Deep Self-paced Active Learning for Image Clustering

Helin Zhao
Anhui Provincial International Joint Research Center for Advanced Technology in Medical Imaging, School of Computer Science and Technology, Anhui University
Hefei, China
e21201088@stu.ahu.edu.cn

Wei Chen
Anhui Provincial International Joint Research Center for Advanced Technology in Medical Imaging, School of Computer Science and Technology, Anhui University
Hefei, China
e23201102@stu.ahu.edu.cn

Peng Zhou
Anhui Provincial International Joint Research Center for Advanced Technology in Medical Imaging, School of Computer Science and Technology, Anhui University
Hefei, China
zhoupeng@ahu.edu.cn

***Abstract*—Image clustering attracts much attention in computer vision and multimedia. Due to the absence of labels, even deep clustering still often achieves unreliable results. Although semi-supervised deep clustering can alleviate this problem, it introduces a new problem in the selection of supervised information during semi-supervised learning. To address this issue, in this paper, we propose a novel active deep clustering method that can actively select informative data for querying human annotations and apply these annotations to guide deep clustering. We seamlessly integrate active learning and self-paced learning into a unified deep clustering framework, which can automatically evaluate the difficulty and representativeness of each data and further find the important data. To demonstrate its effectiveness, we conduct extensive experiments on benchmark image data sets. The results show that our proposed method outperforms the state-of-the-art semi-supervised deep clustering and deep active learning methods. The code is available at https://github.com/wodedazhuozi/DSAC.**

***Index Terms*—Image clustering, deep clustering, active clustering, self-paced learning**

I. Introduction

In computer vision and multimedia, image clustering has attracted increasingly more attention. Since deep learning can learn better representations for given data and achieve promising performance on many tasks, deep image clustering is widely studied [1]–[3]. However, due to the absence of labels, clustering is often an ill-posed problem [4], and even deep clustering often mis-clusters data in real applications.

To address this issue, some deep semi-supervised clustering methods are proposed [5]–[9]. For example, Ren et al. extended the popular Deep Embedded Clustering (DEC) [1] to the semi-supervised clustering task by considering the pairwise constraints [5]; Huang et al. designed a consistency regularization loss for semi-supervised clustering with pairwise constraint [6]; Li et al. further adopted the triple loss constraint and label propagation for semi-supervised clustering [9]. These methods apply some supervised information, e.g. labels of partial data or pairwise constraints between some data pairs, to guide the clustering. Since semi-supervised clustering uses some supervised information, it sometimes achieves more reliable clustering results. However, we should notice that, in these semi-supervised clustering, the supervised information is *pre-given*. As we know, the performance of semi-supervised learning highly depends on the quality of the supervised information, and low-quality supervised information may be unhelpful to the clustering. Unfortunately, all these semi-supervised methods just focus on how to adopt the supervised information whereas ignoring the selection of the supervised information.

To tackle this problem, in this paper, we propose a novel active deep clustering method that can actively select some key images for querying human annotations. Active learning often contains multiple interactive batches where in each batch, several data are selected to query for human annotations [10]–[14]. We carefully design a strategy to select as few as possible images and apply the human annotations of these images to guide the clustering. In more detail, we select the *representative* and *difficult* data for annotations. The representative data can well characterize the clustering structure. The difficult data are those data that the clustering method cannot handle well by itself and thus need human annotations. When selecting the difficult data, we harness the complementarity of active

Peng Zhou is the corresponding author.

learning and self-paced learning and seamlessly integrate them into a unified framework. Specifically, self-paced learning can evaluate the difficulty of each data automatically and prefers to use easy data for learning. On the contrary, active learning focuses on difficult data and tends to select the difficult data for annotations. Therefore, we integrate them into a self-paced active learning framework, which directly selects easy data for training and selects difficult data for querying annotations to guide the training. Since the goal of active learning is to select as few annotations as possible to obtain a good enough learning result, to avoid wasting of the annotations, we also design a strategy to generate a number of pseudo-labels based on the annotations, which can further guide the training.

In this framework, we propose an unsupervised module for self-paced learning and an active selection module to select those representative and difficult data and also generate pseudo-labels. After obtaining the human annotations, we provide a supervised module with a carefully designed supervised loss function. We integrate these three modules into a unified framework to make each module be boosted by each other, leading to our Deep Self-paced Active Clustering (DSAC) method. At last, extensive experiments on benchmark image data sets compared with state-of-the-art semi-supervised and active learning methods show the effectiveness and superiority of the proposed method.

The most related work is Active Deep Clustering (ADC) [15]. This method selected pairs of data for annotation and used both labeled and unlabeled data for training the network. Notice that, ADC annotated data with must-link and cannot-link constraints and thus needs a large number of annotation data for training. Different from it, our method directly labels the image and needs only a small number of annotations. Moreover, our method plugs deep active learning into a self-paced learning framework and designs a more careful selection strategy including pseudo-labeling. Therefore, ours can achieve better performance with much fewer annotations.

The main contributions are summarized as follows:

- We integrate active learning and self-paced learning into a unified deep clustering framework, which can seamlessly evaluate the difficulty of data, select the key data, and do the deep clustering.
- We carefully design a novel active selection module, which can select representative and difficult data for annotations. Moreover, in this module, we also provide a strategy to generate the pseudo-labels which can largely decrease the number of annotations.
- We conduct extensive experiments on benchmark data sets by comparing the proposed method with state-of-the-art methods and demonstrate its superiority.

II. Deep Self-paced Active Clustering

In this section, we introduce our DSAC method. Figure 1 shows its framework. To handle the images, we use ConvNeXt [16] as the backbone network, which is based on a convolutional network and works well in image processing. The framework includes three modules: an unsupervised module, an active selection module, and a supervised module. In the clustering task, to handle the unlabeled data, we need an unsupervised module with self-paced learning to train the network. Then, we actively select some key images according to their embeddings for human annotations in the active selection module. After obtaining the annotations of the selected data, we use them to train the network in the supervised module. In the following, we introduce each module in detail.

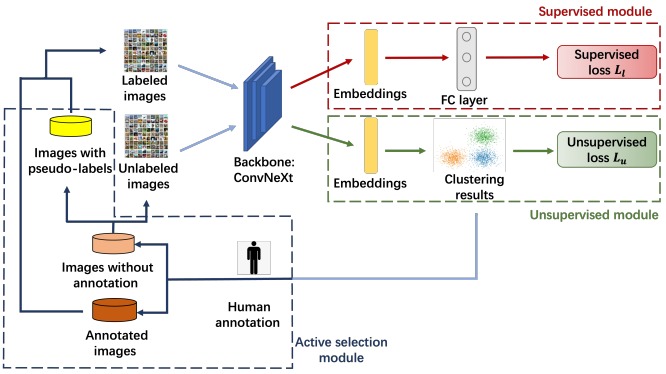


Fig. 1. The framework of DSAC.

A. Unsupervised Module

Given a data set $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} = \mathcal{L} \cup \mathcal{U}$ with n images, where $\mathcal{L}$ is the set of labeled data ($\mathcal{L} = \emptyset$ at the beginning) and $\mathcal{U}$ is the set of unlabeled data, we feed them into the backbone network and obtain their embeddings $F(\mathbf{x}_i; \boldsymbol{\theta}) \in \mathbb{R}^m$, where $\boldsymbol{\theta}$ is the parameters in the backbone network and m is the dimension of the embedding. Then, if $\mathcal{L} \neq \emptyset$, we run semi-supervised kmeans [17] on them, or otherwise, we run standard kmeans on them to obtain c cluster centers.

Let $\mathbf{c}_{\mathbf{x}_i} \in \mathbb{R}^m$ denote the center of the cluster which $\mathbf{x}_i$ belongs to. One natural unsupervised loss function is to make each instance be close to its cluster center, i.e., $\sum_{\mathbf{x}_i \in \mathcal{U}} \|F(\mathbf{x}_i; \boldsymbol{\theta}) - \mathbf{c}_{\mathbf{x}_i}\|_2^2$. However, as discussed before, some data (e.g. the data in the boundary of a cluster) are difficult and may mislead the model. To tackle this problem, we plug it into a self-paced learning framework. In more detail, for each image, we use a weight w_i to indicate its difficulty. The larger w_i is, the easier the image is. To this end, we need a self-paced regularized term $R(w_i) = \lambda(\frac{1}{2}w_i^2 - w_i)$ [18], where λ is an adaptive age parameter, and obtain the following unsupervised loss:

$$\min_{\boldsymbol{\theta}, \mathbf{w}} \mathcal{L}_u = \frac{1}{|\mathcal{U}|} \sum_{\mathbf{x}_i \in \mathcal{U}} \left(w_i \|F(\mathbf{x}_i; \boldsymbol{\theta}) - \mathbf{c}_{\mathbf{x}_i}\|_2^2 + \lambda \left(\frac{1}{2} w_i^2 - w_i \right) \right)$$
$$s.t. \quad w_i \geq 0. \tag{1}$$

When we fix the parameter $\boldsymbol{\theta}$ of the network and optimize the weight w_i, we can obtain its closed-form solution:

$$w_i = \max \left(1 - \frac{\|F(\mathbf{x}_i; \boldsymbol{\theta}) - \mathbf{c}_{\mathbf{x}_i}\|_2^2}{\lambda}, 0 \right). \tag{2}$$

Eq.(2) shows that smaller $\|F(\mathbf{x}_i; \boldsymbol{\theta}) - \mathbf{c}_{\mathbf{x}_i}\|_2^2$, which means $\mathbf{x}_i$ is closer to the center and thus is an easier image, leads to

a larger w_i. It is consistent with the motivation of self-paced learning, i.e., we first use easier data for training and then gradually involve difficult data in training.

B. Active Selection Module

Conventional self-paced learning only focuses on the simple data whereas ignoring the difficult data. In our framework, since we plug active learning into self-paced learning, we also pay attention to the difficult data. When we select key data for annotations, we consider the following two properties:

- *Representativeness.* Since we wish the model can be well trained by as few data as possible, the selected data should be representative enough and can well characterize the cluster structure of all data.
- *Difficulty.* The simple data can be directly used for training as self-paced learning did. Only the difficult data, which the model cannot handle correctly, need the help of human experts.

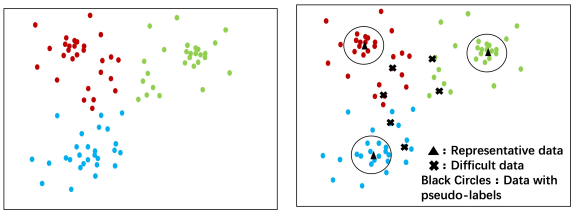
To this end, we design a strategy to select the key data satisfying these two properties. Firstly, we divide the budget into each cluster equally. Assuming that the budget in each batch is b, and there are c clusters, we select $[\frac{b}{c}]$ data from each cluster, where $[\cdot]$ is the rounding operation.

Then, we select the representative data in each cluster. Now, consider the p-th cluster π_p. Firstly, the representative data of a cluster should be near the cluster center. Fortunately, the distance from data to the cluster center can be obtained by the self-paced weights w_i in Eq.(2). The larger the w_i is, the closer the $\mathbf{x}_i$ is to the center. Therefore, we first sort w_i of $\mathbf{x}_i \in \pi_p$ by descending order and choose $\mathbf{x}_i$'s with the top k_1 largest w_i's to form a candidate set. Secondly, representative data should also be compact with other data, which means the representative data should be close to their neighbors. Therefore, in the candidate set, we choose the image with the largest compactness as the representative data of this cluster. To this end, we define the compactness C_i of $\mathbf{x}_i$ as:

$$C_i = \frac{1}{\sum_{\mathbf{x}_j \in KNN(\mathbf{x}_i)} \|F(\mathbf{x}_i;\boldsymbol{\theta}) - F(\mathbf{x}_j;\boldsymbol{\theta})\|_2^2}. \quad (3)$$

where $KNN(\mathbf{x}_i)$ means the k-nearest neighbors of data $\mathbf{x}_i$. If $\mathbf{x}_i$ is closer to its neighbors $\mathbf{x}_j$'s, $\|F(\mathbf{x}_i;\boldsymbol{\theta}) - F(\mathbf{x}_j;\boldsymbol{\theta})\|_2^2$ should be small, and thus C_i should be large.

After selecting the representative data in a cluster, there leave $[\frac{b}{c}] - 1$ budget in this data set. Then, we select $[\frac{b}{c}] - 1$ difficult data in this cluster. Intuitively, it is not necessary to select too many representative data because these data are simple to handle. Contrarily, we should select as many difficult data as possible because the model itself cannot handle them well. That is why in one cluster we just select one representative data and save the remaining budget for the difficult data. As introduced before, since the weight w_i in self-paced learning indicates the difficulty of each instance, we can select the difficult data directly according to w_i. The smaller w_i is, the more difficult the data is. Therefore, we select k_1 data with the smallest w_i's as the candidate set. After obtaining the candidate set, we resort the candidate data according to their compactness which is defined in Eq.(3), and select the $[\frac{b}{c}] - 1$ data with the largest compactness. The reason is that some data far away from the center with small compactness are often outliers which are useless for learning. To avoid selecting the outliers for annotation, we select those with large compactness which contain more information for clustering.



(a) Original data (b) Selection results

Fig. 2. A toy example of active selection module.

Notice that for each cluster we only select one representative data for annotation, which seems not enough. To address this issue, we generate some pseudo-labels according to the annotations of the representative data. Since the data near the cluster centers are reliable data, we can label them directly with pseudo-labels. To this end, in each cluster, we select the nearest k_2 instances from the cluster center and annotate them with the label of the representative data of this cluster. With these pseudo-labels, we can save more budget while achieving better performance.

After the selection and annotation, we move all annotated data together with the data with pseudo-labels from unlabeled set $\mathcal{U}$ to labeled set $\mathcal{L}$. Figure 2 provides a toy example of the active selection module. Figure 2(a) shows the original data and Figure 2(b) shows the selection results. In Figure 2(b), the black triangles denote the representative data and the black crosses denote the difficult data. The data in the circle are those with the pseudo-labels. Algorithm 1 shows the active selection process.

Algorithm 1 Active selection module

Input: c clusters of unlabeled data $\mathcal{U} = \pi_1 \cup \pi_2 \cup, \cdots, \cup \pi_c$, labeled set $\mathcal{L}$, self-paced weights $\mathbf{w}$, budget b.
Output: New labeled set $\mathcal{L}$ and unlabeled set $\mathcal{U}$.

```
Initialize set S = ∅.
for p = 1, ··· , c do
    Construct the candidate set from π_p containing the instances with largest k_1 w_i's.
    Compute the compactness of each instance in the candidate set with Eq.(3) and select the data with the largest compactness as the representative data.
    Select the [b/c] − 1 difficult data according to their w_i's and their compactness.
    Obtain the selected set S' by combining the representative and difficult data, and query for human annotations of S'.
    Obtain the set S'' with the data near the center and generate the pseudo-labels for S''.
    Update S = S ∪ S' ∪ S''.
end for
Update L = L ∪ S and U = U − S.
```

C. Supervised Module

After obtaining the labels and pseudo-labels from the active selection module, we design a supervised module to train the network with the new labeled set $\mathcal{L}$. Given a data $\mathbf{x}_i \in \mathcal{L}$, we denote $\bar{\mathbf{y}}_i$ as its label or pseudo-label. Then, we can adopt the cross-entropy loss as the class loss of $\mathbf{x}_i$. Notice that the dimension of the embedding $F(\mathbf{x}_i;\boldsymbol{\theta})$ is m which is not necessarily equal to the number of clusters c. Therefore, we use an additional fully connected layer (FC layer) $f(\cdot):\mathbb{R}^m \to \mathbb{R}^c$ after $F(\mathbf{x}_i;\boldsymbol{\theta})$ to obtain the final representation $f(F(\mathbf{x}_i;\boldsymbol{\theta}))$ and compute the cross-entropy between $f(F(\mathbf{x}_i;\boldsymbol{\theta}))$ and $\bar{\mathbf{y}}_i$.

Since the importance of each instance is different, here we design a weighted entropy loss, where the important instance will get a large weight. We also use the compactness defined in Eq.(3) as its importance. The data with larger compactness means they are more representative and thus are more important. To this end, we design the following weighted class loss:

$$\mathcal{L}_{class} = \frac{1}{|\mathcal{L}|}\sum_{\mathbf{x}_i\in\mathcal{L}} C_i H(\bar{\mathbf{y}}_i, f(F(\mathbf{x}_i;\boldsymbol{\theta}))) \tag{4}$$

where C_i is the compactness of $\mathbf{x}_i$ defined in Eq.(3), and $H(\mathbf{y},\hat{\mathbf{y}}) = -\sum_{i=1}^{c} y_c log(\hat{y}_c)$ is the cross-entropy loss.

Moreover, to make the embedding have a clearer clustering structure, we also impose the center loss on each instance. The intuition is that each instance should be close to the center of its class. To this end, we define $\mathbf{c}'_{\mathbf{x}_i} \in \mathbb{R}^m$ as the embedding of the center of all data which belong to the same class with $\mathbf{x}_i$ in $\mathcal{L}$. Then we design the following weighted center loss:

$$\mathcal{L}_{center} = \frac{1}{|\mathcal{L}|}\sum_{\mathbf{x}_i\in\mathcal{L}} C_i \|F(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{c}'_{\mathbf{x}_i}\|_2^2 \tag{5}$$

Combining the weighted class loss and weighted center loss with a hyperparameter γ, we obtain the supervised loss:

$$\mathcal{L}_l = \mathcal{L}_{class} + \gamma\mathcal{L}_{center} \tag{6}$$

Combining the unsupervised loss and the supervised loss, we can obtain the whole loss function of our method:

$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_l. \tag{7}$$

We train the neural network by minimizing the loss function Eq.(7). Then, we update the cluster center by feeding all data into the neural network and running semisupervised kmeans on the embedding. At last, we obtain the final clustering results from the final representation $f(F(\mathbf{x}_i;\boldsymbol{\theta}))$. Algorithm 2 shows the whole process of our algorithm.

Algorithm 2 Deep Self-paced Active Clustering

Input: Data $\mathcal{X}$, number of batches for active selection T, the budget of each batch b.
Output: Clustering result.

```
Initialize L = ∅ and U = X. Initialize the parameters of ConvNeXt.
Feed all data into ConvNeXt and run kmeans to obtain the initial cluster centers of the embeddings of all data.
for t = 1, ··· , T do
    Compute the self-paced weight w_i by Eq.(2).
    Select data for querying and update L and U by Algorithm 1.
    Train the ConvNeXt by minimizing Eq.(7).
    Update the cluster centers of unlabeled data by running semi-supervised kmeans on the embeddings.
end for
Obtain the final clustering results from the final embedding f(F(x_i; θ)).
```

III. Experiments

A. Experimental Setup and Implementation Details

We conduct experiments on three widely-used benchmark image data sets including STL-10 [24], CIFAR-10 [25] and CIFAR-100-20 [25]. STL-10 contains 13000 images of 96×96 pixels. CIFAR-10 and CIFAR-100-20 contains 60000 images of 32×32 pixels. In Cifar-100-20, we use the 20 superclasses for evaluation.

We compare our method with the following deep semi-supervised clustering methods including SDEC [5] and AutoEmbedder [19]. We also compare with some state-of-the-art deep active learning methods, including SRAAL [20], TOD [21], Core-GCN [22], ADC [15], and MCDAL [23]. Besides, we also compare with a degenerated version DSAC-R, which replaces the active selection module with randomly selecting the data for annotations, to show the effectiveness of our active selection module. We use Adjusted Rand index (ARI), Normalized Mutual Information (NMI), and Accuracy (ACC) to evaluate the clustering performance.

In the experiments, there are 5 iterations and in each iteration a number of data are selected for annotations. In our method, the budget of each iteration is set as 20 on all data sets. Since the compared methods, including both the semi-supervised and active clustering methods, need much more annotations to achieve acceptable performance, we increase the budgets of these compared methods. In more detail, on the STL-10 data set, which contains 13000 images, we set the budget of one iteration as 200 for all compared methods; on CIFAR-10 and CIFAR-100-20 data sets, which contain 60000 images, we set the budget as 1000 for compared methods. Therefore, the budgets of compared semi-supervised and active learning methods are much larger than ours.

Our method adopts SimCLR [26] to pre-train ConvNeXt. We use AdamW optimizer with a learning rate of 2×10^{-5}. The batch size is 128 on all data sets. The number of epochs are set 100, 50, and 100 on STL-10, CIFAR-10, and CIFAR-100-20, respectively. The dimension of embedding m is fixed as 200. In the supervised module, γ is set as 0.001, 1, 0.01 on STL-10, CIFAR-10, and CIFAR-100-20, respectively. In the unsupervised module, when setting the self-paced parameter λ, we first sort $\|F(\mathbf{x}_i;\boldsymbol{\theta})-\mathbf{c}_{\mathbf{x}_i}\|_2^2$ by ascending order, and then set λ as the first twenty quantile value of $\|F(\mathbf{x}_i;\boldsymbol{\theta})-\mathbf{c}_{\mathbf{x}_i}\|_2^2$, which means the 5% farthest data from the centers will have the weights 0 and other 95% data have non-zero weights. In the active selection module, the size of the candidate set k_1 is fixed to 6 on all data sets. When computing the compactness, we

TABLE I
ARI RESULTS WITH DIFFERENT NUMBERS OF SELECTION ANNOTATIONS ON ALL THE DATA SETS.

Data sets	Number of annotations	SDEC [5]	AutoEmbedder [19]	SRAAL [20]	TOD [21]	CoreGCN [22]	ADC [15]	MCDAL [23]	Number of annotations	DSAC-R	DSAC
STL-10	200	0.0458	0.3552	0.1409	0.1248	0.0104	0.0418	0.1100	20	0.1809	0.7590
	400	0.0457	0.5193	0.1941	0.1610	0.1022	0.0596	0.2555	40	0.3610	0.8058
	600	0.0456	0.5621	0.2236	0.1862	0.0599	0.0804	0.4214	60	0.5116	0.8069
	800	0.0459	0.6120	0.2274	0.2026	0.1487	0.0955	0.5265	80	0.6076	0.8123
	1000	0.0475	0.5690	0.2471	0.2159	0.1076	0.1024	0.5114	100	0.6619	**0.8210**
CIFAR-10	1000	0.0544	0.2794	0.2119	0.2188	0.1103	0.0510	0.2870	20	0.3005	0.4354
	2000	0.0514	0.4038	0.2604	0.3277	0.1438	0.0686	0.3699	40	0.4004	0.7553
	3000	0.0699	0.4057	0.3739	0.4778	0.1685	0.0769	0.4772	60	0.4440	0.7646
	4000	0.0509	0.4511	0.3879	0.6124	0.2264	0.0767	0.5531	80	0.4417	**0.7672**
	5000	0.0503	0.4790	0.3931	0.6797	0.2668	0.0792	0.5971	100	0.4158	0.7664
CIFAR-100 -20	1000	0.0365	0.1450	0.1276	0.0937	0.0556	0.0134	0.1080	20	0.1498	0.2233
	2000	0.0261	0.1553	0.1850	0.1245	0.0597	0.0137	0.1290	40	0.1732	0.3021
	3000	0.0330	0.1430	0.2101	0.1760	0.0762	0.0158	0.1746	60	0.1876	0.3158
	4000	0.0360	0.1912	0.2552	0.2421	0.0992	0.0171	0.2009	80	0.1975	0.3352
	5000	0.0325	0.2318	0.2883	0.3294	0.1142	0.0167	0.2591	100	0.1963	**0.3497**

fix $k = 6$ on all data sets. We initialize $k_2 = 50$ and increase it by 1000 in each iteration. The experiments are conducted on a PC with an NIVDIA GTX 3090Ti GPU.

B. Experimental Results

Tables I and II show the ARI and NMI results of all methods on all data sets, respectively. The increase of our performance with the increase of annotations is fast in the first several iterations. It demonstrates that our selected data are indeed helpful for clustering. DSAC often achieves good results with very few annotations on all data sets, which only needs about 60 to 80 annotations to achieve better performance than other methods, which often need several thousands of annotations. When compared with the random selection version DSAC-R, DSAC achieves better performance. It well demonstrates the effectiveness of the active selection module of DSAC. More experimental results, such as the ACC results and hyperparameter study, are shown in *Appendix*.

Fig. 3. The examples of selection data on STL-10 data set.

Figure 3 shows an example of selection data on STL-10. We show the representative data and difficult data selected by our method in 5 clusters. From Figure 3, we can find that the representative data can well represent their classes, whereas the difficult data are hard to recognize.

C. Ablation Study

We show the effects of the active selection module by ablation study. Firstly, we show the effects of the pseudo-labels. To this end, we design a degenerated version without generating pseudo-labels. Then, we show the effects of selecting the difficult and representative data. To achieve this, we design two variants, where the first one only selects the difficult data and the second one only selects the representative data. Notice that when only selecting the difficult data, we cannot generate the pseudo-labels with the representative data. In this case, we generate the pseudo-labels according to the clustering results. We show the results on STL-10 in Table III. The performance of the variant without pseudo-labels is the worst, which shows the necessity of the pseudo-labels. It is reasonable because the number of annotations is very small and thus we need more data with pseudo-labels to guide the training. The version only selecting the difficult data performs better than the one selecting representative data. The reason is that the difficult data are hard to handle and thus have a greater need for human annotations. Furthermore, DSAC outperforms the one only selecting the difficult data, which means the representative data can further improve the performance.

IV. CONCLUSION

In this paper, we propose a novel deep active image clustering method, which integrates self-paced learning and active learning into a unified deep clustering framework. In the unsupervised module, we apply self-paced learning to evaluate the difficulty of each data. Then, in the active selection module, we carefully select the representative and difficult data for annotations, and generate pseudo-labels to further guide the training. At last, in the supervised module, we design a weighted supervised loss to train the neural network. The extensive experimental results demonstrate the superiority of the proposed method.

ACKNOWLEDGMENTS

This paper is supported by the National Natural Science Foundation of China grants 62176001, and the Natural Science Project of Anhui Provincial Education Department under grant 2023AH030004.

TABLE II
NMI RESULTS WITH DIFFERENT NUMBERS OF SELECTION ANNOTATIONS ON ALL THE DATA SETS.

Data sets	Number of annotations	SDEC [5]	AutoEmbedder [19]	SRAAL [20]	TOD [21]	CoreGCN [22]	ADC [15]	MCDAL [23]	Number of annotations	DSAC-R	DSAC
STL-10	200	0.1193	0.5024	0.0874	0.2061	0.0296	0.0905	0.1906	20	0.2700	0.7965
	400	0.1191	0.5944	0.1221	0.2498	0.1779	0.1197	0.3583	40	0.5134	0.8284
	600	0.1191	0.6506	0.1454	0.2695	0.1162	0.1616	0.5033	60	0.6431	0.8312
	800	0.1195	0.6753	0.1548	0.2872	0.2461	0.1903	0.5841	80	0.7165	0.8353
	1000	0.1192	0.6568	0.1736	0.2935	0.1763	0.2011	0.5858	100	0.7506	**0.8406**
CIFAR-10	1000	0.1207	0.4001	0.1431	0.2901	0.1920	0.0973	0.3535	20	0.4422	0.5953
	2000	0.1168	0.4905	0.1941	0.3925	0.2120	0.1243	0.4380	40	0.5587	0.7850
	3000	0.1104	0.5202	0.3147	0.5205	0.2295	0.1459	0.5300	60	0.5895	0.7926
	4000	0.1174	0.5495	0.3358	0.6354	0.3144	0.1534	0.5889	80	0.5788	**0.7936**
	5000	0.1173	0.5479	0.3479	0.6927	0.3349	0.1594	0.6391	100	0.5897	0.7928
CIFAR-100 -20	1000	0.0879	0.3010	0.0696	0.1594	0.1250	0.0401	0.1777	20	0.2967	0.4004
	2000	0.1142	0.3490	0.1178	0.2002	0.1438	0.0413	0.2106	40	0.3710	0.5023
	3000	0.0920	0.3674	0.1389	0.2574	0.1545	0.0554	0.2598	60	0.3871	0.5234
	4000	0.0923	0.3672	0.1795	0.3234	0.1940	0.0488	0.2913	80	0.3974	**0.5305**
	5000	0.0871	0.4270	0.2093	0.4122	0.2086	0.0503	0.3514	100	0.4037	0.5295

TABLE III
ABLATION STUDY ON STL-10 DATA SET.

Number of annotations	Without pseudo-labels		Only difficult data		Only representative data		DSAC	
	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
20	0.3585	0.2516	0.6992	0.6479	0.7104	0.6650	0.7965	0.7591
40	0.5773	0.4322	0.7505	0.7073	0.7618	0.7244	0.8285	0.8059
60	0.7060	0.6191	0.7649	0.7211	0.7467	0.7804	0.8312	0.8069
80	0.7481	0.6662	0.7730	0.7289	0.7501	0.6978	0.8362	0.8156
100	0.7662	0.6789	0.7883	0.7489	0.7527	0.6982	0.8407	0.8210

REFERENCES

[1] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, volume 48, pages 478–487, 2016.

[2] Xi Peng, Jiashi Feng, Joey Tianyi Zhou, Yingjie Lei, and Shuicheng Yan. Deep subspace clustering. *IEEE TNNLS*, 31(12):5509–5521, Feb. 2020.

[3] Qianqian Wang, Wei Xia, Zhiqiang Tao, Quanxue Gao, and Xiaochun Cao. Deep self-supervised t-sne for multi-modal subspace clustering. In *ACM MM*, pages 1748–1755. ACM, 2021.

[4] Mimi Zhang. Weighted clustering ensemble: A review. *Pattern Recognit.*, 124:108428, 2022.

[5] Yazhou Ren, Kangrong Hu, Xinyi Dai, Lili Pan, Steven C. H. Hoi, and Zenglin Xu. Semi-supervised deep embedded clustering. *Neurocomputing*, 325:121–130, 2019.

[6] D. M. Huang, Jie Hu, Tianrui Li, Shengdong Du, and Hongmei Chen. Consistency regularization for deep semi-supervised clustering with pairwise constraints. *IJMLC*, 13:3359 – 3372, 2022.

[7] Arezoo Hatefi, Xuan-Son Vu, Monowar H. Bhuyan, and Frank Drewes. Cformer: Semi-supervised text clustering based on pseudo labeling. *CIKM*, 2021.

[8] Ying Zhang, Xiangli Li, and Mengxue Jia. Semi-supervised nonnegative matrix factorization with pairwise constraints for image clustering. *IJMLC*, 13:3577 – 3587, 2022.

[9] Xiaocui Li, Hongzhi Yin, Ke Zhou, and Xiaofang Zhou. Semi-supervised clustering with deep metric learning and graph embedding. *WWW*, 23:781–798, 2019.

[10] Steven CH Hoi, Rong Jin, Jianke Zhu, and Michael R Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on Machine learning*, pages 417–424. ACM, 2006.

[11] Rita Chattopadhyay, Zheng Wang, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Batch mode active sampling based on marginal probability distribution matching. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):13, 2013.

[12] Hanmo Wang, Liang Du, Peng Zhou, Lei Shi, and Yi-Dong Shen. Convex batch mode active sampling via α-relative pearson divergence. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[13] Hanmo Wang, Liang Du, Lei Shi, Peng Zhou, Yuhua Qian, and Yi-Dong Shen. Experimental design with multiple kernels. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 419–428. IEEE Computer Society, 2015.

[14] Peng Zhou, Bicheng Sun, Xinwang Liu, Liang Du, and Xuejun Li. Active clustering ensemble with self-paced learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2023.

[15] Bicheng Sun, Peng Zhou, Liang Du, and Xuejun Li. Active deep image clustering. *KBS*, 252:109346, 2022.

[16] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, pages 11966–11976, 2022.

[17] Jordan Yoder and Carey E Priebe. Semi-supervised k-means++. *Journal of Statistical Computation and Simulation*, 87(13):2597–2608, 2017.

[18] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G. Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *ACM MM*, pages 547–556. ACM, 2014.

[19] Abu Quwsar Ohi, Muhammad F. Mridha, Farisa Benta Safir, Md. Abdul Hamid, and Muhammad Mostafa Monowar. Autoembedder: A semi-supervised DNN embedding system for clustering. *KBS*, 204:106190, 2020.

[20] Beichen Zhang, Liang Li, Shijie Yang, Shuhui Wang, Zheng-Jun Zha, and Qingming Huang. State-relabeling adversarial active learning. In *CVPR*, pages 8753–8762, 2020.

[21] Siyu Huang, Tianyang Wang, Haoyi Xiong, Jun Huan, and Dejing Dou. Semi-supervised active learning with temporal output discrepancy. In *ICCV*, pages 3427–3436, 2021.

[22] Razvan Caramalau, Binod Bhattarai, and Tae-Kyun Kim. Sequential graph convolutional network for active learning. In *CVPR*, pages 9583–9592, 2021.

[23] Jae Won Cho, Dong-Jin Kim, Yunjae Jung, and In So Kweon. Mcdal: Maximum classifier discrepancy for active learning. *IEEE TNNLS*, pages 1–11, 2022.

[24] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, volume 119, pages 1597–1607. PMLR, 2020.