then we outline our measure of fairness and how it relates to classic measures of disparate impact in section 6.3. In our approach section 6.4, we introduce our clustering framework and encode our fairness objective as an ILP which can be efficiently solved via our relaxation. A refinement learning algorithm is proposed for end-to-end fair clustering. Finally we empirically evaluate the effectiveness of our approach in section 6.5 and conclude in section 6.6.

6.2 Related Work

Fair Non-Deep Clustering. Fair clustering has received much attention recently [114, 86, 2, 34, 41, 97, 25]. [35] first addressed the disparate impact for clustering problems in the presence of binary PSVs. Their work apriori groups instances into many fairlets which are used as input into standard k-medians style algorithms. Their work is guaranteed to produce a specified level of fairness and achieve a constant factor approximation with respect to cluster quality. We shall see (Theorem 6.3.2) that the balance criteria proposed here is effectively the same criteria we use in DL. [8] improves the fair decomposition algorithm to linear run-time. Later on, [13] propose a general fair clustering algorithm that allows human-specified upper and lower bounds on any protected group in any cluster. Their work can be applied to any clustering problems under ℓ_p norms such as k-median, k-means, and k-center. Besides the centroid-based method, [87] extends the fairness notion to graph spectral clustering problems. [164] propose a general, variational and bound-optimization framework of fair clustering.

Deep Clustering. Previous fair clustering approaches mainly focus on adding fairness constraints into fair non-deep clustering algorithms. In our work, we aim to study the fairness problem for recently proposed deep clustering algorithms [144, 149, 76, 26, 120, 129, 119] by adding a fairness signal to learn from. Deep clustering algorithms connect representation learning and clustering together and have demonstrated their advantages over the two-phase clustering algorithms which use feature transformation first and then clustering. Motivated by the success of deep clustering, the goal of deep fair clustering is to learn a fair and clustering-favored representation. We illustrate the basic intuitions behind fair non-deep clustering methods and deep fair clustering approaches in Figure 6.1. One of the biggest challenges for deep fair clustering is to ensure fairness.

Deep Fair Clustering. One of the earliest works [137] to address the deep fair clustering problem learns a latent representation such that the cluster centroids are equidistant from every "fairoid" (the centroid of all the data belonging to the same protected group). Recently, [92] encodes the fairness constraints as an adversarial loss and concatenates the fairness loss to a centroid-based deep clustering objective as a unified model. Unlike previous deep fair clustering works, we translate the fairness requirements into an ILP problem that generates guaranteed fair solutions given the PSVs. Our deep clustering architecture supports flexible fairness constraints and multi-state PSVs. Moreover, we propose a novel learning framework to train fair clustering models via simultaneous clustering and fitting the self-generated fairness signals.

6.3 Definitions of Group-level Fairness

We begin this section by overviewing the seminal definition of group-level fairness [35] in clustering (see equation 6.1) and then its extension to multi-state PSVs (see equation 6.2). We then go onto show a new measure that our deep clustering framework will optimize (see equation 6.3) and equation 6.2 have the same optimal condition as shown in Theorem 6.3.2.

6.3.1 Notion of Fairness

Let $X \in \mathbb{R}^{N \times D}$ denote N data points with D dimension features. The prediction function ϕ assigns each instance to one unique cluster, $\phi : x \to \{1 \dots K\}$, which forms K disjoint clusters $\{C_1 \dots C_K\}$. Given the protected status variable (denoted as PSV) with T states, X can be partitioned into T demographic groups as $\{G_1, G_2, ...G_T\}$.

Definition 6.3.1. *The seminal proposed measure of fairness for clustering with binary PSV [35] encoded disparate impact as the balance of a cluster as follows:*

$$balance(C_k) = \min\left(\frac{N_k^1}{N_k^2}, \frac{N_k^2}{N_k^1}\right) \in [0, 1] \tag{6.1}$$

Here N_k^1 and N_k^2 represent the populations of the first and second demographic groups in cluster C_k. The clustering model will be fairer with larger balance value. Such a measure of fairness only works for binary PSV. To allow for multi-state PSVs we redefine balance as follows, let $N_k^{min} = \min(N_k^1 \dots N_k^T)$ denotes the smallest (in size) protected group in cluster k and $N_k^{max} = \max(N_k^1 \dots N_k^T)$ denotes the largest group. We can then extend the balance measure for multi-state PSV as:

$$balance(C_k) = \frac{N_k^{min}}{N_k^{max}} \in [0, 1] \tag{6.2}$$

We will show that recently proposed works [107, 13] fairness measures for multi-state PSVs (equation 6.3) are equivalent to our definition in equation 6.2.

Definition 6.3.2. *Let ρ_i be the representation of group G_i in the dataset as $\rho_i = |G_i|/N$, and $\rho_i(k)$ be the representation of group G_i in the cluster C_k: $\rho_i(k) = |C_k \cap G_i|/|C_k|$. Using these two values, the fairness value for cluster C_k is:*

$$fairness(C_k) = \min\left(\frac{\rho_i}{\rho_i(k)}, \frac{\rho_i(k)}{\rho_i}\right) \in [0, 1] \quad \forall i \in \{1, \dots T\} \tag{6.3}$$

The overall fairness of a clustering is defined as the *minimum* fairness value over all the clusters. Similarly, the overall balance is the *minimum* balance value of all the clusters.

6.3.2 Equivalence of Optimizing Fairness and Balance Measures

Here we show that optimizing equation 6.3 is equivalent to optimizing our extended definition of balance in equation 6.2. We see that equation 6.3 achieves maximal fairness when

$P(x \in G_t | x \in C_k) = \rho_t$. Our balance measure in equation 6.2 achieves optimal balance when $P(x \in G_t | x \in C_k) = \frac{1}{T}$ for any protected group G_t in cluster C_k. However, this is an ideal case as protected groups may be imbalanced. Denote the size of each protected group as $|G_i|$ and the size of the data set as N, we now show that the optimal balance is achieved if and only if $P(x \in G_t | x \in C_k) = \rho_t$. This result indicates the equivalence of optimizing fairness (equation 6.3) and generalized balance (equation 6.2).

Lemma 6.3.1. *The optimal balance can be achieved only when all the clusters have the same balance. Formally,* $\forall i, j \in \{1, 2, ..., K\}$*:* $balance(C_i) = balance(C_j)$.

Proof. The proof is by contradiction, we assume the optimal balance can be achieved when not all clusters have the same balance. Let cluster C_i have the largest balance as b_{max} and cluster C_j has the smallest balance as b_{min}. Denote the number of T different groups' instances in C_i as $\{C_{i1}, C_{i2}, \ldots, C_{iT}\}$, similarly the composition for C_j as $\{C_{j1}, C_{j2}, \ldots, C_{jT}\}$. Assume the balance for C_i is achieved by $b_{max} = \frac{C_{i\alpha}}{C_{i\beta}}$, the balance for C_j is achieved by $b_{min} = \frac{C_{j\gamma}}{C_{j\theta}}$. Now we discuss two possible cases:

Case 1: If $\alpha = \gamma$ and $\beta = \theta$. Based on our definition we have $\frac{C_{i\alpha}}{C_{i\beta}} > \frac{C_{j\alpha}}{C_{j\beta}}$. Let $r = \frac{1}{2}(\frac{C_{i\alpha}}{C_{i\beta}} + \frac{C_{j\alpha}}{C_{j\beta}})$, we can move ϵ instances which belong to group α from C_i to C_j to achieve the higher balance r. This can be done by setting $\frac{C_{i\alpha}-\epsilon}{C_{i\beta}} = \frac{C_{j\alpha}+\epsilon}{C_{j\beta}} = r$.

Case 2: If $\alpha \neq \gamma$ or $\beta \neq \theta$. Based on the balance definition we have $\frac{C_{i\gamma}}{C_{i\theta}} \geq \frac{C_{i\alpha}}{C_{i\beta}}$. Based on our definition we have $\frac{C_{i\alpha}}{C_{i\beta}} > \frac{C_{j\gamma}}{C_{j\theta}}$. Thus we have $\frac{C_{i\gamma}}{C_{i\theta}} \geq \frac{C_{i\alpha}}{C_{i\beta}} > \frac{C_{j\gamma}}{C_{j\theta}}$. Similar as in case 1 we can move ϵ' instances which belong to group γ from C_i to C_j to achieve higher balance $r' = \frac{1}{2}(\frac{C_{i\gamma}}{C_{i\theta}} + \frac{C_{j\gamma}}{C_{j\theta}})$.

In both cases, we can swap some instances between clusters to increase the final balance. This contradicts our assumption and completes our proof. □

Theorem 6.3.2. *To achieve optimal balance value for multi-state protected variables, we must satisfy the condition:* $P(x \in G_t | x \in C_k) = \rho_t$ *which is precisely the optimal fairness value for equation 6.2.*

Proof. Given the condition $P(x \in G_t | x \in C_k) = \rho_t$, let G_{min} be the smallest protected group and G_{max} be the largest protected group, the largest balance we can achieve

is $|G_{min}|/|G_{max}|$. The proof by contradiction assumes these exists a solution where all the clusters have the same balance (based on lemma 6.3.1) and have balance $\alpha > |G_{min}|/|G_{max}|$. For each cluster we sort each protected group based on their size in an increasing order. We use $C^*_{i1}, C^*_{i2} \dots C^*_{iT}$ to denote the size of the sorted groups in cluster i. Obviously we have $\alpha = \frac{C^*_{i1}}{C^*_{iT}}$ for any cluster i. Now we sum up the smallest group among all the K clusters as S: $S = \sum_{i=1}^{K} C^*_{i1}$. Similarly we can calculate the sum of the largest group among all the clusters as: $\sum_{i=1}^{K} C^*_{iT} = \frac{1}{\alpha}\sum_{i=1}^{K} C^*_{i1} = \frac{S}{\alpha}$. Consider the allocation of the smallest group G_{min} we have the following inequality: $|G_{min}| \geq \sum_{i=1}^{K} C^*_{i1} = S$. Similarly we have $|G_{max}| \leq \sum_{i=1}^{K} C^*_{iT} = \frac{S}{\alpha}$. By combining previous two inequalities we have $\alpha|G_{max}| \leq S \leq |G_{min}|$ which means $|G_{min}|/|G_{max}| \geq \alpha$. This contradicts with our initial assumption that there exists a α which is larger than $|G_{min}|/|G_{max}|$. Thus we complete the proof. □

6.4 Deep Fair Clustering Algorithm

We introduce our fair learning framework in this section. Our proposed DFDC approach can be viewed as learning fair clustering under a discriminative clustering loss objective (described in section 6.4.1) and a fairness objective with self-generated signals. However, section 6.4.4 shows the versatility of our fair learning framework by using it for other deep clustering backbones. Our method can be used for any deep clustering method that produces a fractional cluster allocation vector for each instance. Section 6.4.2 describes our ILP formulation to take an existing clustering and make it fairer, whilst Section 6.4.3 shows how to use the ILP output as a fairness signal to learn from.

6.4.1 Review of Base Clustering Model

For base clustering model, we show our method applied to previous work [76] which we overview here. We learn a neural network f_θ as a discriminative function to predict the clustering assignments $Y = \sigma(f_\theta(X)) \in \mathbb{R}^{N\times K}$ based on input $X \in \mathbb{R}^{N\times D}$ and softmax function σ. The mutual information $I(X;Y)$ between X and Y is calculated as the

difference between marginal entropy $H(Y)$ and conditional entropy $H(Y|X)$:

$$\begin{aligned} I(X;Y) =& H(Y) - H(Y|X) \\ =& h(\frac{1}{N}\sum_{i=1}^{N}\sigma(f_\theta(x_i))) - \frac{1}{N}\sum_{i=1}^{N}h(\sigma(f_\theta(x_i))) \end{aligned} \tag{6.4}$$

where h is the entropy function. With weight decay term the clustering objective ℓ_C is as follows:

$$\ell_C = \frac{1}{N}\sum_{i=1}^{N}h(\sigma(f_\theta(x_i))) - h(\frac{1}{N}\sum_{i=1}^{N}\sigma(f_\theta(x_i))) + \alpha\sum_{l=1}^{L}\|\theta^l\|^2 \tag{6.5}$$

where α denotes the hyper-parameter for network parameters $\{\theta^1 \dots \theta^L\}$. Maximizing $H(Y)$ will punish imbalanced cluster size and prevent trivial solutions where all the instances are clustered into one cluster while minimizing $H(Y|X)$ will map similar instances x to have similar labels y.

Further, self-augmented training is applied to encourage the representations to be locally invariant. Here a local perturbation of instance x is added such that $x^{'} = x + t$, perturbation t is maximized subjecting to the constraint that the clustering assignments for x and $x^{'}$ are the same. Virtual adversarial training [99] is applied to generate adversarial direction for t. Denote the current model's parameters θ to help estimate the true clustering indicator vector for instance x as $\sigma(f_\theta(x))$, the formulation to compute the adversarial perturbation t_{adv} is as follows:

$$t_{adv} = \underset{t;||t||_2\leq\epsilon}{\operatorname{argmax}}\ \mathrm{KL}(\sigma(f_\theta(x)), \sigma(f_\theta(x+t))) \tag{6.6}$$

With the generated t_{adv}, the augmentation loss ℓ_{Aug} minimizes the KL divergence between clustering assignment $\sigma(f_\theta(x_i))$ and its augmented version's assignment $\sigma(f_\theta(x_i^{'}))$:

$$\ell_{Aug} = \sum_{i=1}^{N}\mathrm{KL}(\sigma(f_\theta(x_i)), \sigma(f_\theta(x_i^{'}))) \tag{6.7}$$

Finally, the base clustering model optimizes the clustering loss ℓ_C and ℓ_{Aug} simultaneously. Note that we favor this probabilistic discriminative clustering model [76] since it has fewer assumptions about the natures of categories that are made and fits our fairness objective which requires fractional clustering assignments as inputs to indicate the degree of cluster assignment belief.

6.4.2 Generating Fair Assignments Under Group-level Fairness Constraints

Let the clustering assignments from the current learned model be $Y = \{y_1, \dots y_N\} \in \mathbb{R}^{N \times K}$. We wish to minimally modify the assignments but make the clustering fairer. To achieve this we solve for a matrix $\hat{Y} = \{\hat{y}_1, \dots \hat{y}_N\} \in \mathbb{Z}^{N \times K}$ that satisfy the fair optimal condition $P(x \in G_t | x \in C_k) = \rho_t$ whilst minimizing the changes to the clustering as below:

$$\text{Objective:} \quad \arg\min_{\hat{Y}} \sum_{i=1}^{N} [1 - y_i \times \hat{y_i}^T] \tag{6.8}$$

Recall y_i is a row vector which represents the probability distribution over the cluster assignments for instance i and $\hat{y}_i$ chooses exactly one cluster to assign instance i to. Naturally the objective is maximized when y_i is assigned to its most probable cluster but this may cause an unfair clustering. To ensure fairness we use the following constraint.

We denote $\rho_i = |G_i|/N$ as the fraction of the protected group G_i in the data set and our aim is for each cluster to have approximately the same density. Let $M \in \mathbb{Z}^{N \times T}$ encode the sensitive attributes for the entire population such that $M_{it} \in \{0, 1\}$ indicates whether an instance x_i belongs to a protected group G_t. To satisfy optimal fair condition $P(x \in G_t | x \in C_k) = \rho_t$ we have the following constraints:

$$\sum_{i=1}^{N} M_{it} \hat{y}_{ij} = \sum_{i=1}^{N} \hat{y}_{ij} \rho_t \quad \forall j \in \{1 \dots K\}, t \in \{1 \dots T\} \tag{6.9}$$

Now we relax the problem by fixing the size of each new cluster to make the constraint matrix totally unimodular. We round the soft probabilistic assignment Y as hard assignments $Y^{'}$ by assigning the cluster with largest probability. Then the size of cluster C_j is $|C_j| = \sum_{i=1}^{N} y^{'}_{ij}$. The constraints for new clusters' size are:

$$\sum_{i=1}^{N} \hat{y}_{ij} = |C_j| \quad \forall j \in \{1 \dots K\} \tag{6.10}$$

Lastly we add constraints for $\hat{Y}$ to ensure each instance is assigned to one cluster:

$$\sum_{j=1}^{K} \hat{y}_{ij} = 1 \quad \forall i \in \{1 \dots N\} \tag{6.11}$$

Note this ILP formulation also supports user-defined ρ_t which can be seen as a flexible fairness rule. Next we show the constraint matrix of our ILP problem is totally unimodular so that we can efficiently solve it with a LP solver and still return integral solutions.

We know that if a constraint matrix of an ILP is totally unimodular (TU) then we can solve the problem using an LP (linear program) solver and the solution will still be integral [116]. Using an LP solver will largely reduce the running time and [130] has shown that the running time for LP is polynomial in the input size. In the above proposed constraints, there are NK unique regular variables (N instances and K categories). To construct the constraint matrix C which encodes constraint 6.9, 6.10 and 6.11, we will use NK regular variables. Matrix C has $T+1$ rows (the first T rows correspond to the fairness constraints in equation 6.9 and last row corresponds to constraints in equation 6.11) and $N+K$ columns. Note the first K columns of the last row are set to 0 and the last N columns of first T rows are set to 0. In matrix C, each entry of C is from $\{-1, 0, 1\}$. Moreover, each column only has one non-zero element. This is because: (1) for constraints set in equation 6.9, each instance only belongs to one protected group, (2) for constraints set in equation 6.11, there is only one row vector with K elements as 1 to ensure the valid assignment.

Lemma 6.4.1. ***TU Identity*** *[116]. Let C be a matrix such that all its entries are from $\{0, 1, -1\}$. Then C is totally unimodular, i.e., each square submatrix of C has determinant 0, 1, or -1 if every subset of rows of C can be split into two parts A and B so that the sum of the rows in A minus the sum of the rows in B produces a vector all of whose entries are from $\{0, 1, -1\}$.*

Theorem 6.4.2. *The matrix C formed by the coefficients of the constraints used to encode our proposed constraints from equation 6.9, 6.10 and equation 6.11 is totally unimodular.*

Proof. We consider any subset F of rows in C. We will show that F can be partitioned into two sets A and B to satisfy the condition in lemma 6.4.1. Our partitioning scheme is as follows: the first row of F is put into A and the remaining rows are put into B. Let row vectors SA and SB denote the sums of the rows within A and B respectively. It is

clear that elements in SA and SB are from $\{1, 0, -1\}$ because each column only has one non-zero element. Now we show all the elements in $SA - SB$ are from $\{1, 0, -1\}$. Firstly there will be only one non-zero element in SA, and we denote the column which has a non-zero element as r. If column r belongs to the fairness constraints (from column 1 to column N in C), then the column r within SB will be 0 since one instance only belongs to one protected group. If the column r belongs to the valid assignment constraints (from column $N+1$ to $N+K$ in C) then column r within SB will be 0 since those elements are filled with 0. Thus column r of $SA - SB$ is still from $\{1, 0, -1\}$. The non-zero elements in other columns will change their sign but still from $\{1, 0, -1\}$. This completes the proof. □

6.4.3 Learning to Be Fairer

To learn a fair clustering model we aim to exploit the fairness assignments $\hat{Y}$ to reshape the features learned via clustering networks f_θ. We treat $\hat{Y}$ as "pseudo-labels" to optimize the following cross entropy loss ℓ_{Fair} for fairer results:

$$\ell_{Fair} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\hat{y}_{ij}logy_{ij} = \frac{1}{N}\sum_{i=1}^{N}\hat{y}_i log(\sigma(f_\theta(x_i))) \tag{6.12}$$

Simply optimizing the fairness loss ℓ_{Fair} will dramatically change the current clustering representations to fit an approximated fair assignment $\hat{Y}$ which harms the clustering properties. Instead, we propose to learn a fairer and clustering-friendly representation simultaneously by combining the clustering loss ℓ_C, augmentation loss ℓ_{Aug} and fairness loss ℓ_{Fair}. Note the fair assignments $\hat{Y}$ are updated after each training epoch as the "nearest" fair assignments for current clustering predictions.

To train our proposed framework, we start with the training on base clustering network $f(\theta)$ via optimizing the clustering loss ℓ_C and augmentation loss ℓ_{Aug} to ensure the data are separated into different meaningful clusters; as clustering model converges we generate fair assignments after each training epoch based on the objective in equation 6.8 and optimize the overall loss function ℓ by concatenating the fairness loss ℓ_{Fair} to clustering objectives as $\ell = \ell_C + \beta\ell_{Fair} + \gamma\ell_{Aug}$ where β, γ are positive weight hyper-parameters. Algorithm 5 summarizes the proposed learning method.

Algorithm 5 Main learning algorithm for deep fair discriminative clustering.

Input: Input $\{x_k\}_{k=1}^N$, sensitive attributes M, cluster size K, clustering network f_θ, hyper-parameters α, β, γ.

Output: Clustering network f_θ, predictions $\{y_k\}_{k=1}^N$.

for each pre-trained epoch **do**
  for sampled mini-batch $\{x_k\}_{k=1}^n$ **do**
    Calculate $\ell_C = \frac{1}{n}\sum_{i=1}^n h(\sigma(f_\theta(x_i))) - h(\frac{1}{n}\sum_{i=1}^n \sigma(f_\theta(x_i))) + \alpha \sum_{l=1}^L \|\theta^l\|^2$
    Generate $x_k^{'} = x_k + t$ via solving t from eq 6.6.
    Calculate $\ell_{Aug} = \sum_{i=1}^n \mathrm{KL}(\sigma(f_\theta(x_i)), \sigma(f_\theta(x_i^{'})))$
    Update network f_θ via minimizing $\ell_C + \gamma \ell_{Aug}$.
  end for
end for
repeat
  Generate predictions $\{y_k\}_{k=1}^N$ based on f_θ.
  Construct a fair assignment problem via objective 6.8 and constraints defined in eq 6.9, 6.10 and 6.11.
  Solve fair assignments $\{\hat{y}_k\}_{k=1}^N$ via LP solver.
  for sampled mini-batch $\{x_k\}_{k=1}^n$ **do**
    Calculate $\ell_{Fair} = \frac{1}{n}\sum_{i=1}^n \hat{y}_i log(\sigma(f_\theta(x_i)))$
    Calculate $\ell_C = \frac{1}{n}\sum_{i=1}^n h(\sigma(f_\theta(x_i))) - h(\frac{1}{n}\sum_{i=1}^n \sigma(f_\theta(x_i))) + \alpha \sum_{l=1}^L \|\theta^l\|^2$
    Generate $x_k^{'} = x_k + t$ via solving t from eq 6.6.
    Calculate $\ell_{Aug} = \sum_{i=1}^n \mathrm{KL}(\sigma(f_\theta(x_i)), \sigma(f_\theta(x_i^{'})))$
    Calculate $\ell = \ell_C + \beta \ell_{Fair} + \gamma \ell_{Aug}$
    Update network f_θ via minimizing ℓ.
  end for
until $\{y_k\}_{k=1}^N$ satisfy *optimal fairness* rules

6.4.4 Can our Proposed Fairness Module Work for Other Clustering Algorithms?

One of the major contributions in our proposed algorithm is the use of "fair pseudo-label" as signal to learn fair clustering. In this subsection, we will show that our fairness module can be implemented on other deep clustering algorithms so long as they output cluster

allocation. Based on algorithm 5, the whole model is first trained on a base clustering algorithm and then fine-tuned with both clustering and fairness objectives. To demonstrate that our proposed fairness module works for other deep clustering frameworks, we choose the Deep Embedded Clustering (DEC) as our alternative clustering backbone. Note DEC is one of the most representative deep clustering algorithms which provides fractional clustering assignments as outputs. We replace the discriminative clustering model with DEC in our fair learning framework and denote it as Fair-DEC. Next we introduce our experimental section which demonstrates the success of our proposed fair learning module.

6.5 Experiments

We conduct experiments[2] to evaluate our approach empirically and report the following key results:

- Our proposed approach achieves better clustering performance and guaranteed fairness results compared against both fair non-deep clustering and deep fair clustering baselines (See Table 6.3 and Figure 6.2).
- Our proposed approach is effective in novel fair clustering settings such as supporting flexible fairness constraints, clustering with multi-state PSVs, and predictive (out of sample) clustering (See Figure 6.3 and 6.4).
- We show how our learned embedding converges to a latent space useful for fair clustering (See Figure 6.5) quickly and also provides insights on tuning hyper-parameter β (See Figure 6.6) in unsupervised way to achieve our fairness goal with a minimum loss on clustering performance.

6.5.1 Experimental Setup

First we describe our data sets, then our evaluation scheme and finally the implementation for reproduction. We implement our framework in PyTorch and *here is the link* for our code and data.

[2]In our experimental work we use the fair clustering data sets used by earlier work for comparison.

Datasets used for deep fair clustering. We first evaluate our work on two visual datasets with binary PSV that has been used in recent deep fair clustering work [92] and then experiment on a telemetry deep clustering dataset with multi-state PSVs:

- MNIST-USPS consists of 67291 training images of hand-written digits. We use the image source (MNIST or USPS) as a binary PSV and cluster the data into 10 classes representing 10 digits.
- Reverse-MNIST takes the 60000 training images from MNIST and creates an inverted duplicate to build this dataset. The binary PSV is then original or inverted and the total number of classes is 10.
- A challenging fair clustering task with multi-state PSV is the Human Activity Recognition (HAR) dataset used in [137]. The HAR dataset [3]

Datasets used for non-deep fair clustering. We use three tabular datasets common for evaluation in non-deep fair clustering[13]:

- Census [4] with 5 attributes ("age", "fnlwgt", 'education-num", "capitalgain", "hours-per-week") and binary PSV gender, we set whether income exceeds 50K as the clustering label.
- Bank [5] data with 3 attributes ("age", "balance", "duration-of-account") and binary PSV "marital", we set whether a client will subscribe a term deposit as the label.
- Credit [6] with 14 features and PSV "marital", we set whether the cardholder will make a payment as label. The characteristics of all six datasets are summarized in Table 6.1.

Evaluation Metrics and Baselines. To measure the clustering quality for deep fair clustering and other baselines, we use both clustering accuracy (ACC) [148, 150] and

[3] https://archive.ics.uci.edu/ml/datasets/HAR contains 10299 instances in total with captured action features for 30 participants. There are 6 actions in total which serve as labels for clustering. The identity of each person is used as the PSV value.
[4] http://archive.ics.uci.edu/ml/datasets/Census+Income
[5] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
[6] https://archive.ics.uci.edu/ml/datasets/credit+card+clients

Dataset	Instances	Features	Domain	PSV
Census	32561	5	Income	Gender
Bank	4521	14	Credit	Marital
Credit	30000	3	Credit	Marital
MNIST-USPS	67291	784	Images	Image Source
Reverse-MNIST	120000	784	Images	Image Source
HAR	10299	561	Sensors	Identity

Table 6.1: Characteristics of datasets

normalized mutual information (NMI) metrics. To evaluate the fairness, we use the balance measure defined in equation 6.2. For all those three measures, higher values indicate better performance. For the deep clustering baselines, we use DEC [144] as a representative method for centroid-based clustering and IMSAT [76] for discriminative clustering approach. For fair clustering algorithms, we choose the scalable fair clustering algorithm [8] and the fair algorithms for clustering [13]. For deep fair clustering baselines, we compare our work with the latest work [92] and the geometric-based fair clustering [137]. As for our own approach, we report both ILP module's final output and the deep neural networks' final output. The final output means the prediction results after the last training epoch. By comparing the ILP's results with deep learner's output, we aim to test whether our model has learned the notion of fairness. Using the ILP results guarantees a fair result. Besides, we also report the Fair-DEC's results to show how general is our proposed fair learning module.

Implementation. In this section we introduce our network architectures with the selected hyper-parameters for reproduction. For the digit datasets like reverse-MNIST and MNIST-USPS, we use a 2-block convolutional architecture with first block consisting of a convolutional layer with 8 (5×5) filters and second block consisting of 4 (5×5) filters. Each block is followed with BatchNorm, ReLU activation and max pooling layers. The second block is finally connected to a fully connected layer with 10 units. For the HAR data we use a stacked fully connected neural network with intermediate layers as

	η	n	Epochs	β	γ	α
HAR	0.0002	256	60	4.00	1.00	0.0001
reverse-MNIST	0.0005	256	80	6.00	1.00	0.0001
MNIST-USPS	0.0005	256	60	4.00	1.00	0.0001
Census	0.0002	256	30	4.00	1.00	0.0001
Credit	0.0002	256	50	5.00	1.00	0.0001
Bank	0.0005	256	30	4.00	1.00	0.0001

Table 6.2: Hyperparameters used in our experiments: n denotes the training batch size, η represents the learning rate, α is the weight-decay parameter and β, γ are the hyper-parameters for fairness module and self-augmented training branch.

$d-1200-1200-k$, note $d = 561$ is the total number of features in HAR and $k = 6$ is the number of clusters. For the non-deep clustering data, we use a stacked fully connected neural network with intermediate layers as $d - 50 - 50 - k$, note the total number of features d for Census, Credit and Bank are $5, 3, 14$ and $k = 2$ is the number of clusters. In unsupervised learning, it is not straightforward to determine hyper-parameters by cross-validation. Hence, we fixed the hyper-parameters (weight decay α and augmentation term γ) across all the datasets; we tuned the hyper-parameter β based on the training set's balance value as mentioned in section 6.5.3 to achieve satisfying fair results. Finally, we have summarized our used hyper-parameters in Table 6.2.

Results on High Dimensional Data. As shown in the Table 6.3, fair non-deep clustering algorithms achieve good fairness results especially ScFC which returns guaranteed fair clusters. However the clustering performance is not good as deep clustering methods due to the lack of representation learning. Both DEC and IMSAT achieve reasonable clustering results but poor balance, this shows the unfairness of existing deep clustering models which motivates our adding fairness rules. Comparing ours DFDC with the recent deep fair clustering works [137, 92] we can see that our approach consistently outperforms these two in terms of both clustering performance and fairness. Note we report both the deep model's results and the ILP's output in the last iteration. We observe that our deep clustering model's predictions almost converge to the final assignments solved from our

	MNIST-USPS			Reverse-MNIST			HAR		
Methods	ACC	NMI	Balance	ACC	NMI	Balance	ACC	NMI	Balance
DEC [144]	0.586	0.686	0.000	0.401	0.480	0.000	0.571	0.662	0.000
IMSAT [76]	0.804	0.787	0.000	0.525	0.630	0.000	0.812	0.803	0.000
ScFC [8]	0.176	0.053	0.120	0.268	0.105	1.000	–	–	–
FAlg [13]	0.621	0.496	0.093	0.295	0.206	0.667	0.642	0.618	0.420
Fairoids Idea [137]	0.725	0.716	0.039	0.425	0.506	0.430	0.607	0.661	0.166
DFCV [92]	0.825	0.789	0.067	0.577	0.679	0.763	–	–	–
Ours Fair-DEC	0.783	0.752	0.118	0.539	0.614	0.937	0.680	0.725	0.458
Ours Fair-DEC (ILP)	0.774	0.746	0.120	0.509	0.590	1.000	0.668	0.705	0.653
Ours DFDC	**0.939**	**0.876**	0.119	**0.589**	**0.690**	0.946	**0.862**	**0.845**	0.468
Ours DFDC (ILP)	0.936	0.867	0.120	0.583	0.680	1.000	0.842	0.827	0.653

Table 6.3: Comparison of clustering and fairness performance on MNIST-USPS, Reverse-MNIST and HAR. HAR consists of *multi-state PSV* that baselines with dashes are not *applicable*. The first group are plain deep clustering methods, the second group are fair non-deep clustering methods and the third group are deep fair clustering methods including our own. Bold results are the best results among all the baselines except the guaranteed fairness results which are marked with blue. Note we report our average performance results after 10 trials.

ILP module. Moreover, Ours-Fair-DEC largely improve the balance results of DEC and outperforms all the baselines. This result demonstrate that our fair learning algorithm is effective for different clustering backbones. Comparing the deep fair clustering algorithms' results with the plain deep clustering results in Table 6.3, we can see the fairness rules can be seen as positive guidance to improve the clustering ACC in these datasets, our approach is shown to be able to learn from this guidance and improve fairness as well as accuracy.

Results on Tabular Data. We evaluate our approach on tabular data and present the results in Figure 6.2. ScFC [8] is one representative Non-deep fair clustering algorithm which is demonstrated to be effective and efficient on non-deep fair clustering tasks. Here we compare ours DFDC model with it and find that our model achieves similar clustering results as ScFC on tabular data. In terms of fairness, our model can also achieve similar balance comparing to ScFC. We conclude our proposed DFDC also works for tabular

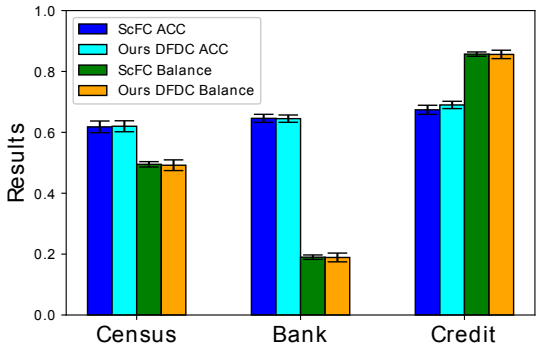
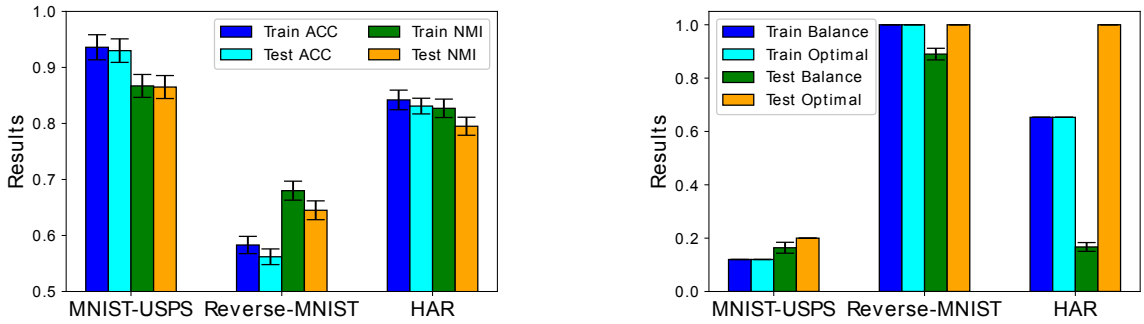


Figure 6.2: Clustering ACC and balance on tabular data. We compare Ours DFDC (Deep Model) to non-deep baseline ScFC.

datasets. This is a surprising result as the competitor ScFC is proposed for this setting to guarantee fairness in classic clustering.

6.5.2 Results Analysis



(a) Predictive Clustering Results

(b) Predictive Clustering Fairness

Figure 6.3: Experimental results on predictive (out-of-sample) clustering settings.

Novel Predictive (Out-of-sample) Clustering. Here we evaluate our method's ability to make predictions on test data *without PSV* information which is a new setting in the fair clustering literature. That is we have already clustered another data set with PSV values and are now making predictions using the model learnt. This is particularly important for practitioners who are, for instance, deploying models on the web (where individuals are reluctant to share PSV information) and we see our results in Figure 6.3.

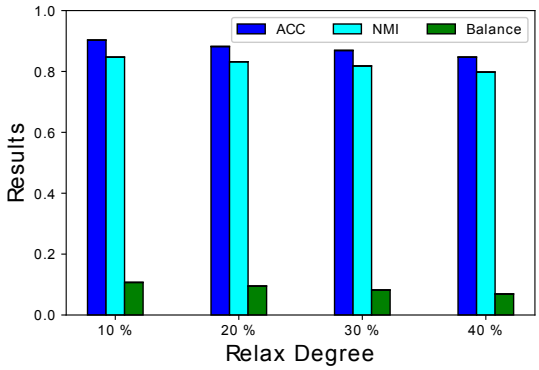


Figure 6.4: Flexible fairness experiments on MNIST-USPS. With larger relaxation the balance drops as expected.

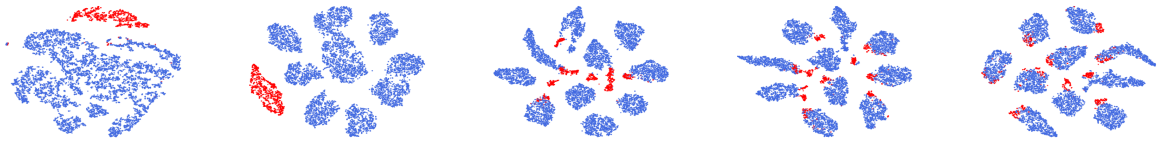
Our approach performs consistently across both train and test sets in terms of clustering performance and fairness. One exception is that the test balance in HAR is much lower than the training balance, we hypothesize this is due to the over-fitting and the different distributions between training and test set within HAR.

Flexible Fairness Constraints. Here we explore how relaxing the optimal fair condition defined as ρ_t produces flexible constraints. We let the new fairness requirement be $P(x \in G_t | x \in C_k) \in [\rho_t * (1-\epsilon), \rho_t * (1+\epsilon)]$ for group t in cluster C_k where ϵ is the relaxation degree. In Figure 6.4, we can see a larger ϵ leads to a lower balance which as expected; since the fairness signals can serve as positive guidance for clustering in MNIST-USPS, we observe the ACC and NMI are decreasing with larger ϵ. Allowing flexible constraints are important as the fairness rules vary across regions.

6.5.3 Further Analysis on Our Model

Here we conduct experiments to better understand its performance by feature space visualization, parameter sensitivity and empirical convergence study.

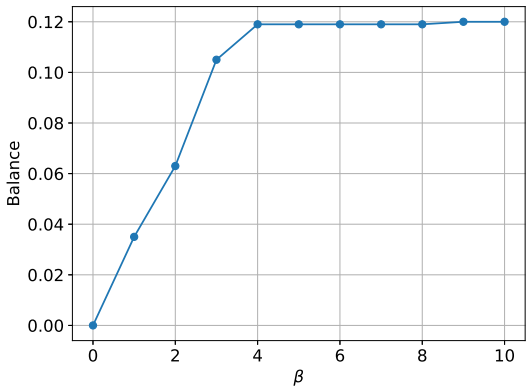
Feature space visualizing. To understand how our model learns a fair representation, we have applied t-SNE [132] to visualize the feature space of MNIST-USPS during different training epochs in Figure 6.5. The initial model is trained with clustering objectives which yield unfair results, once we introduce fairness signals the red instances start to move to different clusters. Meanwhile, we observe our learned representations maintain good



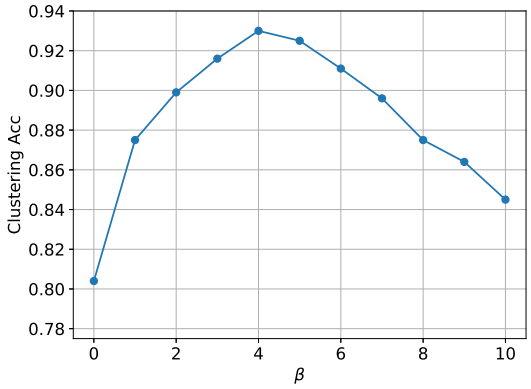
(a) epoch 0 (b) epoch 15 (c) epoch 30 (d) epoch 45 (e) epoch 60

Figure 6.5: t-SNE visualization of learned embedding (MNIST-USPS), color red and blue indicate different PSV values.

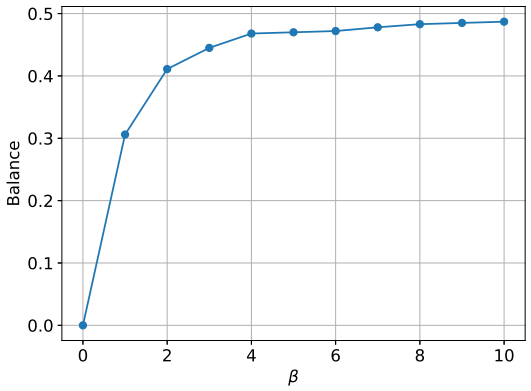
clustering properties.



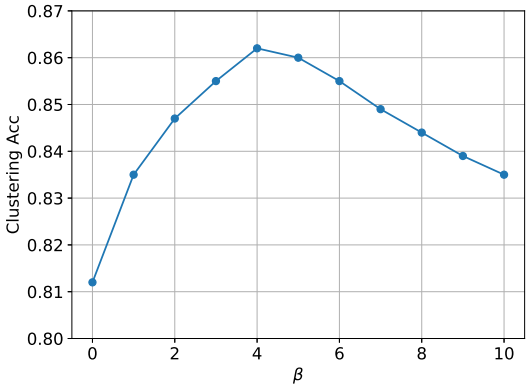
(a) MNIST-USPS (Balance)



(b) MNIST-USPS (ACC)



(c) HAR (Balance)

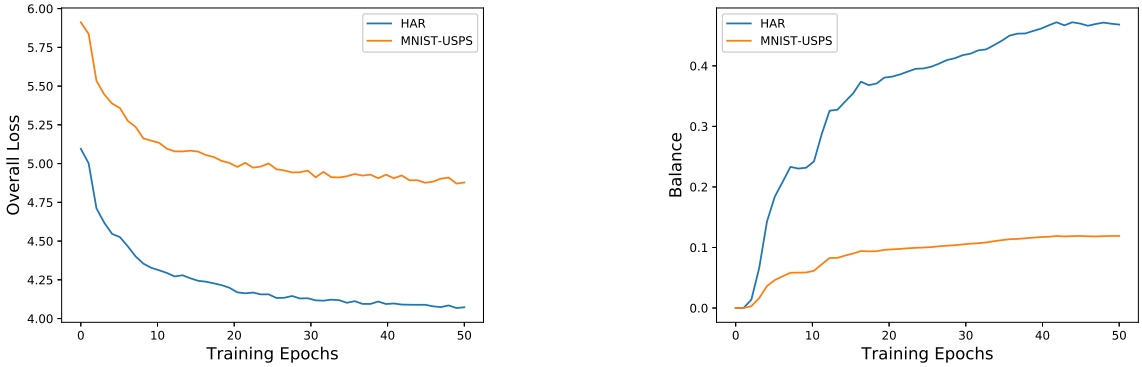


(d) HAR (ACC)

Figure 6.6: Sensitivity analysis of hyper-parameter β which serves as the weights for fairness objective.

Tuning the weight of fairness objective. We experiment on the choices of hyper-parameter β which controls the weight of the fairness objective and report the clustering

results in Figure 6.6. It is straightforward to see from (a) and (c) that as β increases, the training balance increases. Meanwhile, based on (b) and (d) we can find the ACC goes up and down as β increases. Our previous result shows that the fairness constraints can serve as positive guidance for both MNIST-USPS and HAR. That is why the clustering accuracy goes up when we increase β from 0. But we also observe that with a very large β the clustering accuracy will drop. We hypothesize this is because the fairness objective dominates the overall objective so that the impact of clustering objective is hindered. As balance can be tracked during the training process for free, our insight for selecting hyper-parameter β is to pick the smallest β that achieves satisfying balance results.



(a) Convergence Study (Train Loss)

(b) Convergence Study (Balance)

Figure 6.7: Visualizing the learning curves of training loss and fairness measured by the balance on HAR and MNIST-USPS.

Empirical convergence analysis. To investigate the smoothness of learning with clustering and fairness objectives together, we present the learning curves of overall training loss and the balance results in Figure 6.7. We can see from the plots that our model's overall training loss drops quickly and converges after 50 epochs. Meanwhile, our model's balance result also converges after 50 epochs.

6.6 Conclusion

In this paper, we explore the novel direction of adding fairness into deep clustering. This is a challenging problem given the end-to-end deep learning setting which does not

facilitate pre-processing into fairlets and the need for scalability to large data sets. We formulate a group level measure of fairness as integer linear programming and show the problem can be solved efficiently due to total unimodularity (Theorem 6.4.2). We then add this solver into a deep learner and show that our formulation works with multi-state PSV as well as flexible fairness constraints that can occur in real-life applications. Extensive experiments demonstrate the strong performance of our proposed approach and an in-depth analysis including feature visualization, hyper-parameter tuning, convergence analysis, and investigating flexible fair constraints shows its versatility.

Chapter 7

Conclusion

Unlike supervised learning which needs tremendous amounts of labeling to learn a particular task, deep unsupervised learning is motivated to learn and generalize from unlimited unlabeled data which resembles how humans learn the world. As a result, many advances have emerged in deep unsupervised learning, but the gap still exists between the current models and human demands. This includes issues ranging from: (1) the problem of how to improve the quality of learned representation and leverage those representation for downstream applications such as clustering and few-shot learning; (2) understanding or interpreting the predictions from deep unsupervised learning models; (3) dealing with biased predictions for common deep unsupervised learning applications such as clustering and anomaly detection.

This dissertation considers injecting human knowledge into deep unsupervised learning algorithms. We propose novel formulations that learn with various types of human knowledge to address the challenges in the quality of learned representation, explainability, and fairness of deep unsupervised learning. As for the quality of learned representation, we encode a range of constraints (from instance-level to global size level) as loss functions to deep clustering algorithms to improve the clustering quality; moreover, we propose a self-supervised formulation to learn image representations that work well for downstream tasks including clustering and few-shot classification. To address the explainability of deep unsupervised learning, we introduce human-interpretable tags to deep clustering and learn to generate cluster-level explanations and clusters simultaneously. Finally, we address the

fairness problem for deep clustering via "pseudo fair labels" generated based on a minimal modification algorithm and propose to leverage adversarial learning to de-bias deep anomaly detection algorithms.

There are several promising directions to extend our work. The first direction is to develop interactive algorithms that can ask humans for knowledge or guidance. While our constrained clustering work improves the clustering performance largely with different forms of constraints, sometimes the model still suffers from the "negative effect" due to the randomness of the constraints set given to the learner. An ideal interactive algorithm will mitigate this problem via actively querying multiple types of constraints from humans to enhance the performance with minimal cost. Our work on simultaneously clustering and explaining with human interpretable tags can be extended in two directions: (1) we can apply our method on other interesting clustering applications like graph clustering to explain social networks given the interpretable tags for each instance; (2) besides the current form of explanation which provides a list of tags to explain discovered clusters, we can seek for other novel forms of explanations such as combining the conjunction and disjunction of tags or counterfactual explanations. Another emerging line of research would be to explore the fairness problems of deep unsupervised learning, and we can see our works as enforcing group-level fairness rules into some deep clustering and deep anomaly detection models. We feel it is promising to build a general fair representation learning framework that can work with different types of fairness rules such as individual level fairness [50].

Inspired by how humans explore and learn from the real world, our work in this dissertation enhances the representation learning ability, explainability, and fairness of deep unsupervised learning models with novel formulations of learning from various human knowledge. Furthermore, we anticipate that our proposed methods will help advance different aspects of deep unsupervised learning with broader forms of human knowledge.

References

[1] Adadi, A., and Berrada, M. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access 6* (2018), 52138–52160.

[2] Ahmadian, S., Epasto, A., Kumar, R., and Mahdian, M. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 267–275.

[3] Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., and Cremers, D. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648* (2018).

[4] An, J., and Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE 2*, 1 (2015), 1–18.

[5] Angwin, J., Larson, J., Mattu, S., and Kirchner, L. Machine bias. *ProPublica, May 23* (2016), 2016.

[6] Antoniou, A., Edwards, H., and Storkey, A. How to train your maml. *arXiv preprint arXiv:1810.09502* (2018).

[7] Antoniou, A., and Storkey, A. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *arXiv preprint arXiv:1902.09884* (2019).

[8] Backurs, A., Indyk, P., Onak, K., Schieber, B., Vakilian, A., and Wagner, T. Scalable fair clustering. In *International Conference on Machine Learning* (2019), pp. 405–413.

[9] Bade, K., and Nürnberger, A. Creating a cluster hierarchy under constraints of a partially known hierarchy. In *Proceedings of the 2008 SIAM international conference on data mining* (2008), SIAM, pp. 13–24.

[10] Baldi, P. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning* (2012), JMLR Workshop and Conference Proceedings, pp. 37–49.

[11] Basu, S., Bilenko, M., and Mooney, R. J. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (2004), ACM, pp. 59–68.

[12] Basu, S., Davidson, I., and Wagstaff, K. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.

[13] Bera, S., Chakrabarty, D., Flores, N., and Negahbani, M. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems* (2019), pp. 4955–4966.

[14] Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543* (2018).

[15] Bertsimas, D., Orfanoudaki, A., and Wiberg, H. Interpretable clustering: an optimization approach. *Machine Learning* (2020), 1–50.

[16] Beutel, A., Chen, J., Zhao, Z., and Chi, E. H. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075* (2017).

[17] Bickel, S., and Scheffer, T. Multi-view clustering. In *Fourth IEEE International Conference on Data Mining (ICDM'04)* (2004), IEEE, pp. 19–26.

[18] Biddle, D. *Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing*. Gower Publishing, Ltd., 2006.

[19] Bilenko, M., Basu, S., and Mooney, R. J. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning* (2004), ACM, p. 11.

[20] Binns, R. On the apparent conflict between individual and group fairness. In *Proceedings of the 2020 conference on fairness, accountability, and transparency* (2020), pp. 514–524.

[21] Bourlard, H., and Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics 59*, 4-5 (1988), 291–294.

[22] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (2000), pp. 93–104.

[23] Bridle, J. S., Heading, A. J., and MacKay, D. J. Unsupervised classifiers, mutual information and'phantom targets. In *Advances in neural information processing systems* (1992), pp. 1096–1101.

[24] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems 33* (2020), 1877–1901.

[25] Brubach, B., Chakrabarti, D., Dickerson, J., Khuller, S., Srinivasan, A., and Tsepenekas, L. A pairwise fair and community-preserving approach to k-center clustering. In *International Conference on Machine Learning* (2020), PMLR, pp. 1178–1189.

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

[27] Celis, L. E., Huang, L., Keswani, V., and Vishnoi, N. K. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (2019), pp. 319–328.

[28] Chalapathy, R., and Chawla, S. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[29] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 1–58.

[30] Chatziafratis, V., Niazadeh, R., and Charikar, M. Hierarchical clustering with structural constraints. In *International Conference on Machine Learning* (2018), pp. 774–783.

[31] Chen, J., Sathe, S., Aggarwal, C., and Turaga, D. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM international conference on data mining* (2017), SIAM, pp. 90–98.

[32] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709* (2020).

[33] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems* (2016), pp. 2172–2180.

[34] Chen, X., Fain, B., Lyu, L., and Munagala, K. Proportionally fair clustering. In *International Conference on Machine Learning* (2019), PMLR, pp. 1032–1041.

[35] Chierichetti, F., Kumar, R., Lattanzi, S., and Vassilvitskii, S. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems* (2017), pp. 5029–5037.

[36] Chouldechova, A., and Roth, A. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810* (2018).

[37] Dao, T.-B.-H., Kuo, C.-T., Ravi, S., Vrain, C., and Davidson, I. Descriptive clustering: Ilp and cp formulations with applications. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), pp. 1263–1269.

[38] Dao, T.-B.-H., Vrain, C., Duong, K.-C., and Davidson, I. A framework for actionable clustering using constraint programming. In *ECAI* (2016).

[39] Davidson, I., Gourru, A., and Ravi, S. The cluster description problem-complexity results, formulations and approximations. *Advances in Neural Information Processing Systems 31* (2018), 6190–6200.

[40] Davidson, I., and Ravi, S. Intractability and clustering with constraints. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 201–208.

[41] Davidson, I., and Ravi, S. Making existing clusterings fairer: Algorithms, complexity results and insights. In *Thirty-Fourth AAAI Conference on Artificial Intelligence* (2020).

[42] Davidson, I., and Ravis, S. A framework for determining the fairness of outlier detection. In *European Conference on Artificial Intelligence* (2020).

[43] Davidson, I., Wagstaff, K. L., and Basu, S. Measuring constraint-set utility for partitional clustering algorithms. In *Knowledge Discovery in Databases: PKDD 2006*. Springer, 2006, pp. 115–126.

[44] Deecke, L., Vandermeulen, R., Ruff, L., Mandt, S., and Kloft, M. Image anomaly detection with generative adversarial networks. In *Joint european conference on machine learning and knowledge discovery in databases* (2018), Springer, pp. 3–17.

[45] Deepak, P., and Abraham, S. S. Fair outlier detection. In *International Conference on Web Information Systems Engineering* (2020), Springer, pp. 447–462.

[46] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[47] Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. *arXiv preprint arXiv:1605.09782* (2016).

[48] Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J. S., and Pontil, M. Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems* (2018), pp. 2791–2801.

[49] Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems* (2014), pp. 766–774.

[50] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference* (2012), pp. 214–226.

[51] Elazar, Y., and Goldberg, Y. Adversarial removal of demographic attributes from text data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), pp. 11–21.

[52] Erfani, S. M., Rajasegarar, S., Karunasekera, S., and Leckie, C. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition 58* (2016), 121–134.

[53] Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), IEEE, pp. 1778–1785.

[54] Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (2017), JMLR. org, pp. 1126–1135.

[55] Fogel, S., Averbuch-Elor, H., Cohen-Or, D., and Goldberger, J. Clustering-driven deep embedding with pairwise constraints. *IEEE computer graphics and applications 39*, 4 (2019), 16–27.

[56] Fraiman, R., Ghattas, B., and Svarc, M. Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification 7*, 2 (2013), 125–145.

[57] Ghasedi Dizaji, K., Herandi, A., Deng, C., Cai, W., and Huang, H. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 5736–5745.

[58] Ghattas, B., Michel, P., and Boyer, L. Clustering nominal data using unsupervised binary decision trees: Comparisons with the state of the art methods. *Pattern Recognition 67* (2017), 177–185.

[59] Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728* (2018).

[60] Golan, I., and El-Yaniv, R. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems* (2018), pp. 9758–9769.

[61] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[62] Gress, A., and Davidson, I. Probabilistic formulations of regression with mixed guidance. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on* (2016), IEEE, pp. 895–900.

[63] Görnitz, N., Kloft, M., Rieck, K., and Brefeld, U. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research 46* (2013), 235–262.

[64] Guo, X., Gao, L., Liu, X., and Yin, J. Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence (IJCAI-17)* (2017), pp. 1753–1759.

[65] Haeusser, P., Plapp, J., Golkov, V., Aljalbout, E., and Cremers, D. Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition* (2018), Springer, pp. 18–32.

[66] Han, K., Vedaldi, A., and Zisserman, A. Learning to discover novel visual categories via deep transfer clustering. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 8401–8409.

[67] Hawkins, D. M. *Identification of outliers*, vol. 11. Springer, 1980.

[68] Hawkins, S., He, H., Williams, G., and Baxter, R. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery* (2002), Springer, pp. 170–180.

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

[70] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[71] Hendrycks, D., Mazeika, M., Kadavath, S., and Song, D. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems* (2019), pp. 15663–15674.

[72] Hsu, K., Levine, S., and Finn, C. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334* (2018).

[73] Hsu, Y.-C., and Kira, Z. Neural network-based clustering using pairwise constraints. *arXiv preprint arXiv:1511.06321* (2015).

[74] Hu, L., and Chen, Y. Fair classification and social welfare. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (2020), pp. 535–545.

[75] Hu, M., and Chen, S. Doubly aligned incomplete multi-view clustering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), pp. 2262–2268.

[76] Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. Learning discrete representations via information maximizing self-augmented training. In *International Conference on Machine Learning* (2017), pp. 1558–1567.

[77] Huang, C., Cao, J., Ye, F., Li, M., Zhang, Y., and Lu, C. Inverse-transform autoencoder for anomaly detection. *arXiv preprint arXiv:1911.10676* (2019).

[78] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (2015), pp. 448–456.

[79] Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering: a review. *ACM computing surveys (CSUR) 31*, 3 (1999), 264–323.

[80] Ji, X., Henriques, J. F., and Vedaldi, A. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 9865–9874.

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

[82] Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (2002), ACM, pp. 133–142.

[83] Khodadadeh, S., Boloni, L., and Shah, M. Unsupervised meta-learning for few-shot image classification. In *Advances in neural information processing systems* (2019), pp. 10132–10142.

[84] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations* (2015).

[85] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[86] Kleindessner, M., Awasthi, P., and Morgenstern, J. Fair k-center clustering for data summarization. In *International Conference on Machine Learning* (2019), pp. 3448–3457.

[87] Kleindessner, M., Samadi, S., Awasthi, P., and Morgenstern, J. Guarantees for spectral clustering with fairness constraints. In *International Conference on Machine Learning* (2019), pp. 3458–3467.

[88] Krause, A., Perona, P., and Gomes, R. Discriminative clustering by regularized information maximization. *Advances in neural information processing systems 23* (2010), 775–783.

[89] Lampert, C. H., Nickisch, H., and Harmeling, S. Attribute-based classification for zero-shot visual object categorization. *IEEE transactions on pattern analysis and machine intelligence 36*, 3 (2013), 453–465.

[90] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[91] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research 5*, Apr (2004), 361–397.

[92] Li, P., Zhao, H., and Liu, H. Deep fair clustering for visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 9070–9079.

[93] Liu, B., Xia, Y., and Yu, P. S. Clustering via decision tree construction. In *Foundations and advances in data mining*. Springer, 2005, pp. 97–124.

[94] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)* (December 2015).

[95] Lu, Z., and Carreira-Perpinan, M. A. Constrained spectral clustering through affinity propagation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8.

[96] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[97] Mahabadi, S., and Vakilian, A. Individual fairness for k-clustering. In *International Conference on Machine Learning* (2020), PMLR, pp. 6586–6596.

[98] Masci, J., Meier, U., Cire**c***san*, D., and Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks* (2011), Springer, pp. 52–59.

[99] Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence 41*, 8 (2018), 1979–1993.

[100] Moshkovitz, M., Dasgupta, S., Rashtchian, C., and Frost, N. Explainable k-means and k-medians clustering. In *International Conference on Machine Learning* (2020), PMLR, pp. 7055–7065.

[101] Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics 5*, 1 (1957), 32–38.

[102] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), pp. 807–814.

[103] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. *Deep Learning and Unsupervised Feature Learning Workshop, NIPS* (2011).

[104] Noroozi, M., and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision* (2016), Springer, pp. 69–84.

[105] Pang, G., Shen, C., and van den Hengel, A. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 353–362.

[106] Ribeiro, M. T., Singh, S., and Guestrin, C. " 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* (2016), pp. 1135–1144.

[107] Rösner, C., and Schmidt, M. Privacy preserving clustering with constraints. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)* (2018), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[108] Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. Deep one-class classification. In *International conference on machine learning* (2018), pp. 4393–4402.

[109] Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K.-R., and Kloft, M. Deep semi-supervised anomaly detection. In *International Conference on Learning Representations* (2019).

[110] Sakurada, M., and Yairi, T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis* (2014), pp. 4–11.

[111] Sambaturu, P., Gupta, A., Davidson, I., Ravi, S. S., Vullikanti, A., and Warren, A. Efficient algorithms for generating provably near-optimal cluster descriptors for explainability. *Proceedings of the AAAI Conference on Artificial Intelligence 34* (Apr. 2020), 1636–1643.

[112] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International conference on machine learning* (2016), pp. 1842–1850.

[113] Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging* (2017), Springer, pp. 146–157.

[114] Schmidt, M., Schwiegelshohn, C., and Sohler, C. Fair coresets and streaming algorithms for fair k-means. In *International Workshop on Approximation and Online Algorithms* (2019), Springer, pp. 232–251.

[115] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. Estimating the support of a high-dimensional distribution. *Neural computation 13*, 7 (2001), 1443–1471.

[116] Schrijver, A. *Theory of linear and integer programming.* John Wiley & Sons, 1998.

[117] Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 815–823.

[118] Schultz, M., and Joachims, T. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems* (2004), pp. 41–48.

[119] Shah, S. A., and Koltun, V. Deep continuous clustering. *arXiv preprint arXiv:1803.01449* (2018).

[120] Shaham, U., Stanton, K., Li, H., Basri, R., Nadler, B., and Kluger, Y. Spectralnet: Spectral clustering using deep neural networks. In *International Conference on Learning Representations* (2018).

[121] Shao, W., He, L., and Philip, S. Y. Multiple incomplete views clustering via weighted nonnegative matrix factorization with $l_{2,1}$ regularization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2015), Springer, pp. 318–334.

[122] Slack, D., Friedler, S. A., and Givental, E. Fairness warnings and fair-maml: learning fairly with minimal data. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (2020), pp. 200–209.

[123] Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in neural information processing systems* (2017), pp. 4077–4087.

[124] Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems* (2016), pp. 1857–1865.

[125] Strehl, A., Ghosh, J., and Mooney, R. Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)* (2000), vol. 58, p. 64.

[126] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *CVPR* (2018), pp. 1199–1208.

[127] Sweeney, C., and Najafian, M. Reducing sentiment polarity for demographic attributes in word embeddings using adversarial learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (2020), pp. 359–368.

[128] Tao, Z., Liu, H., Li, S., Ding, Z., and Fu, Y. From ensemble clustering to multi-view clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (2017), pp. 2843–2849.

[129] Tzoreff, E., Kogan, O., and Choukroun, Y. Deep discriminative latent space for clustering. *arXiv preprint arXiv:1805.10795* (2018).

[130] Vaidya, P. M. Speeding-up linear programming using fast matrix multiplication. In *30th annual symposium on foundations of computer science* (1989), IEEE Computer Society, pp. 332–337.

[131] Van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv e-prints* (2018), arXiv–1807.

[132] Van der Maaten, L., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research 9*, 11 (2008).

[133] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research 11*, Dec (2010), 3371–3408.

[134] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems* (2016), pp. 3630–3638.

[135] Wagstaff, K., and Cardie, C. Clustering with instance-level constraints. *AAAI/IAAI 1097* (2000), 577–584.

[136] Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. Constrained k-means clustering with background knowledge. In *ICML* (2001), vol. 1, pp. 577–584.

[137] Wang, B., and Davidson, I. Towards fair deep clustering with multi-state protected variables. *arXiv preprint arXiv:1901.10053* (2019).

[138] Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., and Zhang, C. Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (2019), pp. 3670–3676.

[139] Wang, S., Zeng, Y., Liu, X., Zhu, E., Yin, J., Xu, C., and Kloft, M. Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. In *Advances in Neural Information Processing Systems* (2019), pp. 5962–5975.

[140] Wang, X., and Davidson, I. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), ACM, pp. 563–572.

[141] Wang, Y., Wu, L., Lin, X., and Gao, J. Multiview spectral clustering via structured low-rank matrix factorization. *IEEE transactions on neural networks and learning systems 29*, 10 (2018), 4833–4843.

[142] Xia, Y., Cao, X., Wen, F., Hua, G., and Sun, J. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1511–1519.

[143] Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[144] Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (2016), pp. 478–487.

[145] Xing, E. P., Jordan, M. I., Russell, S. J., and Ng, A. Y. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems* (2003), pp. 521–528.

[146] Xu, C., Tao, D., and Xu, C. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634* (2013).

[147] Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Broeck, G. A semantic loss function for deep learning with symbolic knowledge. In *International Conference on Machine Learning* (2018), pp. 5502–5511.

[148] Xu, W., Liu, X., and Gong, Y. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), ACM, pp. 267–273.

[149] Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning* (2017), PMLR, pp. 3861–3870.

[150] Yang, Y., Xu, D., Nie, F., Yan, S., and Zhuang, Y. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing 19*, 10 (2010), 2761–2773.

[151] Zafar, M. B., Valera, I., Rogriguez, M. G., and Gummadi, K. P. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics* (2017), PMLR, pp. 962–970.

[152] Zenati, H., Foo, C. S., Lecouat, B., Manek, G., and Chandrasekhar, V. R. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222* (2018).

[153] Zhang, B. H., Lemoine, B., and Mitchell, M. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (2018), pp. 335–340.

[154] Zhang, H., Basu, S., and Davidson, I. Deep constrained clustering-algorithms and advances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2019), pp. 57–72.

[155] Zhang, H., Basu, S., and Davidson, I. A framework for deep constrained clustering-algorithms and advances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2019), Springer, pp. 57–72.

[156] Zhang, H., and Davidson, I. Deep descriptive clustering. In *IJCAI* (2021).

[157] Zhang, H., and Davidson, I. Deep fair discriminative clustering. *arXiv preprint arXiv:2105.14146* (2021).

[158] Zhang, H., and Davidson, I. Towards fair deep anomaly detection. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (2021), pp. 138–148.

[159] Zhang, H., Zhan, T., Basu, S., and Davidson, I. A framework for deep constrained clustering. *Data Mining and Knowledge Discovery 35*, 2 (2021), 593–620.

[160] Zhang, H., Zhan, T., and Davidson, I. A self-supervised deep learning framework for unsupervised few-shot learning and clustering. *Pattern Recognition Letters 148* (2021), 75–81.

[161] Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *European conference on computer vision* (2016), Springer, pp. 649–666.

[162] Zhao, H., Liu, H., and Fu, Y. Incomplete multi-modal visual data grouping. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), pp. 2392–2398.

[163] Zhou, C., and Paffenroth, R. C. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (2017), pp. 665–674.

[164] Ziko, I. M., Yuan, J., Granger, E., and Ayed, I. B. Variational fair clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 11202–11209.