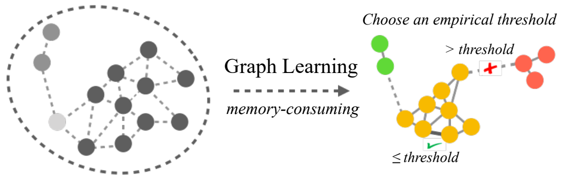
Learn to Cluster Faces via Pairwise Classification

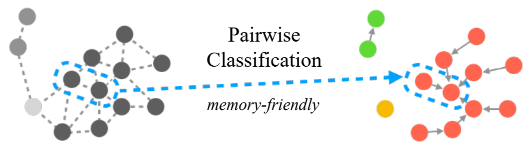
Junfu Liu*, Di Qiu*, Pengfei Yan†, Xiaolin Wei
Meituan

Abstract

Face clustering plays an essential role in exploiting massive unlabeled face data. Recently, graph-based face clustering methods are getting popular for their satisfying performances. However, they usually suffer from excessive memory consumption especially on large-scale graphs, and rely on empirical thresholds to determine the connectivities between samples in inference, which restricts their applications in various real-world scenes. To address such problems, in this paper, we explore face clustering from the pairwise angle. Specifically, we formulate the face clustering task as a pairwise relationship classification task, avoiding the memory-consuming learning on large-scale graphs. The classifier can directly determine the relationship between samples and is enhanced by taking advantage of the contextual information. Moreover, to further facilitate the efficiency of our method, we propose a rank-weighted density to guide the selection of pairs sent to the classifier. Experimental results demonstrate that our method achieves state-of-the-art performances on several public clustering benchmarks at the fastest speed and shows a great advantage in comparison with graph-based clustering methods on memory consumption.



(a) Graph-based methods



(b) Our method

Figure 1. An overview of the main differences between popular graph-based face clustering methods and ours. The nodes with different colors represent different classes. (a) Graph-based methods focus on the learning of discriminative embeddings but need to set an empirical threshold to connect or disconnect samples. (b) Our method can directly determine the pairwise relationship through a classifier and generate the final clusters. Besides, by dealing with the face clustering task from graph-level to pair-level, our method largely reduces the memory consumption.

1. Introduction

Massive face data are accessible on the Internet nowadays, thus boosting the development of face analysis technology [1, 13]. However, the large-scale unlabeled face data has resulted in quite high annotation prices, and human annotations are not always reliable. Therefore, exploiting unlabeled face data has attracted great interest. Aiming at assigning pseudo labels to unlabeled face images, face clustering is a fundamental face analysis task and has wide applications in real-world scenarios like the dataset preparation or cleaning for face recognition [22, 23] and photo albums management [19].

Traditional clustering methods suffer from impractical assumptions and their performances could be sensitive to hyper-parameters. For instance, DBSCAN [2] assumes clusters to be in the same density, and the performance of K-Means [3] highly depends on the original k. Recently, graph-based face clustering methods have drawn much attention for their satisfying performances by learning representative embeddings without making oversimplified assumptions [4, 5, 6]. But the computational costs and demands for memory usage stand out especially on large-scale face clustering scenes, due to the surprisingly high price of learning on large graphs [20, 26]. And some works have been proposed to alleviate such problems. In [7], Yang *et al.* designed a graph convolutional network-based confidence estimator to select nodes with large confidence, which reduces the number of highly overlapped subgraphs and leads to promotions on efficiency. Although the confidence estimator contains only one single layer, an out-of-memory issue still occurs when dealing with large-scale graphs, as will

*Co-first authors. liujeff97@gmail.com, qiudi@meituan.com
†Corresponding author. yanpengfei03@meituan.com

be revealed in Section 4. Guo *et al.* [8] exploited cluster-level feature distribution by defining a density-aware graph, which reduces memory consumption by avoiding learning on the whole graph, but this method can still fall on large-scale face clustering scenes when limited memory capacity is available. More importantly, like previous graph-based clustering methods, to generate the final clusters in the inference stage, it still requires a manual-setting threshold to determine the connectivities between samples. The final performance of these clustering methods is so sensitive to the empirical threshold that can drop heavily when the threshold is set improperly, which restricts the generalization of various real-world scenes.

To address the problems above, in this work, we focus on the fundamental concerns of face clustering tasks, *i.e.*, the homogeneity of samples. Specifically, we explore a memory-friendly and threshold-free face clustering method from the pairwise angle, leading to promotions in both efficiency and accuracy. Instead of learning on the graph-level or cluster-level, we handle face clustering on the minimum level, *i.e.*, the pair-level, which largely reduces memory consumption. We adopt a simple yet effective classifier network and the classifier directly gives the connection relationship of a pair of samples, which also frees us from the manual setting of thresholds. Besides, inspired by the graph learning-based methods where contextual information is emphasized, the classifier is trained on our designed weighted-neighbor features, which brings considerable performance gains to both the classifier and the overall clustering task. As Figure 1 illustrates, popular graph-based methods rely on an empirical threshold to determine the connectivities between samples after learning distinguishable feature embeddings with high consumption on memory. Contrastively, our method can directly give the pairwise relationships and generate the final clusters efficiently with very limited memory usage.

Moreover, to enhance the efficiency of the whole pipeline, we propose a rank-weighted density. The rank-weighted density gives more attention to nearby neighbors, which eases the influences of outliers in the procedure of finding neighbors with higher density. The introduction of this more precise rank-weighted density not only reduces the time complexity via selecting fewer pairs sent to the classifier by an order of magnitude, but also makes a further contribution to the performance of our method.

Our main contributions are summarized as follows: (1) Different from popular graph-based methods, we propose a face clustering method based on pairwise classification, which is much more memory-friendly and also threshold-free. (2) A rank-weighted density is proposed to instruct the pair selection sent to the classifier, resulting in further promotions in both efficiency and accuracy. (3) Our method improves the performance on public face clustering benchmarks, achieving Pairwise F-score at 90.67 and BCubed F-score at 89.54 on the MS-Celeb-1M dataset, surpassing previous public state-of-the-art methods at 90.60 [8] and 86.09 [7] respectively.

The following Section 2 will give a brief review of the related work. Section 3 details our proposed method. Experimental results and analysis are presented in Section 4 and we finally conclude the paper in Section 5.

2. Related Work

The challenges in face clustering mainly result from the large variations of face representations in the real world [25]. Most existing graph-based clustering methods achieve satisfying performance by learning representative features from an affinity graph, yet relies on empirical thresholds to finally decide whether two samples should belong to the same class. Thus, performances of such methods are heavily influenced by manual settings and could drop severely in real-world face clustering scenes.

Pairwise-relationship Prediction in Face Clustering. Determining the pairwise relationship between samples is an essential part of the face clustering task in inference. Zhan *et al.* [6] aggregated multi-view information and discovered robust pairs by an MLP classifier, and finally set an empirical threshold to determine pairwise relationships between two samples. Wang *et al.* [4] predicted linkage-probability between nodes and found connections with dynamic thresholds. Yang *et al.* [7] utilized a pre-defined threshold to cut off the edges with small similarities despite the introduced connectivity estimator. Guo *et al.* [8] proposed the learnable local and long-range features to impair sensitivity to the clustering threshold, but a suitable threshold was still needed to decide the ultimate pairwise relationship. Different from previous approaches, our method focuses on exploiting a robust module to discover the pairwise relationship and aims at getting rid of the empirical threshold. This allows our method to be more generalizable in real-world face clustering scenes.

GNN-based Face Clustering. Recent works on face clustering task primarily adopt GNN-based models to learn representative feature embeddings, and tend to estimate relative order between samples by utilizing the "density" [21, 24] or "confidence" [7] of each sample, leading to state-of-the-art performances. Wang *et al.* [4] leveraged a graph convolutional network (GCN) model to find the connectivities between a pivot node and its neighbors. Yang *et al.* [5] learned clusters in GCN detection and segmentation modules. In [7], Yang *et al.* proposed an improved GCN-based approach to learn vertex confidence on the massive affinity graph and edge connectivity on the local subgraphs afterward. Guo *et al.* [8] exploited cluster-level feature distribution by learning context-aware feature embeddings based on GCNs. The success of GNN-based methods mainly lies in

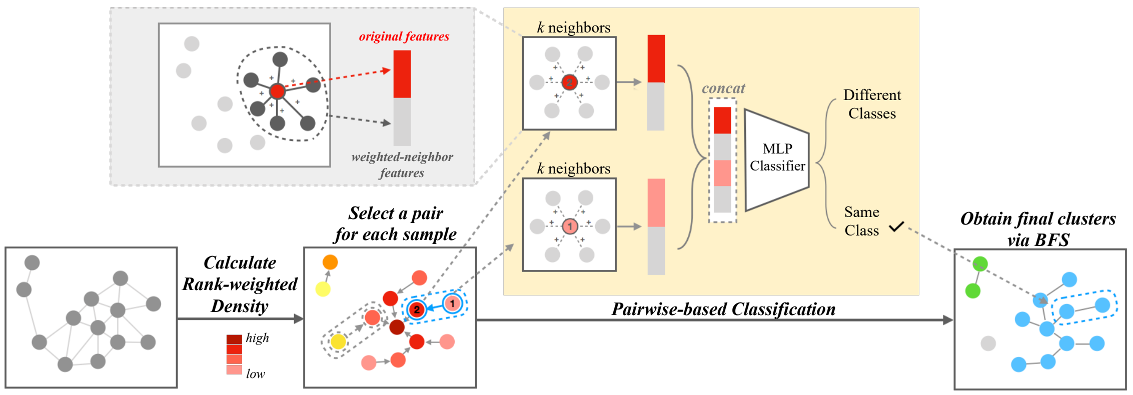

original features
weighted-neighbor features
k neighbors
k neighbors
concat
MLP Classifier
Different Classes
Same Class
Select a pair for each sample
Calculate Rank-weighted Density
high
low
Pairwise-based Classification
Obtain final clusters via BFS

Figure 2. Overview of our proposed face clustering framework. We first use rank-weighted density to form pairs for each sample. The feature vector of each sample is updated by concatenating its original feature and weighted-neighbor feature, and we combine the updated features of a pair of samples to form the pair features, then each pair is sent to a trained classifier to determine the pairwise relationship. After the linkages of samples are determined, we apply BFS algorithm to obtain the final clusters.

the learning of context-aware features, where the neighborhood information is greatly emphasized. However, GNN-based methods can easily cause high computational cost, thus cannot work on large-scale clustering scenes on the condition of memory insufficiency.

In this paper, we address the memory-consuming problem raised in popular state-of-the-art methods and free the face clustering methods from setting empirical thresholds by learning robust pairwise relationship, and further improves the accuracy and efficiency by introducing rank-weighted density.

3. Methodology

As has been discussed above, current GNN-based face clustering algorithms could be very memory-consuming, and rely on expert knowledge to set proper thresholds to determine the connectivities between faces. To address such challenges, we propose a face clustering method based on pairwise classification. We formulate the face clustering task as a classification task between faces and train a classifier to directly determine whether two faces should belong to the identical class. Furthermore, to generate face clusters more efficiently, we select face pairs sent to the classifier based on a novel rank-weighted density manner, which turns out to be more insensitive to outliers. Figure 2 illustrates an overview of our proposed method.

3.1. Pairwise-based Classification

Given a face dataset $F = [\boldsymbol{f_1}, \boldsymbol{f_2}, ..., \boldsymbol{f_N}] \in \mathbb{R}^{N\times D}$, where N is the number of face images and D the dimension of features extracted from a trained CNN, face clustering is a task of dividing faces into different clusters where faces in the same cluster share one identity. Many popular face clustering methods focus on learning representative embeddings, then use empirical thresholds to determine the connectivities between samples, and such thresholds have great influences on the final performance of face clustering but rely heavily on experiences and vary greatly with different datasets. Meanwhile, the graph learning on large-scale graphs (*e.g.*, with nodes more than 10 million) can be very memory-consuming. To resolve these problems, we change the learning from the *graph-level* to the *pair-level*. Given a pair of faces, what we look for is a binary classifier to directly give the relationship between these two faces, *i.e.*, whether they belong to the same identity or not.

Motivation of classifier. The huge memory consumption in graph learning-based methods results from the huge size N of the dataset. But we do not necessarily need to learn knowledge on such a big scale and face clustering actually tends to assign samples with pseudo labels, so we solve the face clustering task on the minimum level, the pair-level, and train a classifier to predict whether two samples should belong to the same class. Take a pair of image features $\boldsymbol{f_1}, \boldsymbol{f_2}$ as input, naturally, we can concatenate the two feature vectors to form a new feature vector sent to the network, and we only need a network as simple as Multi-Layer Perceptron (MLP). One can also see that our method can be heuristically generalized to other kinds of classifiers.

Inspired from GNN-based face clustering methods which utilize contextual information to help cluster faces, and the intuition that a sample is very likely to share identity with its neighbors, we use neighbor features to refine its features. For sample x_i, we use k-NN to select its top k nearest neighbors x_{i1}, x_{i2},..., x_{ik}, and similarities s_{i1}, s_{i2},..., s_{ik} between sample x_i and its neighbors can be calculated as the inner product of two features. Instead of a simple summation, we apply the similarities as the weights to calculate

```
Algorithm 1 Pairwise-based Clustering
Input: feature sets $\mathcal{F}$, the number of neighbors $k$, trained
    classifier $\mathcal{C}$.
Output: clusters $\mathbb{C}$
 1: procedure CLUSTERING
 2:     $\mathbb{P}$ = FIND PAIRS VIA DENSITY($\mathcal{F}$, $k$)
 3:     $\mathbb{R}$ = Use $\mathcal{C}$ to get relationships of pairs in $\mathbb{P}$.
 4:     $\mathbb{C}$ = Use BFS Algorithm to generate clusters.
 5:     return $\mathbb{C}$
 6: end procedure

 7: function FIND PAIRS VIA DENSITY($\mathcal{F}$, $k$)
 8:     $\mathbb{P} = \emptyset$
 9:     for all sample $\boldsymbol{x}$ in $\mathcal{F}$ do
10:         Obtain similarities with its $k$ nearest neighbors.
11:         Calculate its rank-weighted density.
12:     end for
13:     for all sample $\boldsymbol{x}$ in $\mathcal{F}$ do
14:         Find the first neighbor $\boldsymbol{y}$ with higher density.
15:         if $\boldsymbol{y}$ exists then
16:             $\mathbb{P} = \mathbb{P} \cup \{(\boldsymbol{x}, \boldsymbol{y})\}$
17:         end if
18:     end for
19:     return $\mathbb{P}$
20: end function
```

the weighted-neighbor features of a sample:

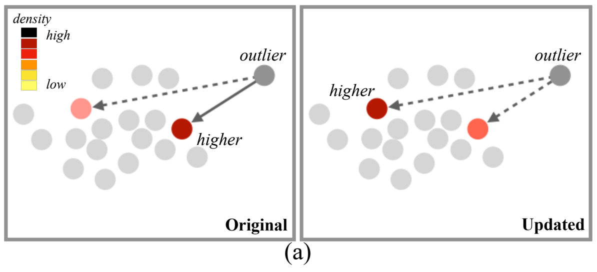
$$\boldsymbol{f}_i' = \sum_{j=0}^{k} s_{ij} \boldsymbol{f}_{ij}, \tag{1}$$

where $j = 0$ indicates the feature of the sample itself and s_{i0} is set to 1. This actually makes the intra-class features more compact and the inter-class samples more separable. However, considering samples spreading nearby the boundaries of different classes in feature space, the overladen attention to neighbors may bring errors to the classification, we thus concatenate the original features and the weighted-neighbor features to form the combined features, the pair features are finally determined as:

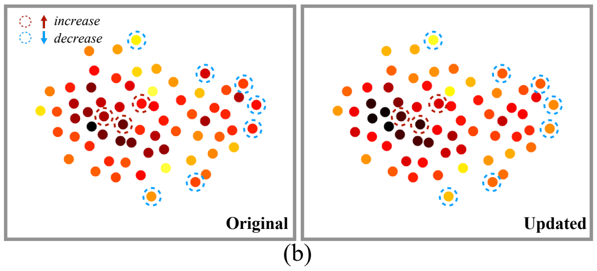
$$\boldsymbol{F}_{12} = [\boldsymbol{f}_1, \boldsymbol{f}_1', \boldsymbol{f}_2, \boldsymbol{f}_2']. \tag{2}$$

Compared with graph-based methods, by learning on the pair-level, the memory usage is largely reduced in both the training and inference stages, and we can flexibly change the batch size to fit in the memory capacity available.

Training set formulation. For convenience, we define the pairs with the same class as positive pairs and pairs with different classes as negative ones. To formulate the training set, for each face, we collect the set of faces of the same class and combine all these sets as the positive samples. For negative samples, we use k-NN to find the top k nearest neighbors of a face. Then we choose the neighbors with different classes as negative samples, which are naturally hard negative samples, for they are close enough in the feature space but belong to different classes. We vary the k value until the number of negative samples is almost equal to positive ones in the training set.



(a)



(b)

Figure 3. The visualization of how our proposed rank-weighted density outperforms the original density. (a) The updated rank-weighted density can ease the cluster-center-shift caused by outliers. (b) Feature distribution of a cluster from the MS-Celeb-1M dataset [1] on t-SNE. The relative orders of densities of border samples decrease and those of centroid samples increase.

3.2. Rank-weighted Density

It is time-consuming to predict the relationships of all pairs in the inference stage, for a test set of size N faces will result in a test pair set of size N^2, and inference results of such big test pairs set will cause insufficiency in the breadth-first search (BFS) process to obtain the final clusters. To enhance the efficiency of the clustering pipeline, we use density to select appropriate pairs, thus reducing the number of pairs sent to the classifier.

Intuitively, in a cluster, the *border* samples can be not quite similar to each other, but the *centroid* samples are very likely to be similar to almost all samples. So we try to pair a sample with the *centroid* sample. The original "density"[2, 24, 8] of a sample can measure how likely a sample to be cluster center or how close a sample is from the cluster center, and is defined as the sum of the similarities of the top k nearest samples with the target sample. The calculation of similarity has been introduced previously. For instance, given a sample, the similarities between its neighbors and itself can be calculated, and the density d_i of sample x_i is defined as:

$$d_i = \sum_{j=1}^{k} s_{ij}, \tag{3}$$

where *k* is the number of neighbors.

However, due to the existence of outliers which begets that samples may get a higher density but further to the cluster center compared to another sample, the original density can cause inferior density orders and generate inappropriate pairs. And the outliers can influence the density of samples even more greatly in dense cluster distribution. To depress the adverse effects of outliers, considering that they are often relatively distant neighbors of most samples, we update density in such manner:

$$d_i^{'} = \sum_{j=1}^{k} f(j) s_{ij}, \tag{4}$$

where $f(j)$ is a monotonically decreasing function, which indicates that the closer neighbors should weight more in the calculation of density. When our updated rank-weighted density disagrees with the original density, for instance, sample x_i has a higher updated density compared to sample x_j while lower in original density, that is because the samples not close with the target sample x_j are given less attention despite that they have high similarities with x_j. Figure 3 (a) illustrates how our updated density outperforms the original density. In original density, although x_i tends to be in a more dense situation and more likely to be close to the cluster center, the outlier helps x_j gain a higher density, thus cause the ***cluster-center-shift*** among samples. We present the feature distribution of a cluster sampled from the MS-Celeb-1M dataset [1] in Figure 3 (b). It can be clearly seen that the densities of both *border* samples and *centroid* samples are optimized. With our novel rank-weighted density, the likelihood of a sample being cluster center is more precise and less insensitive to outliers.

We now use our updated rank-weighted density to instruct the selection of pairs sent to the classifier in inference. For each sample, we select the first sample with a higher density among its *k* nearest neighbors and form a pair. The selected sample is closer to the cluster center yet still similar enough to share the same class with the target sample. If a sample has no neighbors with higher density, no neighbors would be selected. But it can be seen that such samples are very likely to be picked by other samples and form pairs. And we only need to test size of no more than N pairs sent to the classifier, which largely increases the efficiency in inference.

3.3. Complexity Analysis

The time complexity of our method mainly arises from the k-NN search. Thanks to the approximate nearest search proposed in [28], the time complexity is reduced to $\mathcal{O}(nlogn)$ and can be further accelerated by GPU [18]. Once the k-NN search is finished, as shown in Algorithm 1 the density calculation of each sample, the pair selection and the feature construction of all test samples are all $\mathcal{O}(n)$. The number of pairs selected sent to the classifier is no more than the number of the samples, so the time cost is also $\mathcal{O}(n)$. Consequently, the overall time complexity of our method is $\mathcal{O}(nlogn)$, and the time consumption mainly lies in the k-NN search.

Methods	F_P	F_B	Time
K-Means [3, 9]	79.21	81.23	11.5h
HAC [10]	70.63	70.46	12.7h
DBSCAN [2]	67.93	67.17	1.9m
ARO [11]	13.6	17	27.5m
CDP [6]	75.02	78.7	2.3m
L-GCN [4]	78.68	84.37	86.8m
LTC [5]	85.66	85.52	62.2m
GCN(V+E) [7]	87.93	86.09	11.5m
DANet [8]	90.6	–	5.5m
Ours	**90.67**	**89.54**	**1.7m**

Table 1. Comparison on MS-Celeb-1M.

Our method is also memory-friendly. The popular graph learning-based method takes $\mathcal{O}(ND)$ space complexity to learn on the whole graph, where N is the number of samples which can be up to millions and D the length of features. Commonly, $N \gg D$. In our proposed method, the inputs of the network are only concatenated feature vectors with lengths at D, and the large number N is divided into batches in training or inference, so our method reduces memory consumption by a very considerable margin. The details of our time and memory consumption will be revealed in Sec. 4.

4. Experiments

4.1. Experimental Settings

Datasets. We first evaluate the proposed method on two large public face clustering benchmarks, MS-Celb-1M[1] and IJB-B[31]. Following the widely used annotations from ArcFace [29], a reliable subset containing 5.8*M* images from 86*K* ids of MS-Cleb-1M is collected. The training and testing sets are divided following the settings as [7], where the subset was split into 10 parts almost equally. We train on one labeled part and select 1, 3, 5, 7 and all parts from the other 9 unlabeled parts, resulting in test sets with sizes of 584*K*, 1.74*M*, 2.89*M*, 4.05*M* and 5.21*M* images respectively. For IJB-B, we use the same settings as in [4], adopting a random subset of the CASIA dataset [30] which contains 5*K* ids and 200*K* samples as training set, and test on the three largest subtasks of IJB-B. The number of ids

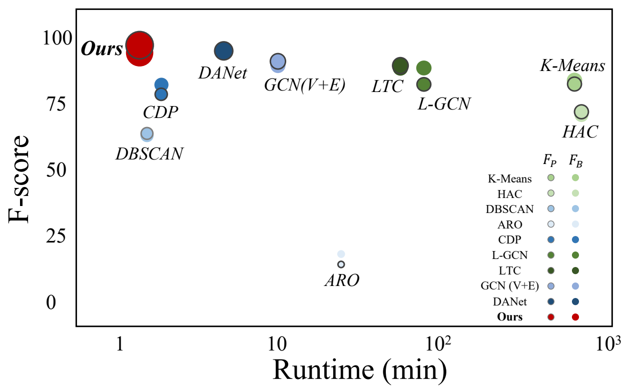


Figure 4. The trade-off between accuracy and efficiency of our method and comparison methods. Note that the runtime axis is in log-scale.

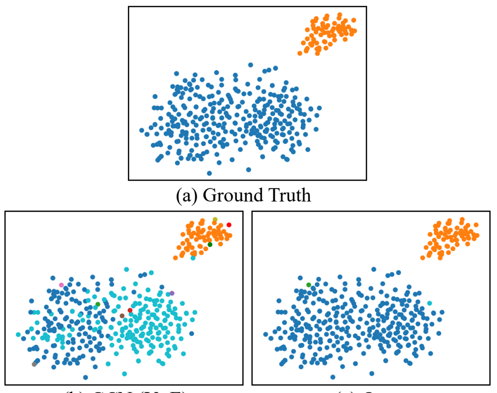


Figure 5. The feature distribution of two clusters of samples from the MS-Celeb-1M dataset on t-SNE. (a) The ground truth. Nodes with the same colors belong to the same class. (b) Result of GCN (V+E) [7]. (c) Result of our method. Our method predicts more accurately and generates fewer singleton clusters.

in the three subtasks are 512, 1,024 and 1,845, containing 18,171, 36,575 and 68,195 images respectively. We also evaluate our clustering approach on datasets other than face images to prove the generalization of our proposed method on clustering tasks. For a fair comparison, we use the same settings as in [7], where the training set with around 26K images from 4K categories and test set with around 27K images from 4K categories are sampled from the DeepFashion [17] benchmark.

Metrics. The performances of our proposed method and comparison methods are evaluated on the two popular clustering metrics [15], namely Pairwise F-score and BCubed F-score [16] respectively. Both metrics are calculated as the harmonic mean of precision and recall, and will be referred to as F_P and F_B in the following sections.

Implementation Details. To construct the neighbor features in pairwise relationship classifier and calculate the rank-weighted density, for each dataset, we use the same k in all k-NN scenes. The choice of k mainly depends on the sizes of clusters in the training set. Specifically, the k is set to 80, 40 and 5 for MS-Celeb-1M, IJB-B and DeepFashion respectively. The classifier is stacked with 3 fully connected layers. In training, we use SGD optimizer with start learning rate 0.01, momentum 0.9 and weight decay 5e-4. The batch size is set to 2048 and the training stops after 60 epochs. In the calculation of rank-weighted density, we use the power function as the weighting function:

$$f_i = (k - i)^p. \tag{5}$$

The setting of p will be further discussed in ablation studies.

4.2. Method Comparison

We compare our method with a series of face clustering baselines. These approaches can be generally categorized to conventional methods and learning based methods. The representative approaches in the former category include K-means [3, 9], Hierarchical Agglomerative Clustering (HAC) [10, 27], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [2], and Approximate Rank Order (ARO) [11]. The latter shows more promising results, including Consensus-Driven Propagation (CDP) [6], L-GCN [4], Learning to Cluster (LTC) [5], GCN (V+E) [7] and DANet [8], where the last two achieve the previously best performances on F_B and F_P respectively.

Results. We firstly test our method and other comparison methods on the first test part of the MS-Celeb-1M data, containing 580K images of 8,573 identities as in [6, 8]. The results presented in Table 1 compares the Pairwise F-score and BCubed F-score performances of different approaches on this set. Our method achieves state-of-the-art performance in terms of both metrics, surpassing all previous methods. Figure 5 visualizes two clusters of the test set on t-SNE, where we can see that our method performs better with generating fewer singleton clusters and clustering samples from a large cluster more accurately. By formulating face clustering as pairwise classification and learning on pair-level instead of graph-level, our method actually handles the face clustering task from a fundamental perspective and faces the ultimate question of clustering, *i.e.*, the homogeneity of several samples.

To demonstrate the robustness of our threshold-free method, the F_P and F_B performances of the state-of-the-art method GCN (V+E) [7] under different thresholds in inference are assessed. Figure 6 shows that performances of this method are quite sensitive to the threshold, which introduces inconsistency when applied to various real-world scenes. But our method demands no threshold and thus is a more robust method.

Number of images	584K		1.74M		2.89M		4.05M		5.21M	
Methods/ Metrics	F_P	F_B	F_P	F_B	F_P	F_B	F_P	F_B	F_P	F_B
K-Means [3, 9]	79.21	81.23	73.04	75.2	69.83	72.34	67.9	70.57	66.47	69.42
HAC [10]	70.63	70.46	54.4	69.53	11.08	68.62	1.4	67.69	0.37	66.96
DBSCAN [2]	67.93	67.17	63.41	66.53	52.5	66.26	45.24	44.87	44.94	44.74
ARO [11]	13.6	17	8.78	12.42	7.3	10.96	6.86	10.5	6.35	10.01
CDP [6]	75.02	78.7	70.75	75.82	69.51	74.58	68.62	73.62	68.06	72.92
L-GCN [4]	78.68	84.37	75.83	81.61	74.29	80.11	73.7	79.33	72.99	78.6
LTC [5]	85.66	85.52	82.41	83.01	80.32	81.1	78.98	79.84	77.87	78.86
GCN(V+E) [7]	87.93	86.09	84.04	82.84	82.1	81.24	80.45	80.09	79.3	79.25
Ours	**90.67**	**89.54**	**86.91**	**86.25**	**85.06**	**84.55**	**83.51**	**83.49**	**82.41**	**82.4**

Table 2. Comparison on face clustering with different numbers of unlabeled images from the MS-Celeb-1M dataset.

Methods	F_{512}	F_{1024}	F_{1845}
K-Means [3]	61.2	60.3	60.0
DBSCAN [2]	75.3	72.5	69.5
ARO [11]	76.3	75.8	75.5
L-GCN [4]	83.3	83.3	81.4
DANet [8]	83.4	83.3	**82.8**
Ours	**84.4**	**83.3**	82.7

Table 3. Comparison on IJB-B. F_{512}, F_{1024} and F_{1845} are Pairwise F-scores of different test sets.

Methods	Clusters	F_P	F_B	Time
K-Means [3]	3991	32.86	53.77	573s
HAC [10]	17410	22.54	48.77	112s
DBSCAN [2]	14350	25.07	53.23	2.2s
MeanShift [12]	8435	31.61	56.73	2.2h
Spectral [14]	2504	29.02	46.4	2.1h
ARO [11]	10504	26.03	53.01	6.7s
CDP [6]	6622	28.28	57.83	1.3s
L-GCN [4]	10137	28.85	58.91	23.3s
LTC [5]	9246	29.14	59.11	13.1s
GCN(V+E) [7]	6079	**38.47**	60.06	18.5s
Ours	**6018**	37.67	**62.17**	**0.6s**

Table 4. Comparison on DeepFashion.

In addition, we perform an experiment that compares the generalization of different methods on incremental numbers of unlabeled images. The results in Table 2 shows that our method can generalize better and achieve consistent improvements on larger test sets compared with other methods. It demonstrates that our proposed method can obtain more robust and excellent performances in large-scale scenes. Table 3 gives the results of our method and comparison methods on another large face clustering benchmark IJB-B. It can be seen that our method gains comparative or even better performances on the three largest subtasks of this dataset.

Table 4 shows that in other clustering benchmarks like DeepFashion, our method also gains satisfying performances by outperforming state-of-the-art methods in BCubed F-score with a 2.11% lead, which further reveals the generalization ability of our proposed method on clustering tasks.

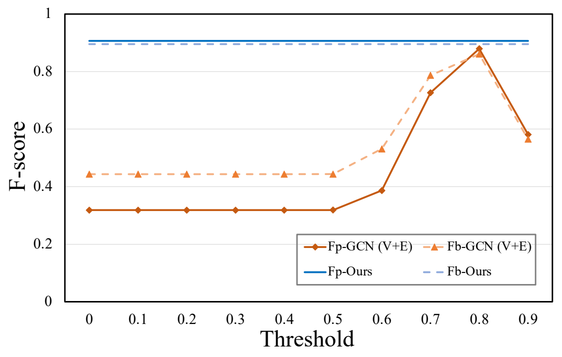


Figure 6. The performances of GCN (V+E) [7] under different thresholds in inference on MS-Celeb-1M dataset. The two lines on top are the F_P and F_B performances of our method.

Memory and Time Consumption. We run our experiments on a single Telsa P40 with 24G memory capacity. As has been discussed above, the graph learning on large-scale graphs could be very memory-consuming, thus become impractical in certain real-world scenes. Take GCN (V+E) [7] for example, to reduce memory consumption, GCN-V only uses one layer but still raises an out-of-memory error when the size of test set comes to 2.89M on MS-Celeb-1M dataset in inference. One can use mini-batch in inference stage to reduce memory consumption in graph-based methods. However, the mini-batch will cause the lost of the links between batches, which can influence the performance of the method. Besides, the different divisions of graph to mini-batches will bring different performances, which makes those methods unstable. Yet in our approach, since the inputs of the classifier are only 1-D features with batch input, we can apply our method in any size of datasets and flexibly set the batch size to fit our accessible calculation resources, even only on CPU. And the batch size of 2048

Feature Setting	Precision	Recall	Accuracy
original	92.2	83.2	84.7
weighted-neighbor	94.4	**97.9**	95.2
combined	**95.4**	97.5	**95.6**

Table 5. Performances of our classifier based on different features. The positives are sample pairs with the same class.

Power	Pre_P	Rec_P	F_P	Pre_B	Rec_B	F_B
0	88.04	89.1	88.57	91.63	85.19	88.3
0.5	89.16	89.12	89.14	92.33	85.37	88.71
1	90.26	**89.13**	89.69	92.91	**85.5**	89.05
3	92.11	88.88	90.47	93.84	85.44	89.45
5	**92.94**	88.5	**90.67**	**94.51**	85.06	**89.54**
7	92.84	87.93	90.32	94.33	84.72	89.27

Table 6. The influence of power on the MS-Celeb-1M dataset.

only needs 0.7G memory.

From Table 1, our method is also time-efficient, and actually the fastest one, even surpassing the conventional methods. For a fair comparison, we analyze the inference time of all supervised methods on MS-Celeb-1M with $N = 584K$ as in [7], and our method takes about $1.7m$ on a single GPU with batch size 2048. And the F_{512} subtask on IJB-B takes 23.5s. Figure 4 illustrates the trade-off between accuracy and efficiency. It shows that our method not only achieves state-of-the-art performance, but also enjoys a large advantage on time consumption. To conclude, our method achieves state-of-the-art performance at the fastest speed with very limited memory usage.

4.3. Ablation Study

We conduct the ablation study mainly on the MS-Celeb-1M dataset.

Design of input features. We explore different designs of input features sent to the classifier. As has been discussed in Sec. 3, given two features, the first thought would be a simple concatenation of the two original feature vectors, which we note as *original features*. To emphasize the importance of neighbor features, we design the similarity-weighted summation of neighbor features and note them as *weighted-neighbor features*. The concatenation of the original features and the weighted neighbor features are finally noted as *combined features*. We use exactly the same hyper-parameters to train the classifier based on different kinds of features for a clearer comparison.

As shown in Table 5, the performance of the classifier based on weighted-neighbor features largely outperforms that on original features, for the nearby neighbors are more likely to share the same identity with the sample, thus providing contextual information on categories, which finally boosts the performance of the classifier. The combined features show a further improvement than weighted-neighbor

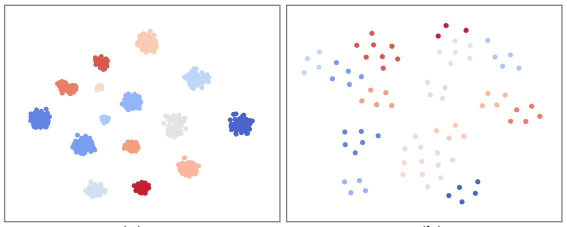


Figure 7. The feature distribution from different datasets on t-SNE. (a) The MS-Celeb-1M dataset. (b) The DeepFashion dataset.

features, for it gives more attention to the sample itself, which may reduce the possible influence of neighbors with different classes.

Design of rank-weighted density. To give more attention to nearer neighbors in the calculation of density and reduce the influence of outliers, a monotonically decreasing function is applied to add weights on the summation of similarities. We choose the simple power function $f(i) = (k - i)^p$ and vary the power to obtain the best results. Table 6 shows the influence of power on the MS-Celeb-1M dataset. The rank-weighted density turns into original density when power is set to 0. We can see that the designed rank-weighted density brings consistent performance gains and power 5 achieves the best balance between precision and recall. We believe the best choice of power on each dataset depends on the sparsity of the dataset itself. Figure 7 makes visualization of feature distribution from different datasets on t-SNE. On datasets like DeepFashion, the samples are distributed more evenly, and the influence of different choices of power is very limited. However, on large-scale datasets with more than $1M$ images like MS-Celeb-1M, the samples distribute very densely in feature space, the power should be higher to make the influences of neighbors with different distance ranks more distinguishable.

5. Conclusion

This paper has proposed a simple yet elegant face clustering framework based on pairwise classification. We adopt a classifier to determine the relationships between samples, which largely reduce the memory consumption by learning on the pair-level instead of graph-level, and also frees face clustering task from the manual setting of thresholds in inference. Besides, to further enhance the efficiency, we design a novel rank-weighted density to guide the selection of pairs sent to the classifier. Extensive experimental results on public benchmarks demonstrate that our method achieves excellent performances in terms of both accuracy and efficiency, and also generalizes well in larger test sets and other clustering tasks.

References

[1] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV*. Springer, 2016.

[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.

[3] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[4] Zhongdao Wang, Liang Zheng, Yali Li, and Shengjin Wang. Linkage based face clustering via graph convolution network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1117–1125, 2019.

[5] Lei Yang, Xiaohang Zhan, Dapeng Chen, Junjie Yan, Chen Change Loy, and Dahua Lin. Learning to cluster faces on an affinity graph. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2298–2306, 2019.

[6] Xiaohang Zhan, Ziwei Liu, Junjie Yan, Dahua Lin, and Chen Change Loy. Consensus-driven propagation in massive unlabeled data for face recognition. In *ECCV*, 2018.

[7] Lei Yang, Dapeng Chen, Xiaohang Zhan, Rui Zhao, Chen Change Loy, and Dahua Lin. Learning to Cluster Faces via Confidence and Connectivity Estimation. In *CVPR*, 2020.

[8] Senhui Guo, Jing Xu, Dapeng Chen, Chao Zhang, Xiaogang Wang, and Rui Zhao. Density-Aware Feature Embedding for Face Clustering. In *CVPR*, 2020.

[9] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[10] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

[11] Clustering millions of faces by identity. *TPAMI*, 40(2):289–303, 2018.

[12] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[13] Aaron Nech and Ira Kemelmacher-Shlizerman. Level playing field for million scale face recognition. *CVPR*, 2017.

[14] Jeffrey Ho, Ming-Hsuan Yang, Jongwoo Lim, Kuang-Chih Lee, and David Kriegman. Clustering appearances of objects under varying illumination conditions. In *CVPR*, 2003.

[15] Yichun Shi, Charles Otto, and Anil K Jain. Face clustering: representation and pairwise constraints. *IEEE Transactions on Information Forensics and Security*, 13(7):1626–1640, 2018.

[16] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486, 2009.

[17] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[19] Chunhui Zhu, Fang Wen, and Jian Sun. A rank-order distance based clustering algorithm for face tagging. *CVPR*, 2017.

[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[21] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[23] Xingcheng Zhang, Lei Yang, Junjie Yan, and Dahua Lin. Accelerated training for massive classification via dynamic class selection. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[25] Yue He, Kaidi Cao, Cheng Li, and Chen Change Loy. Merge or not? learning to group faces via imitation learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[26] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[27] Daniel Müllner et al. fastcluster: Fast hierarchical, agglomerative clustering routines for r and python. *Journal of Statistical Software*, 53(9):1–18, 2013.

[28] Patrick Wieschollek, Oliver Wang, Alexander Sorkine-Hornung, and Hendrik Lensch. Efficient large-scale approximate nearest neighbor search on the gpu. In *CVPR*, 2016.

[29] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *CVPR*, 2019.

[30] Dong Yi, Zhen Lei, Shengcai Liao and Stan Z. Li. Learning Face Representation from Scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[31] Cameron Whitelam, Emma Taborsky, Austin Blanton, Brianna Maze, Jocelyn Adams, Tim Miller, Nathan Kalka, Anil K Jain, James A Duncan, Kristen Allen, et al. Iarpa janus benchmark-b face dataset. *CVPR Workshops*, pages 592–600, 2017.