

# A Framework for Deep Constrained Clustering

Hongjing Zhang · Tianyang Zhan ·  
Sugato Basu · Ian Davidson

Received: date / Accepted: date

**Abstract** 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>.

---

H. Zhang

Department of Computer Science, University of California, Davis. Davis, CA 95616, USA.  
E-mail: [hjzzhang@ucdavis.edu](mailto:hjzzhang@ucdavis.edu)

T. Zhan

Department of Computer Science, University of California, Davis. Davis, CA 95616, USA.  
E-mail: [tzhan@ucdavis.edu](mailto:tzhan@ucdavis.edu)

S. Basu

Google Research, Mountain View, CA, 94043, USA.  
E-mail: [sugato@google.com](mailto:sugato@google.com)

I. Davidson

Department of Computer Science, University of California, Davis. Davis, CA 95616, USA.  
E-mail: [davidson@cs.ucdavis.edu](mailto:davidson@cs.ucdavis.edu)

**Keywords** Constrained Clustering · Deep Learning · Representation Learning · Semi-supervised Learning

## 1 Introduction

Constrained clustering has a long history in machine learning with many standard algorithms being adapted to be constrained (Basu et al., 2008) including Expectation-maximization (EM) (Basu et al., 2004), K-Means (Wagstaff et al., 2001) and spectral methods (Wang and Davidson, 2010). The addition of constraints generated from ground truth labels allows a semi-supervised setting to increase accuracy (Wagstaff et al., 2001) 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 (Davidson et al., 2006) 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 (Davidson and Ravi, 2007). Relaxed formulations (i.e. spectral methods (Lu and Carreira-Perpinan, 2008; Wang and Davidson, 2010)) require solving a full rank eigen-decomposition 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 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 5).
- Deep constrained clustering overcomes a long term issue we reported earlier (Davidson et al., 2006) 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).
- We show the robustness of our proposed framework (see Section 6.4.6) and demonstrate the scalability of our framework on large-scale data set (see Section 6.4.8).
- We conduct ablation study and analyze the contributions of each component within our algorithm (see Section 6.4.7).

This paper is an extension of our previous work (Zhang et al., 2019) 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. We then propose four forms of constraints in section 3 and introduce how to train the clustering network with these constraints in section 4. We then discuss the new way of generating constraints and how to learn multiple types of constraints together in section 5. In section 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 7.

## 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 (Wagstaff and Cardie, 2000; Wagstaff et al., 2001; Xing et al., 2003; Bilenko et al., 2004; Wang and Davidson, 2010). 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 (Davidson et al., 2006) 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.

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 (Joachims, 2002; Schultz and Joachims, 2004; Schroff et al., 2015; Gress and Davidson, 2016) 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 (Bade and Nürnberger, 2008; Chatziafratis et al., 2018) 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 (Xie et al., 2016; Jiang et al., 2017; Yang et al., 2017; Guo et al., 2017; Hu et al., 2017; Ghasedi Dizaji et al., 2017; Caron et al., 2018; Haeusser et al., 2018; Aljalbout et al., 2018; Shaham et al., 2018; Ji et al., 2019; Han et al., 2019). There are approaches (Yang et al., 2017; Hu et al., 2017; Caron et al., 2018; Shaham et al., 2018) 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 (Xie et al., 2016; Guo et al., 2017) 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: (Hsu and Kira, 2015) uses pairwise constraints 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 (Fogel et al., 2019) 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.

### 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 (Xie et al., 2016)). We sketch this method first for completeness.

#### 3.1 Deep Embedded Clustering

We choose to apply our constraints formulation to the deep embedded clustering method DEC (Xie et al., 2016), 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 (Maaten and Hinton, 2008). Note that  $v$  is a constant as  $v = 1$  and  $q_{ij}$  is a soft assignment:

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

**Conversion of  $Q$  To Hard Cluster Assignments  $P$ .** The above normalized similarities 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:

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

**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.

$$\ell_C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (3)$$

The DEC method requires the initial centroids given ( $\mu$ ) 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.

### 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.

#### 3.2.1 Pairwise Constraints

Pairwise constraints (must-link and cannot-link) are well studied (Basu et al., 2008) and we showed they are capable of defining any ground truth set partitions (Davidson and Ravi, 2007). 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:

$$\ell_{ML} = - \sum_{(a,b) \in ML} \log \sum_j q_{aj} * q_{bj} \quad (4)$$

Similarly loss for cannot-link constraints set CL is:

$$\ell_{CL} = - \sum_{(a,b) \in CL} \log (1 - \sum_j q_{aj} * q_{bj}) \quad (5)$$

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

#### 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:

$$\ell_I = \sum_{t \in \{M_t < 0\}} -M_t \sum_j q_{tj}^2 - \sum_{s \in \{M_s > 0\}} M_s \sum_j q_{sj}^2 \quad (6)$$

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.

### 3.2.3 Triplet Constraints

Although pairwise constraints are capable of defining any ground truth set partitions from labeled data (Davidson and Ravi, 2007), 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:

$$\ell_T = \sum_{(a,p,n) \in T} \max(d(q_a, q_n) - d(q_a, q_p) + \theta, 0) \quad (7)$$

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  $\theta$  controls the gap distance between positive and negative instances.  $\ell_T$  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.

### 3.2.4 Global Size Constraints

Experts may more naturally give guidance at a cluster level, previous work (Ghasedi Dizaji et al., 2017) 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:

$$\ell_G = \sum_{c \in \{1, \dots, k\}} \left( \sum_{i=1}^n q_{ic} / n - \frac{1}{k} \right)^2 \quad (8)$$

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 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.

## 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 (Yang et al., 2017) 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:

$$\ell_R = \ell(g(f(x_i)), x_i) \quad (9)$$

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

### 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 (Dao et al., 2016) 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:

$$\ell_{Cardinality} = \sum_{c \in \{1, \dots, k\}} \left( \sum_{P_i=M} q_{ic}/n - \sum_{P_j=F} q_{jc}/n \right)^2 \quad (10)$$

For upper-bound and lower-bound based cardinality constraints (Dao et al., 2016), 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:

$$\ell_{CardinalityBound} = \sum_{c \in \{1, \dots, k\}} \left( \min(0, \sum_{P_i=M} q_{ic} - L)^2 + \max(0, \sum_{P_i=M} q_{ic} - U)^2 \right) \quad (11)$$

**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 ( $\text{Together}(x_a, x_b) \rightarrow \text{Apart}(x_i, x_j)$ ). 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  $CL(x_g, x_h)$ .

#### 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 additive 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 (Davidson et al., 2006), 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 (Guo et al., 2017), the clustering loss  $\ell_C$  and reconstruction loss  $\ell_R$  were added together. To this, we add the instance difficulty loss  $\ell_I$ . 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  $\ell_G$  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  $\ell_P$  or triplet loss  $\ell_T$  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.

#### 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’

**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  $\mu$  via k-means on embedding  $Z$ .  
**for**  $epoch = 1$  **to**  $m$  **do**  
  **for**  $batch = 1$  **to**  $N$  **do**  
    Calculate  $\ell_C$  via Eqn (3),  $\ell_R$  via Eqn (9).  
    Calculate  $\ell_I$  via Eqn (6) or  $\ell_G$  via Eqn (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  $\ell_P$  via Eqn (4, 5) or  $\ell_T$  via Eqn (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**

---

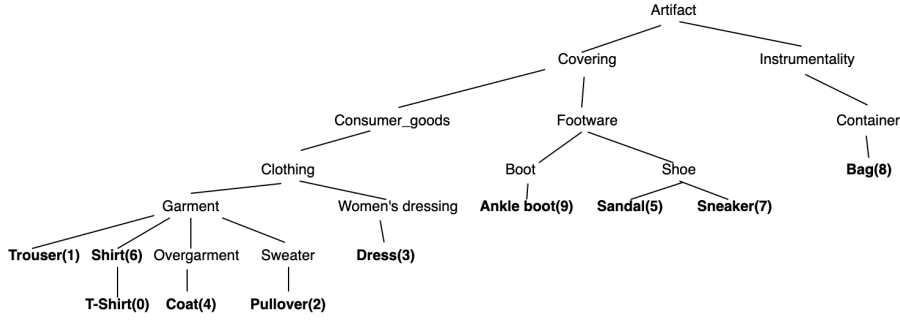
distance among the triplet (i.e., the positive instance should be close to the anchor in latent embedding space while the negative instance will be further). However, to get a good latent embedding, we still need a large amount of ground-truth labels to learn the representation. Here we seek different 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. Here we choose to use WordNet<sup>1</sup> 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 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 1.

We measure the similarity between different classes based on the shortest path that 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  $\text{sim}(C_i, C_j) = \frac{1}{d_{ij}+1}$ . 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

---

<sup>1</sup> <https://wordnet.princeton.edu/>



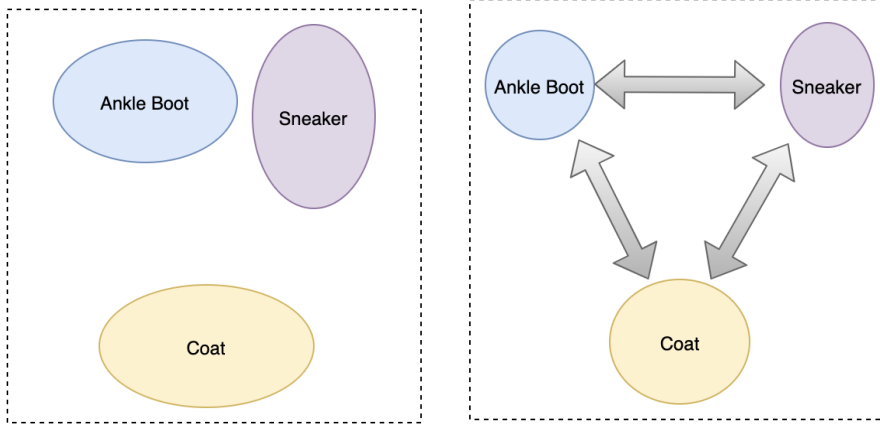
**Fig. 1** WordNet hierarchy of Fashion MNIST data set. The bolded classes are classes we will cluster upon.

requirements we can set the positive threshold  $\theta_p$  to enforce the  $\text{sim}(a, p) > \theta_p$ , similarly we can set negative threshold  $\theta_n$  to ensure both  $\text{sim}(a, n) < \theta_n$  and  $\text{sim}(p, n) < \theta_n$ . Further we require  $\theta_n < \theta_p$ . Note that when  $\theta_p$  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.

Given adequate pairwise constraints, the model can find correct clusters as can be seen from the right-hand side in Figure 2, but the latent semantic similarity relationships 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 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  $\ell_C$ , pairwise loss  $\ell_P$  and triplet loss  $\ell_T$  in an iterative way. The entire learning process is detailed in Algorithm 2.



**Fig. 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.

---

**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 denoising autoencoder to obtain  $Z$

Initialize centroids  $\mu$  via k-means on embedding  $Z$ .

```

for  $epoch = 1$  to  $m$  do
  for  $batch = 1$  to  $N$  do
    Calculate  $\ell_C$  via Eqn (3),  $\ell_R$  via Eqn (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  $\ell_P$  via Eqn (4, 5).
    Update network parameters based on  $\ell_P$ .
  end for
  for  $batch = 1$  to  $N_t$  do
    Calculate  $\ell_T$  via Eqn (7).
    Update network parameters based on  $\ell_T$ .
  end for
  Forward pass to compute  $Z$  and  $S_i = \text{argmax}_j q_{ij}$ .
end for

```

---

## 6 Experiments

All data and code used to perform these experiments are available online ([http://github.com/blueocean92/deep\\_constrained\\_clustering](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 constraints compare with traditional constrained clustering methods? The latter is given the same auto-encoding representation  $Z$  used to initialize our method. (see Table 2)
- Are the new types of constraints we create for the deep clustering method useful in practice? (see Section 6.4.1, 6.4.3, 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 (Davidson et al., 2006)? How our learned embedding overcomes the negative effects of constraints? (see Section 6.4.2)
- How the model performs with constraints generated from ontologies? (see Section 6.4.4)
- How is the proposed model’s robustness towards noisy constraints? (see Section 6.4.6)
- How do the different components of our approach contribute to our final performance? (see our Ablation study in Section 6.4.7)
- How is the scalability of our proposed framework? (see Section 6.4.8)

## 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 (LeCun et al., 1998).

**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 (Lewis et al., 2004). 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.

## 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**) (Strehl et al., 2000; Xu et al., 2003) and clustering accuracy (**Acc**) (Xu et al., 2003). For data point  $x_i$ , let  $l_i$  and  $c_i$  denote its true label and predicted cluster

respectively. Let  $l = (l_1, \dots, l_n)$  and similarity  $c = (c_1, \dots, c_n)$ . **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 (Munkres, 1957). 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.

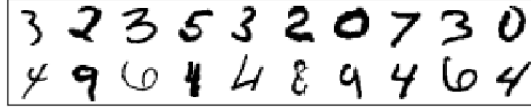
### 6.3 Implementation Details

**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 (Nair and Hinton, 2010) 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**) (Strehl et al., 2000; Xu et al., 2003) and clustering accuracy (**Acc**) (Xu et al., 2003). In our baseline comparisons, we use IDEC (Guo et al., 2017), 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 (Davidson and Ravi, 2007). 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.

**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 3, we give some example difficulty constraints found using this method.



**Fig. 3** Example of instance difficulty constraints. Top row shows the “easy” instances and second row shows the “difficult” instances.

**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 4 shows the generated triplets constraints. Through grid search we set the triplet loss margin  $\theta = 0.1$ .



**Fig. 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.

**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.

## 6.4 Experimental Results

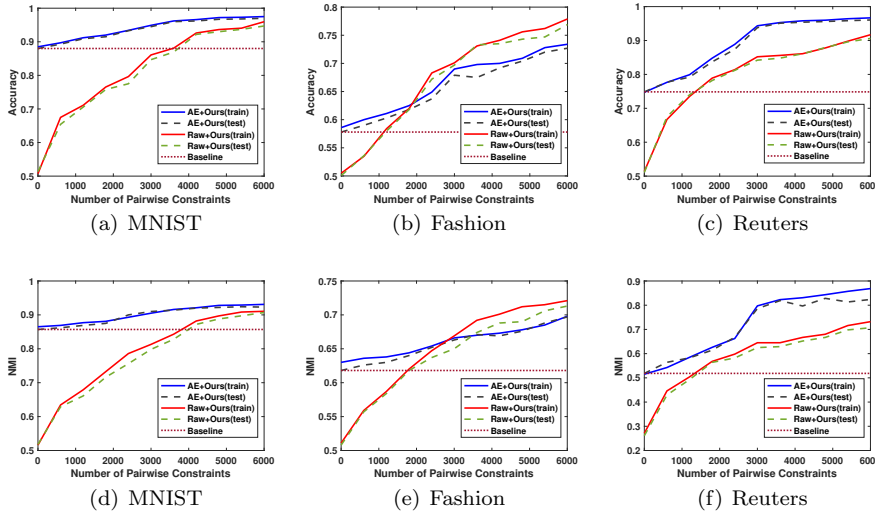
### 6.4.1 Experiments on instance difficulty.

In Table 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

**Table 1** Left table shows baseline results for Improved DEC (Guo et al., 2017) averaged over 20 trials. Right table lists experiments using instance difficulty constraints (mean  $\pm$  std) averaged over 20 trials.

	MNIST	Fashion	Reuters		MNIST	Fashion	Reuters
Acc(%)	88.29 $\pm$ 0.05	58.74 $\pm$ 0.08	75.20 $\pm$ 0.07	Acc(%)	91.02 $\pm$ 0.34	62.17 $\pm$ 0.06	78.01 $\pm$ 0.13
NMI(%)	86.12 $\pm$ 0.09	63.27 $\pm$ 0.11	54.16 $\pm$ 1.73	NMI(%)	88.08 $\pm$ 0.14	64.95 $\pm$ 0.04	56.02 $\pm$ 0.21
Epoch	87.60 $\pm$ 12.53	77.20 $\pm$ 11.28	12.90 $\pm$ 2.03	Epoch	29.70 $\pm$ 4.25	47.60 $\pm$ 6.98	9.50 $\pm$ 1.80

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.



**Fig. 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 (Guo et al., 2017) test set performance.

#### 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 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

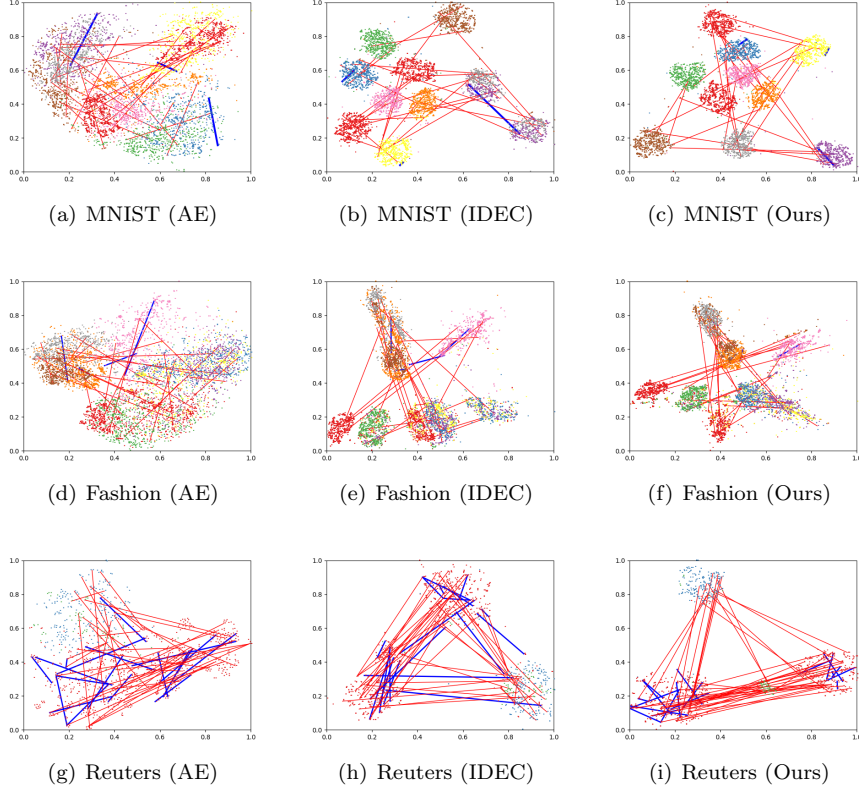
significantly especially for the setup with raw data. We also notice that learning with raw data in Fashion achieves a better result than using autoencoder’s features. This shows that the autoencoder’s features may not always be suitable for DEC’s clustering objective. Overall our results show pairwise constraints can help reshape the representation and improve the clustering results.

We also compare the results with recent work (Hsu and Kira, 2015): 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 (Fogel et al., 2019) due to an issue in their code repository.

**Table 2** Pairwise constrained clustering performance (mean  $\pm$  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 6 and text for an explanation why our method performs well.

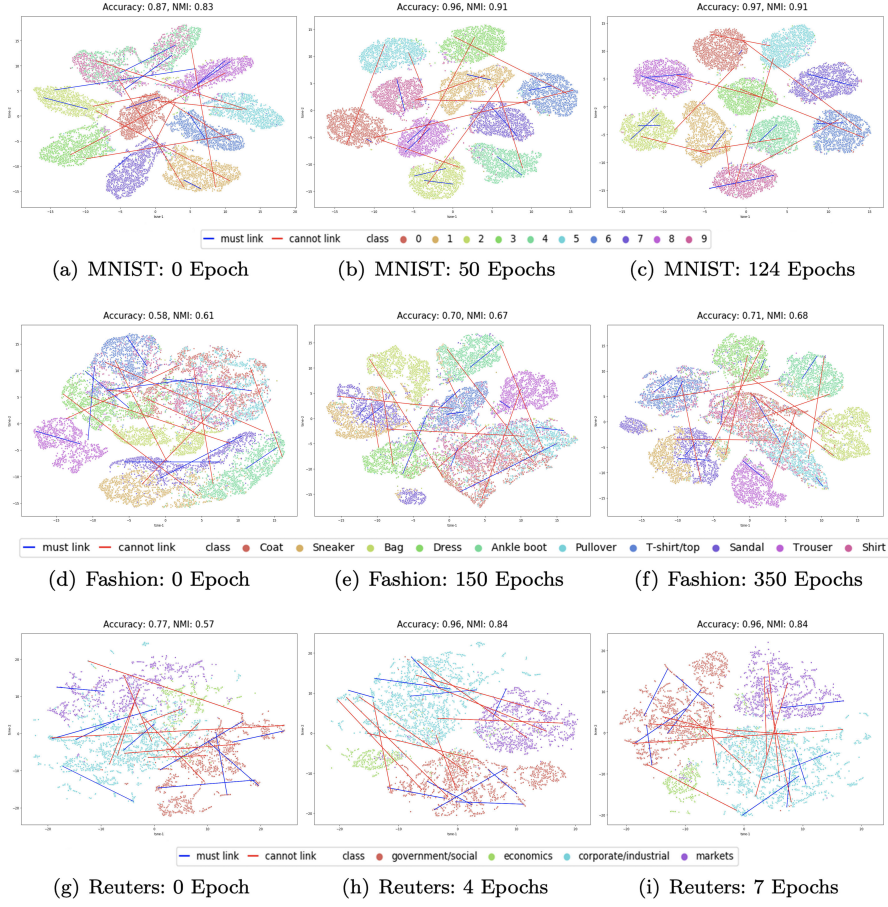
	Flexible CSP*	COP-KMeans	MPCKMeans	Ours
MNIST Acc	0.628 $\pm$ 0.07	0.816 $\pm$ 0.06	0.846 $\pm$ 0.04	<b>0.963 <math>\pm</math> 0.01</b>
MNIST NMI	0.587 $\pm$ 0.06	0.773 $\pm$ 0.02	0.808 $\pm$ 0.04	<b>0.918 <math>\pm</math> 0.01</b>
Negative Ratio	19%	45%	11%	<b>0 %</b>
Fashion Acc	0.417 $\pm$ 0.05	0.548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

**Negative Effects of Constraints.** Our earlier work (Davidson et al., 2006) 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, 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 (Davidson et al., 2006). 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 (Wagstaff et al., 2001; Bilenko et al., 2004; Wang and Davidson, 2010).



**Fig. 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 visualized the embeddings learned. Figure 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 6, we can see the autoencoders embedding is noisy, and lot's of constraints are inconsistent based on our earlier definition (Davidson et al., 2006). 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



**Fig. 7** The embedding for a subset of instances and inconsistent pairwise constraints after several training epochs

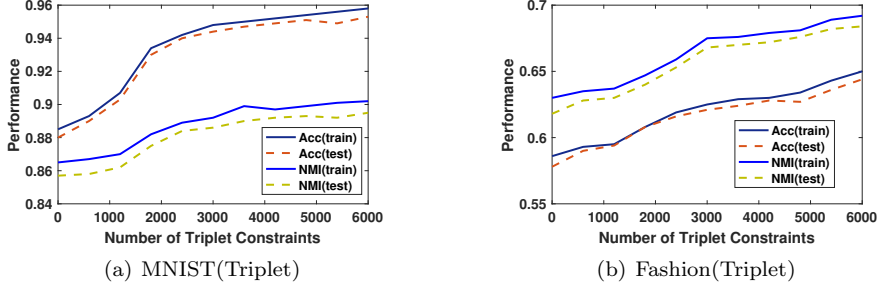
and clustering the data. This result has profound practical significance as practitioners typically only have one constraint set to work with.

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 7.

#### 6.4.3 Experiments on triplet constraints

We experimented on MNIST and FASHION datasets. Figure 4 visualizes example triplet constraints (based on embedding similarity), note the positive instances are closer to anchors than negative instances. In Figure 8, we show the clustering Acc/NMI improves consistently as the number of constraints increasing. Comparing with Figure 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.

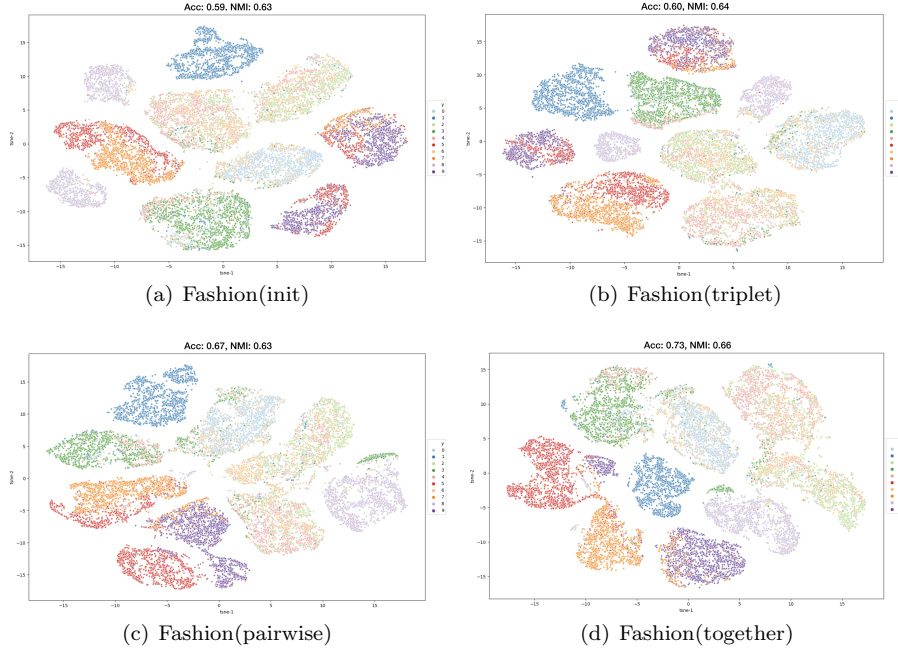


**Fig. 8** Evaluation of the effectiveness of triplet constraints in terms of Acc/NMI.

#### 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 5. Note the threshold for selecting positive pairs  $\theta_p$  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  $\theta_n$  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 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.



**Fig. 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.

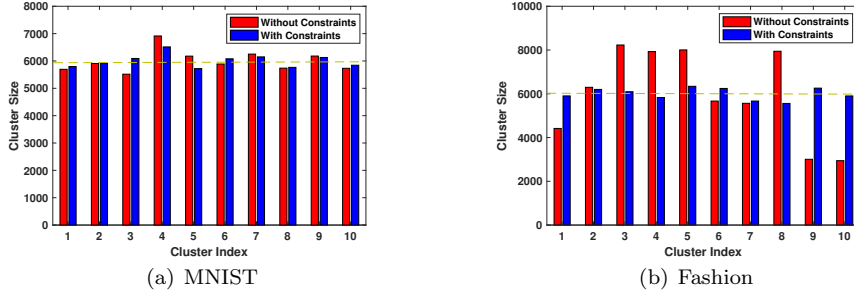
#### 6.4.5 Experiments on global size constraints

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 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.

We also evaluate the clustering performance with global constraints on MNIST (Acc:0.91, NMI:0.86) and Fashion (Acc:0.57, NMI:0.59). Comparing to the baselines in table 1, interestingly, we find the performance improved slightly on MNIST but dropped slightly on Fashion.

#### 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 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

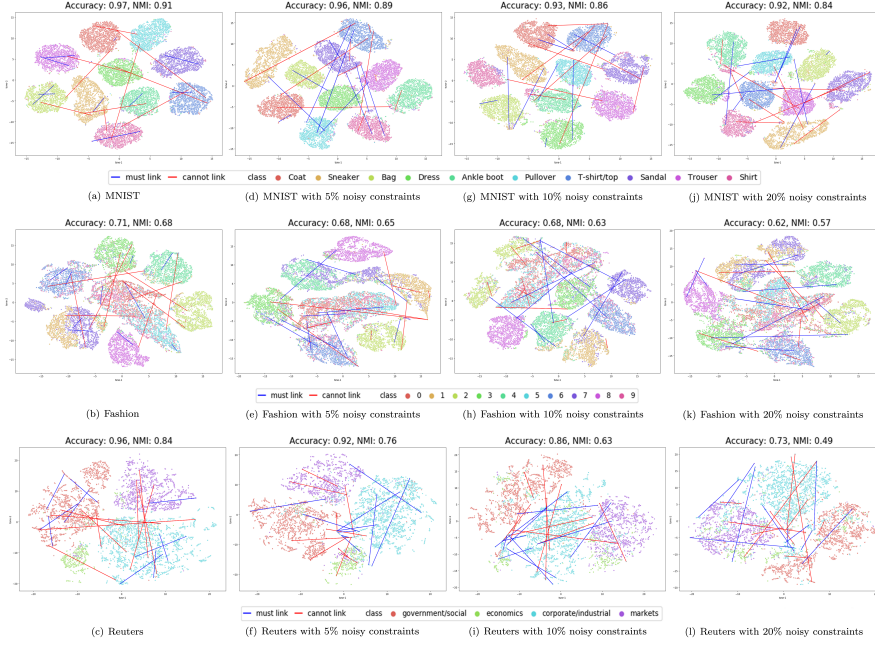


**Fig. 10** Evaluation of the global size constraints. This plot shows each cluster’s size before/after adding global size constraints.

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 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 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 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.

**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 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%.



**Fig. 11** Effects of noisy constraints for MNIST, Fashion and Reuters Dataset.

**Table 3** Pairwise constrained clustering performance (mean  $\pm$  std) averaged over 50 random noisy constraints sets. Baseline model is the model without using pairwise constraints.

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

#### 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.

**Table 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  $< 0.001$ .

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

In Table 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 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.

**Table 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  $\ell_C$  means adding the clustering loss function.

	600	1200	1800	2400	3000
MNIST Acc	0.33	0.40	0.45	0.47	0.49
MNIST Acc (with $\ell_C$ )	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 $\ell_C$ )	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 $\ell_C$ )	0.77	0.79	0.81	0.83	0.85

**Evaluating the Contribution of Clustering Loss.** To measure the contribution of clustering loss  $\ell_C$  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 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.

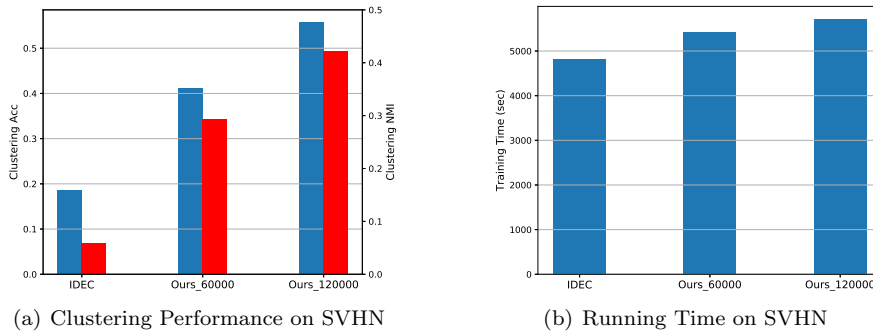
#### 6.4.8 Experiments on Very Large Data Sets

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 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.

**Table 6** Runtime analysis for our proposed approach with different types of constraints. We use the same experimental setting for each type of constraints and average the running time (sec) over 10 trails.

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

We now study our method’s run time on a very large data set, SVHN (Netzer et al., 2011), 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



**Fig. 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.

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 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.

## 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). Importantly, our approach does not suffer from the negative effects of constraints (Davidson et al., 2006) 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 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.

## Acknowledgments

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

## References

- Aljalbout E, Golkov V, Siddiqui Y, Strobel M, Cremers D (2018) Clustering with deep learning: Taxonomy and new methods. arXiv preprint arXiv:180107648
- Bade K, Nürnberger A (2008) Creating a cluster hierarchy under constraints of a partially known hierarchy. In: Proceedings of the 2008 SIAM international conference on data mining, SIAM, pp 13–24
- Basu S, Bilenko M, Mooney RJ (2004) A probabilistic framework for semi-supervised clustering. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 59–68
- Basu S, Davidson I, Wagstaff K (2008) Constrained clustering: Advances in algorithms, theory, and applications. CRC Press

- Bilenko M, Basu S, Mooney RJ (2004) Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the twenty-first international conference on Machine learning, ACM, p 11
- Caron M, Bojanowski P, Joulin A, Douze M (2018) Deep clustering for unsupervised learning of visual features. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 132–149
- Chatziafratis V, Niazadeh R, Charikar M (2018) Hierarchical clustering with structural constraints. In: International Conference on Machine Learning, pp 774–783
- Dao TBH, Vrain C, Duong KC, Davidson I (2016) A framework for actionable clustering using constraint programming. In: ECAI
- Davidson I, Ravi S (2007) Intractability and clustering with constraints. In: Proceedings of the 24th international conference on Machine learning, ACM, pp 201–208
- Davidson I, Wagstaff KL, Basu S (2006) Measuring constraint-set utility for partitional clustering algorithms. In: Knowledge Discovery in Databases: PKDD 2006, Springer, pp 115–126
- Fogel S, Averbuch-Elor H, Cohen-Or D, Goldberger J (2019) Clustering-driven deep embedding with pairwise constraints. *IEEE computer graphics and applications* 39(4):16–27
- Ghasedi Dizaji K, Herandi A, Deng C, Cai W, Huang H (2017) Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In: Proceedings of the IEEE international conference on computer vision, pp 5736–5745
- Gress A, Davidson I (2016) Probabilistic formulations of regression with mixed guidance. In: Data Mining (ICDM), 2016 IEEE 16th International Conference on, IEEE, pp 895–900
- Guo X, Gao L, Liu X, Yin J (2017) Improved deep embedded clustering with local structure preservation. In: International Joint Conference on Artificial Intelligence (IJCAI-17), pp 1753–1759
- Haeusser P, Plapp J, Golkov V, Aljalbout E, Cremers D (2018) Associative deep clustering: Training a classification network with no labels. In: German Conference on Pattern Recognition, Springer, pp 18–32
- Han K, Vedaldi A, Zisserman A (2019) Learning to discover novel visual categories via deep transfer clustering. In: Proceedings of the IEEE International Conference on Computer Vision, pp 8401–8409
- Hsu YC, Kira Z (2015) Neural network-based clustering using pairwise constraints. *arXiv preprint arXiv:151106321*
- Hu W, Miyato T, Tokui S, Matsumoto E, Sugiyama M (2017) Learning discrete representations via information maximizing self-augmented training. In: International Conference on Machine Learning, pp 1558–1567
- Ji X, Henriques JF, Vedaldi A (2019) Invariant information clustering for unsupervised image classification and segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp 9865–9874
- Jiang Z, Zheng Y, Tan H, Tang B, Zhou H (2017) Variational deep embedding: an unsupervised and generative approach to clustering. In: Proceedings of the

- 26th International Joint Conference on Artificial Intelligence, pp 1965–1972
- Joachims T (2002) Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 133–142
- LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324
- Lewis DD, Yang Y, Rose TG, Li F (2004) Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5(Apr):361–397
- Lu Z, Carreira-Perpinan MA (2008) Constrained spectral clustering through affinity propagation. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, IEEE, pp 1–8
- Maaten Lvd, Hinton G (2008) Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605
- Munkres J (1957) Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics* 5(1):32–38
- Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp 807–814
- Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. *Deep Learning and Unsupervised Feature Learning Workshop, NIPS*
- Schroff F, Kalenichenko D, Philbin J (2015) Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 815–823
- Schultz M, Joachims T (2004) Learning a distance metric from relative comparisons. In: *Advances in neural information processing systems*, pp 41–48
- Shaham U, Stanton K, Li H, Basri R, Nadler B, Kluger Y (2018) Spectralnet: Spectral clustering using deep neural networks. In: *International Conference on Learning Representations*
- Strehl A, Ghosh J, Mooney R (2000) Impact of similarity measures on web-page clustering. In: *Workshop on artificial intelligence for web search (AAAI 2000)*, vol 58, p 64
- Wagstaff K, Cardie C (2000) Clustering with instance-level constraints. *AAAI/IAAI* 1097:577–584
- Wagstaff K, Cardie C, Rogers S, Schrödl S, et al. (2001) Constrained k-means clustering with background knowledge. In: *ICML*, vol 1, pp 577–584
- Wang X, Davidson I (2010) Flexible constrained spectral clustering. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp 563–572
- Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: *International conference on machine learning*, pp 478–487
- Xing EP, Jordan MI, Russell SJ, Ng AY (2003) Distance metric learning with application to clustering with side-information. In: *Advances in neural information processing systems*, pp 521–528

- Xu W, Liu X, Gong Y (2003) 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, ACM, pp 267–273
- Yang B, Fu X, Sidiropoulos ND, Hong M (2017) Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In: international conference on machine learning, PMLR, pp 3861–3870
- Zhang H, Basu S, Davidson I (2019) A framework for deep constrained clustering-algorithms and advances. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp 57–72