Learning to Cluster Faces on an Affinity Graph

Lei Yang,[1] Xiaohang Zhan,[1] Dapeng Chen,[2] Junjie Yan,[2] Chen Change Loy,[3] Dahua Lin,[1]
[1]CUHK - SenseTime Joint Lab, The Chinese University of Hong Kong
[2]SenseTime Group Limited, [3]Nanyang Technological University
{yl016, zx017, dhlin}@ie.cuhk.edu.hk, {chendapeng, yanjunjie}@sensetime.com, ccloy@ntu.edu.sg

Abstract

*Face recognition sees remarkable progress in recent years, and its performance has reached a very high level. Taking it to a next level requires substantially larger data, which would involve prohibitive annotation cost. Hence, exploiting unlabeled data becomes an appealing alternative. Recent works have shown that clustering unlabeled faces is a promising approach, often leading to notable performance gains. Yet, how to effectively cluster, especially on a large-scale (*i.e. *million-level or above) dataset, remains an open question. A key challenge lies in the complex variations of cluster patterns, which make it difficult for conventional clustering methods to meet the needed accuracy. This work explores a novel approach, namely, learning to cluster instead of relying on hand-crafted criteria. Specifically, we propose a framework based on graph convolutional network, which combines a detection and a segmentation module to pinpoint face clusters. Experiments show that our method yields significantly more accurate face clusters, which, as a result, also lead to further performance gain in face recognition.*

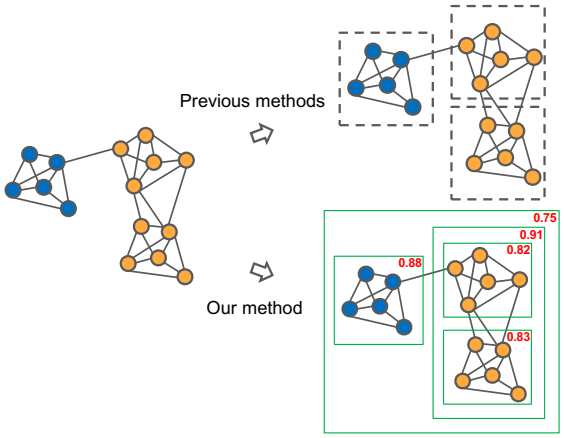


Figure 1: A case to demonstrate the difference between existing methods and our approach. The blue vertices and orange vertices represent two classes respectively. Previous unsupervised methods relying on specific clustering policies may not be able to handle the orange cluster with complex intra-structures. While our approach, through learning from the structures, is able to evaluate different combinations of cluster proposals (green boxes) and output the clusters with high scores.

1. Introduction

Thanks to the advances in deep learning techniques, the performance of face recognition has been remarkably boosted [25, 22, 27, 3, 31]. However, it should be noted that the high accuracy of modern face recognition systems relies heavily on the availability of large-scale annotated training data. While one can easily collect a vast quantity of facial images from the Internet, annotating them is prohibitively expensive. Therefore, exploiting unlabeled data, *e.g.* through unsupervised or semi-supervised learning, becomes a compelling option and has attracted lots of interest from both academia and industry [30, 1].

A natural idea to exploit unlabeled data is to cluster them into "pseudo classes", such that they can be used like labeled data and fed to a supervised learning pipeline. Recent works [30] have shown that this approach can bring performance gains. Yet, current implementations of this approach still leave a lot to be desired. Particularly, they often resort to unsupervised methods, such as K-means [19], spectral clustering [11], hierarchical clustering [32], and approximate rank-order [1], to group unlabeled faces. These methods rely on simplistic assumptions, *e.g.*, K-means implicitly assumes that the samples in each cluster are around a single center; spectral clustering requires that the cluster sizes are relatively balanced, etc. Consequently, they lack the capability of coping with complicated cluster structures, thus often giving rise to noisy clusters, especially when applied to large-scale datasets collected from real-world settings. This problem seriously limits the performance improvement.

Hence, to effectively exploit unlabeled face data, we need to develop an effective clustering algorithm that is able to cope with the complicated cluster structures arising fre-

quently in practice. Clearly, relying on simple assumptions would not provide this capability. In this work, we explore a fundamentally different approach, that is, to *learn* how to cluster from data. Particularly, we desire to draw on the strong expressive power of graph convolutional network to capture the common patterns in face clusters, and leverage them to help to partition the unlabeled data.

We propose a framework for face clustering based on graph convolutional networks [15]. This framework adopts a pipeline similar to the Mask R-CNN [10] for instance segmentation, *i.e.*, generating proposals, identifying the positive ones, and then refining them with masks. These steps are accomplished respectively by an iterative proposal generator based on super-vertex, a graph detection network, and a graph segmentation network. It should be noted that while we are inspired by Mask R-CNN, our framework still differs essentially: the former operates on a 2D image grid while the latter operates on an affinity graph with arbitrary structures. As shown in Figure 1, relying on the structural patterns learned based on a graph convolutional network instead of some simplistic assumptions, our framework is able to handle clusters with complicated structures.

The proposed method significantly improves the clustering accuracy on large-scale face data, achieving a F-score at 85.66, which is not only superior to the best result obtained by unsupervised clustering methods (F-score 68.39) but also higher than a recent state of the art [30] (F-score 75.01). Using this clustering framework to process the unlabeled data, we improve the performance of a face recognition model on MegaFace from 60.29 to 78.64, which is quite close to the performance obtained by supervised learning on all the data (80.75).

The main contributions lie in three aspects: (1) We make the first attempt to perform top-down face clustering in a supervised manner. (2) It is the first work that formulates clustering as a detection and segmentation pipeline based on graph convolution networks. (3) Our method achieves state-of-the-art performance in large-scale face clustering, and boosts the face recognition model close to the supervised result when applying the discovered clusters.

2. Related Work

Face Clustering Clustering is a basic task in machine learning. Jain *et al*. [12] provide a survey for classical clustering methods. Most existing clustering methods are unsupervised. Face clustering provides a way to exploit massive unlabeled data. The study along this direction remains at an early stage. The question of how to cluster faces on large-scale data remains open.

Early works use hand-crafted features and classical clustering algorithms. For example, Ho *et al*. [11] used gradient and pixel intensity as face features. Cui *et al*. [2] used LBP features. Both of them adopt spectral clustering. Recent methods make use of learned features. [13] performed top-down clustering in an unsupervised way. Finley *et al*. [5] proposed an SVM-based supervised method in a bottom-up manner. Otto *et al*. [1] used deep features from a CNN-based face model and proposed an approximate rank-order metric to link images pairs to be clusters. Lin *et al*. [18] designed a similarity measure based on linear SVM trained on the nearest neighbours of data samples. Shi *et al*. [23] proposed Conditional Pairwise Clustering, formulating clustering as a conditional random field to cluster faces by pairwise similarities. Lin *et al*. [17] proposed to exploit local structures of deep features by introducing minimal covering spheres of neighbourhoods to improve similarity measure. Zhan *et al*. [30] trained a MLP classifier to aggregate information and thus discover more robust linkages, then obtained clusters by finding connected components.

Though using deep features, these works mainly concentrate on designing new similarity metrics, and still rely on unsupervised methods to perform clustering. Unlike all the works above, our method *learns* how to cluster in a top-down manner, based on a detection-segmentation paradigm. This allows the model to handle clusters with complicated structures.

Graph Convolutional Networks Graph Convolutional Networks (GCNs) [15] extend CNNs to process graph-structured data. Existing work has shown the advantages of GCNs, such as the strong capability of modeling complex graphical patterns. On various tasks, the use of GCNs has led to considerable performance improvement [15, 9, 26, 29]. For example, Kipf *et al*. [15] applied the GCNs to semi-supervised classification. Hamilton *et al*. [9] leveraged GCNs to learn feature representations. Berg *et al*. [26] showed that GCNs are superior to other methods in link prediction. Yan *et al*. [29] employed GCNs to model human joints for skeleton-based action recognition.

In this paper, we adopt GCN as the basic machinery to capture cluster patterns on an affinity graph. To our best knowledge, this is the first work that uses GCN to learn how to cluster in a supervised way.

3. Methodology

In large-scale face clustering, the complex variations of the cluster patterns become the main challenge for further performance gain. To tackle the challenge, we explore a supervised approach, that is, to learn the cluster patterns based on graph convolutional networks. Specifically, we formulate this as a joint detection and segmentation problem on an affinity graph.

Given a face dataset, we extract the feature for each face image with a trained CNN, forming a set of features $\mathcal{D} = \{\mathbf{f}_i\}_{i=1}^{N}$, where $\mathbf{f}_i$ is a d-dimensional vector. To construct the affinity graph, we regard each sample as a vertex

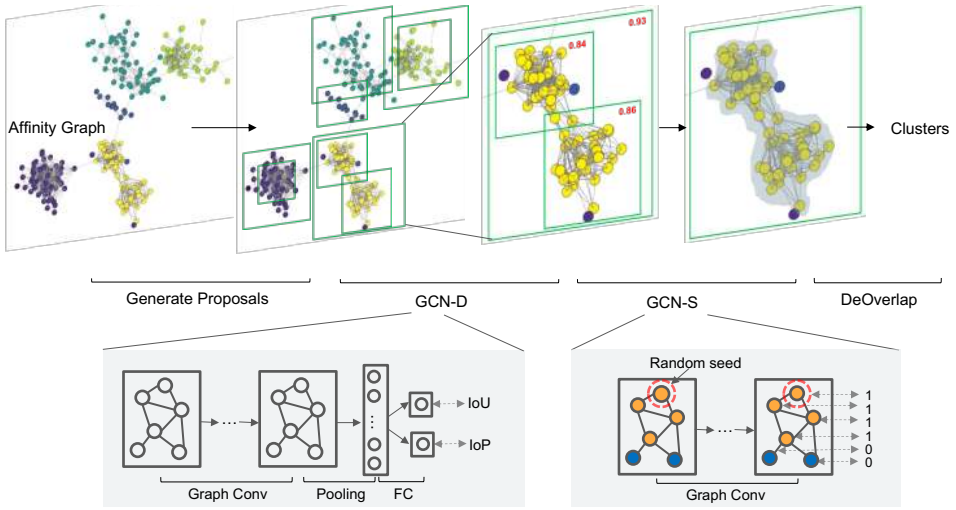


Figure 2: Overview of graph detection and segmentation for clustering.

and use cosine similarity to find K nearest neighbors for each sample. By connecting between neighbors, we obtain an affinity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for the whole dataset. Alternatively, the affinity graph $\mathcal{G}$ can also be represented by a symmetric adjacent matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where the element $a_{i,j}$ is the cosine similarity between $\mathbf{f}_i$ and $\mathbf{f}_j$ if two vertices are connected, or zero otherwise. The affinity graph is a large-scale graph with millions of vertices. From such a graph, we desire to find clusters that have the following properties: (1) different clusters contain the images with different labels; and (2) images in one cluster are with the same label.

3.1. Framework Overview

As shown in Figure 2, our clustering framework consists of three modules, namely proposal generator, GCN-D, and GCN-S. The first module generates cluster proposals, *i.e.*, sub-graphs likely to be clusters, from the affinity graph. With all the cluster proposals, we then introduce two GCN modules, GCN-D and GCN-S, to form a two-stage procedure, which first selects high-quality proposals and then refines the selected proposals by removing the noises therein. Specifically, GCN-D performs cluster detection. Taking a cluster proposal as input, it evaluates how likely the proposal constitutes a desired cluster. Then GCN-S performs the segmentation to refine the selected proposals. Particularly, given a cluster, it estimates the probability of being noise for each vertex, and prunes the cluster by discarding the outliers. According to the outputs of these two GCNs, we can efficiently obtain high-quality clusters.

3.2. Cluster Proposals

Instead of processing the large affinity graph directly, we first generate cluster proposals. It is inspired by the way of generating region proposals in object detection [7, 6]. Such a strategy can substantially reduce the computational cost, since in this way, only a limited number of cluster candidates need to be evaluated. A cluster proposal $\mathcal{P}_i$ is a sub-graph of the affinity graph $\mathcal{G}$. All the proposals compose a set $\mathcal{P} = \{\mathcal{P}_i\}_{i=1}^{N_p}$. The cluster proposals are generated based on super-vertices, and all the super-vertices form a set $\mathcal{S} = \{\mathcal{S}_i\}_{i=1}^{N_s}$. In this section, we first introduce the generation of super-vertex, and then devise an algorithm to compose cluster proposals thereon.

Super-Vertex. A super-vertex is a sub-graph containing a small number of vertices that are closely connected to each other. Hence, it is natural to use connected components to represent super-vertex. However, the connected component directly derived from the graph $\mathcal{G}$ can be overly large. To maintain high connectivity within each super-vertice, we remove those edges whose affinity values are below a threshold e_τ and constrain the size of super-vertices below a maximum s_{max}. Alg. 1 shows the detailed procedure to produce the super-vertex set $\mathcal{S}$. Generally, an affinity graph with $1M$ vertices can be partitioned into $50K$ super-vertices, with each containing 20 vertices on average.

Proposal Generation. Compared with the desired clusters, the super-vertex is a conservative formation. Although the vertices in a super-vertex are highly possible to describe the same person, the samples of a person may distribute into several super-vertices. Inspired by the multi-scale proposals in object detection [7, 6], we design an algorithm to generate multi-scale cluster proposals. As Alg. 2 shows, we construct a higher-level graph on top of the super-vertices, with the centers of super-vertices as the vertices and the affinities between these centers as the edges. With this higher-level graph, we can apply Alg. 1 again and obtain proposals of larger sizes. By iteratively applying this construction for I times, we obtain proposals with multiple scales.

Algorithm 1 Super-vertex Generation

Input: Affinity Graph $\mathbf{A}$, edge threshold e_τ, maximum size s_{max}, threshold step Δ.
Output: Super-Vertices $\mathcal{S}$

$\mathcal{S} = \emptyset, \mathcal{R} = \emptyset$
$\mathcal{C}, \mathcal{R}$ = FINDSUPERVERTICES($\mathbf{A}$, e_τ, s_{max})
$\mathcal{S} = \mathcal{S} \cup \mathcal{C}$
while $\mathcal{R} \neq \emptyset$ **do**
    $e_\tau = e_\tau + \Delta$
    $\mathcal{C}, \mathcal{R}$ = FINDSUPERVERTICES($\mathbf{A}_\mathcal{R}$, e_τ, s_{max})
    $\mathcal{S} = \mathcal{S} \cup \mathcal{C}$
end while
return $\mathcal{S}$

function FINDSUPERVERTICES($\mathbf{A}$, e_τ, s_{max})
    $\mathbf{A}'$ = PRUNEEDGE($\mathbf{A}$, e_τ)
    $\mathcal{X}$ = FINDCONNECTEDCOMPONENTS($\mathbf{A}'$)
    $\mathcal{C} = \{c | c \in \mathcal{X}, |c| < s_{max}\}$
    $\mathcal{R} = \mathcal{X} \setminus \mathcal{C}$
    return $\mathcal{C}, \mathcal{R}$
end function

Algorithm 2 Iterative Proposal Generation

Input: Super-Vertex set $\mathcal{S}$, Iterative Number I, edge threshold e_τ, maximum size s_{max}, threshold step Δ.
Output: Proposal set $\mathcal{P}$

$\mathcal{P} = \emptyset, i = 0, \mathcal{S}' = \mathcal{S}$
while $i < I$ **do**
    $\mathcal{P} = \mathcal{P} \cup \mathcal{S}'$
    $\mathcal{D} = \{f_s | s \in \mathcal{S}'\}$, where f_s is the average feature of the vertices in s.
    $\mathbf{A}$ = BUILDAFFINITYGRAPH($\mathcal{D}$)
    $\mathcal{S}'$ = ALGORITHM 1($\mathbf{A}, e_\tau, s_{max}, \Delta$)
    $i = i + 1$
end while
return $\mathcal{P}$

3.3. Cluster Detection

We devise *GCN-D*, a module based on a graph convolutional network (GCN), to select high-quality clusters from the generated cluster proposals. Here, the quality is measured by two metrics, namely *IoU* and *IoP* scores. Given a cluster proposal $\mathcal{P}$, these scores are defined as

$$IoU(\mathcal{P}) = \frac{|\mathcal{P} \cap \widehat{\mathcal{P}}|}{|\mathcal{P} \cup \widehat{\mathcal{P}}|}, \quad IoP(\mathcal{P}) = \frac{|\mathcal{P} \cap \widehat{\mathcal{P}}|}{|\mathcal{P}|}, \qquad (1)$$

where $\widehat{\mathcal{P}}$ is the ground-truth set comprised all the vertices with label $l(\mathcal{P})$, and $l(\mathcal{P})$ is the *majority label* of the cluster $\mathcal{P}$, *i.e.* the label that occurs the most in $\mathcal{P}$. Intuitively, *IoU* reflects how close $\mathcal{P}$ is to the desired ground-truth $\widehat{\mathcal{P}}$; while *IoP* reflects the purity, *i.e.* the proportion of vertices in $\mathcal{P}$ that are with the majority label $l(\mathcal{P})$.

Design of GCN-D. We assume that high quality clusters usually exhibit certain structural patterns among the vertices. We introduce a GCN to identify such clusters. Specifically, given a cluster proposal $\mathcal{P}_i$, the GCN takes the visual features associated with its vertices (denoted as $\mathbf{F}_0(\mathcal{P}_i)$) and the affinity sub-matrix (denoted as $\mathbf{A}(\mathcal{P}_i)$) as input, and predicts both the IoU and IoP scores.

The GCN networks consist of L layers and the computation of each layer can be formulated as:

$$\mathbf{F}_{l+1}(\mathcal{P}_i) = \sigma\left(\tilde{\mathbf{D}}(\mathcal{P}_i)^{-1}(\mathbf{A}(\mathcal{P}_i) + \mathbf{I})\mathbf{F}_l(\mathcal{P}_i)\mathbf{W}_l\right), \qquad (2)$$

where $\tilde{\mathbf{D}} = \sum_j \tilde{\mathbf{A}}_{ij}(\mathcal{P}_i)$ is a diagonal degree matrix. $\mathbf{F}_l(\mathcal{P}_i)$ contains the embeddings of the l-th layer. $\mathbf{W}_l$ is a matrix to transform the embeddings and σ is the non-linear activation function (*ReLU* is chosen in this work). Intuitively, this formula expresses a procedure of taking weighted average of the embedded features of each vertex and its neighbors, transforming them with $\mathbf{W}_l$, and then feeding them through a nonlinear activation. This is similar to a typical block in CNN, except that it operates on a graph with arbitrary topology. On the top-level embeddings $\mathbf{F}_L(\mathcal{P}_i)$, we apply a max pooling over all the vertices in $\mathcal{P}_i$, and obtain a feature vector that provides an overall summary. Two fully-connected layers are then employed to predict the IoU and IoP scores, respectively.

Training and Inference. Given a training set with class labels, we can obtain the ground-truth IoU and IoP scores following Eq.(1) for each cluster proposal $\mathcal{P}_i$. Then we train the GCN-D module, with the objective to minimize the *mean square error(MSE)* between ground-truth and predicted scores. We experimentally show that, without any fancy techniques, GCN can give accurate prediction. During inference, we use the trained GCN-D to predict both the IoU and IoP scores for each proposal. The IoU scores will be used in sec. 3.5 to first retain proposals with high IoU. The IoP scores will be used in the next stage to determine whether a proposal needs to be refined.

3.4. Cluster Segmentation

The top proposals identified by GCN-D may not be completely pure. These proposals may still contain a few *outliers*, which need to be eliminated. To this end, we develop a cluster segmentation module, named *GCN-S*, to exclude the outliers from the proposal.

Design of GCN-S. The structure of *GCN-S* is similar to that of *GCN-D*. The differences mainly lie in the values to be predicted. Instead of predicting quality scores of an entire cluster $\mathcal{P}$, GCN-S outputs a probability value for each vertex v to indicate how likely it is a genuine member instead of an outlier.

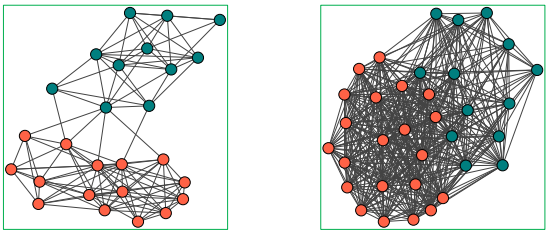


Figure 3: This figure shows two noisy proposals. Vertices of the same color belong to the same ground-truth class. The number of red vertices is slightly larger than the number of green vertices. As outliers are defined with respect to the proposal, *i.e.*, not absolute outliers in a proposal, both red and green segmentation results are correct. (Best viewed in color.)

Identifying Outliers To train the GCN-S, we need to prepare the ground-truth, *i.e.* identifying the outliers. A natural way is to treat all the vertices whose labels are different from the majority label as outliers. However, as shown in Fig. 3, this way may encounter difficulties for a proposal that contains an almost equal number of vertices with different labels. To avoid overfitting to manually defined outliers, we encourage the model to learn different segmentation patterns. As long as the segmentation result contains vertices from one class, no matter it is majority label or not, it is regarded as a reasonable solution. Specifically, we randomly select a vertex in the proposal as the seed. We concat a value to each vertex feature, where the value of the selected seed is one while others is zero. The vertices that have the same label with the seed are regarded as the positive vertices while others are considered as outliers. We apply this scheme multiple times with randomly chosen seeds and thus acquire multiple training samples from each proposal $\mathcal{P}$.

Training and Inference. With the process above, we can prepare a set of training samples from the retained proposals. Each sample contains a set of feature vectors, each for a vertex, an affinity matrix, as well as a binary vector to indicate whether the vertices are positive or not. Then we train the GCN-S module, using the vertex-wise binary cross-entropy as the loss function. During inference, we also draw multiple hypotheses for a generated cluster proposal, and only keep the predicted results that have the most positive vertices (with a threshold of 0.5). This strategy avoids being misled by the case where a vertex associated with very few positive counterparts is chosen as the seed.

We only feed the proposals with *IoP* between 0.3 and 0.7 to GCN-S. Because when the proposal is very pure, the outliers are usually hard examples that need not be removed. When the proposal is very impure, it is probable that none of the classes dominate, therefore the proposal might not be suitable to be processed by GCN-S. With the GCN-S predictions, we remove the outliers from the proposals.

Algorithm 3 De-overlapping

Input: Ranked Cluster Proposals $\{\mathcal{P}'_0, \mathcal{P}'_1, ..., \mathcal{P}'_{N_p-1}\}$
Output: Final Clusters $\mathcal{C}$

```
Cluster set C = ∅, Image set I = ∅, i = 1,
while i ≤ N_p do
    C_i = P'_i \ I
    C = C ∪ {C_i}
    I = I ∪ C_i
    i = i + 1
end while
return C
```

3.5. De-Overlapping

The three stages described above result in a collection of clusters. However, it is still possible that different clusters may overlap, *i.e.* sharing certain vertices. This may cause an adverse effect to the face recognition training performed thereon. Here, we propose a simple and fast *de-overlapping* algorithm to tackle this problem. Specifically, we first rank the cluster proposals in descending order of IoU scores. We sequentially collect the proposals from the ranked list, and modify each proposal by removing the vertices seen in preceding ones. The detailed algorithm is described in Alg. 3.

Compared to the Non-Maximum Suppression (NMS) in object detection, the de-overlapping method is more efficient. Particularly, the former has a complexity of $O(N^2)$, while the latter has $O(N)$. This process can be further accelerated by setting a threshold of IoU for de-overlapping.

4. Experiments

4.1. Experimental Settings

Training set. MS-Celeb-1M [8] is a large-scale face recognition dataset consists of $100K$ identities, and each identity has about 100 facial images. As the original identity labels are obtained automatically from webpages and thus are very noisy. We clean the labels based on the annotations from ArcFace [3], yielding a reliable subset that contains $5.8M$ images from $86K$ classes. The cleaned dataset is randomly split into 10 parts with an almost equal number of identities. Each part contains $8.6K$ identities with around $580K$ images. We randomly select 1 part as labeled data and the other 9 parts as unlabeled data. Youtube Face Dataset [28] contains $3,425$ videos, from which we extract $155,882$ frames for evaluation. Particularly, we use $14,653$ frames with 159 identities for training and the other $140,629$ images with $1,436$ identities for testing.

Testing set. MegaFace [14] is the largest public benchmark for face recognition. It includes a probe set from FaceScrub [21] with $3,530$ images and a gallery set containing $1M$ images. IJB-A [16] is another face recognition benchmark containing $5,712$ images from 500 identities.

Methods	#clusters	Precision	Recall	F-score	Time
K-Means [19]	5000	52.52	70.45	60.18	13h
DBSCAN [4]	352385	72.88	42.46	53.5	**100s**
HAC [24]	117392	66.84	70.01	68.39	18h
Approximate Rank Order [1]	307265	81.1	7.3	13.34	250s
CDP [30]	29658	80.19	70.47	75.01	350s
GCN-D	19879	95.72	76.42	84.99	2000s
GCN-D + GCN-S	19879	98.24	75.93	**85.66**	2200s

Table 1: Comparison on face clustering. (MS-Celeb-1M)

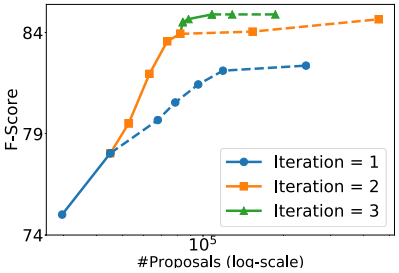


Figure 4: Proposal strategies.

Method	#clusters	Precision	Recall	F-score	Time
K-means	1000	60.65	59.6	60.12	39min
HAC	3621	99.64	87.31	93.07	1h
CDP	3081	98.32	89.84	93.89	175s
Ours	2369	96.75	92.27	**94.46**	537s

Table 2: Comparison on face clustering. (YTF)

Metrics. We assess the performance on two tasks, namely *face clustering* and *face recognition*. Face clustering is to cluster all the images of the same identity into a cluster, where the performance is measured by *pairwise* recall and *pairwise* precision. To consider both precision and recall, we report the widely used *F-score*, *i.e.*, the harmonic mean of precision and recall. Face recognition is evaluated with *face identification* benchmark in MegaFace and *face verification* protocol of IJB-A. We adopt top-1 identification hit rate in MegaFace, which is to rank the top-1 image from the $1M$ gallery images and compute the top-1 hit rate. For IJB-A, we adopt the protocol of face verification, which is to determine whether two given face images are from the same identity. We use *true positive rate* under the condition that the *false positive rate* is 0.001 for evaluation.

Implementation Details. We use GCN with two hidden layers in our experiments. The momentum SGD is used with a start learning rate 0.01. Proposals are generated by $e_\tau \in \{0.6, 0.65, 0.7, 0.75\}$ and $s_{max} = 300$ as in Alg. 1.

4.2. Method Comparison

4.2.1 Face Clustering

We compare the proposed method with a series of clustering baselines. These methods are briefly described below.

(1) K-means [19], the most commonly used clustering algorithm. With a given number of clusters k, K-means minimizes the total intra-cluster variance.

(2) DBSCAN [4], a density-based clustering algorithm. It extracts clusters according to a designed density criterion and leaves the sparse background as noises.

(3) HAC [24], hierarchical agglomerative clustering is a bottom-up approach to iteratively merge close clusters based on some criteria.

(4) Approximate Rank Order [1], develops an algorithm as a form of HAC. It only performs one iteration of clustering with a modified distance measure.

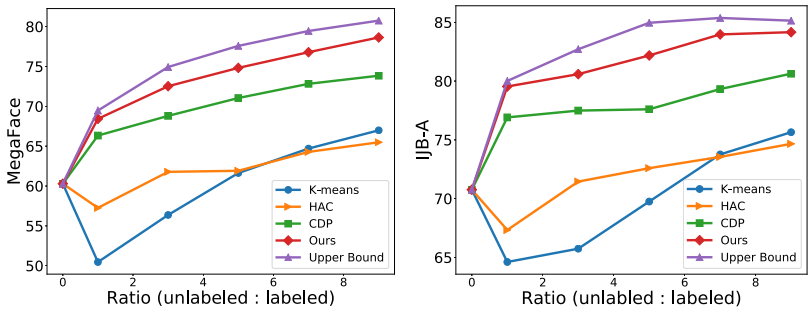
(5) CDP [30], a recent work that proposes a graph-based clustering algorithm. It better exploits the pairwise relationship in a bottom-up manner.

(6) GCN-D, the first module of the proposed method. It applies a GCN to learn cluster pattern in a supervised way.

(7) GCN-D + GCN-S, the two-stage version of the proposed method. GCN-S is introduced to refine the output of GCN-D, which detects and discards noises inside clusters.

Results To control the experimental time, we randomly select one part of the data for evaluation, containing $580K$ images of $8,573$ identities. Tab. 1 compares the performance of different methods on this set. The clustering performance is evaluated by both F-score and the time cost. We also report the number of clusters, pairwise precision and pairwise recall for better understanding the advantages and disadvantages of each method.

The results show: (1) For K-means, the performance is influenced greatly by the number of clusters k. We vary k in a range of numbers and report the result with high F-score. (2) DBSCAN reaches a high precision but suffers from the low recall. It may fail to deal with large density differences in large-scale face clustering. (3) HAC gives more robust results than previous methods. Note that the standard algorithm consumes $O(N^2)$ memory, which goes beyond the memory capacity when N is as large as $580K$. We use an adapted hierarchical clustering [20] for comparison, which requires only $O(Nd)$ memory. (4) Approximate Rank Order is very efficient due to its one iteration design, but the performance is inferior to other methods in our setting. (5) As a recent work designed to exploit unlabeled data for face recognition, CDP achieves a good balance of precision and recall. For a fair comparison, we compare with the single model version of CDP. Note that the idea of CDP and our approach are complementary, which can be combined to further improve the performance. (6) Our method applies GCN to learn cluster patterns. It improves the precision and recall simultaneously. Tab. 2 demonstrates that our method is robust and can be applied to datasets with different distri-



(a) MegaFace top-1 Identification@1M (b) IJB-A TPR@FPR=0.001

Figure 5: Comparison on face recognition. The performance is measured by MegaFace Identification and IJB-A Verification. The leftmost point is the performance obtained when only labeled data are used; the rightmost point is MegaFace performance obtained when both labeled and unlabeled data are used.

butions. Since the GCN is trained using multi-scale cluster proposals, it may better capture the properties of the desired clusters. As shown in Fig. 8, our method is capable of pinpointing some clusters with complex structure. (7) The GCN-S module further refines the cluster proposals from the first stage. It improves the precision by sacrificing a little recall, resulting in the overall performance gain.

Runtime Analysis The whole procedure of our method takes about $2200s$, where generating $150K$ proposals takes up to $1000s$ on a CPU and the inference of GCN-D and GCN-S takes $1000s$ and $200s$ respectively on a GPU with the batch size of 32. To compare the runtime fairly, we also test all our modules on CPU. Our method takes $3700s$ in total on CPU, which is still faster than most methods we compared. The speed gain of using GPU is not very significant in this work, as the main computing cost is on GCN. Since GCN relies on sparse matrix multiplication, it cannot make full use of GPU parallelism. The runtime of our method grows linearly with the number of unlabeled data and the process can be further accelerated by increasing batch size or parallelizing with more GPUs.

4.2.2 Face Recognition

With the trained clustering model, we apply it to unlabeled data to obtain pseudo labels. We investigate how the unlabeled data with pseudo labels enhance the performance of face recognition. Particularly, we follow the following steps to train face recognition models: (1) train the initial recognition model with labeled data in a supervised way; (2) train the clustering model on the labeled set, using the feature representation derived from the initial model; (3) apply the clustering model to group unlabeled data with various amounts (1, 3, 5, 7, 9 parts), and thus attach to them *"pseudo-labels"*; and (4) train the final recognition model using the whole dataset, with both original labeled data and the others with assigned pseudo-labels. The model trained only on the 1 part labeled data is regarded as the lower bound, while the model supervised by all the parts with ground-truth labels serves as the upper bound in our problem. For all clustering methods, each unlabeled image belongs to an unique cluster after clustering. We assign a pseudo label to each image as its cluster id.

Fig. 5 indicates that performance of face clustering is crucial for improving face recognition. For K-means and HAC, although the recall is good, the low precision indicates noisy predicted clusters. When the ratio of unlabeled and labeled data is small, the noisy clusters severely impair face recognition training. As the ratio of unlabeled and labeled data increases, the gain brought by the increase of unlabeled data alleviates the influence of noise. However, the overall improvement is limited. Both CDP and our approach benefit from the increase of the unlabeled data. Owing to the performance gain in clustering, our approach outperforms CDP consistently and improve the performance of face recognition model on MegaFace from 60.29 to 78.64, which is close to the fully supervised upper bound (80.75).

4.3. Ablation Study

We randomly select one part of the unlabeled data, containing $580K$ images of $8,573$ identities, to study some important design choices in our framework.

4.3.1 Proposal Strategies

Cluster proposals generation is the fundamental module in our framework. With a fixed $K = 80$ and different I, e_τ and s_{max}, we generate a large number of proposals with multiple scales. Generally, a larger number of proposals result in a better clustering performance. There is a trade-off between performance and computational cost in choosing the proper number of proposals. As illustrated in Fig. 4, each point represents the F-score under certain number of proposals. Different colors imply different iteration steps. (1) When $I = 1$, only the super-vertices generated by Alg. 1 will be used. By choosing different e_τ, more proposals are

Method	Channels	Pooling	Vertex Feature	F-score
a	128, 32	mean	✓	76.97
b	128, 32	sum	✓	53.75
c	128, 32	max	✓	83.61
d	128, 32	max	×	73.06
e	256, 64	max	✓	84.16
f	256, 128, 64	max	✓	77.95

Table 3: Design choice of GCN-D

obtained to increase the F-score. The performance gradually saturates as the number increases beyond $100K$. (2) When $I = 2$, different combinations of super-vertices are added to the proposals. Recall that it leverages the similarity between super-vertices, thus it enlarges the receptive field of the proposals effectively. With a small number of proposals added, it boosts the F-score by 5%. (3) When $I = 3$, it further merges similar proposals from previous stages to create proposals with larger scales, which continues to contribute the performance gain. However, with the increasing proposal scales, more noises will be introduced to the proposals, hence the performance gain saturates.

4.3.2 Design choice of GCN-D

Although the training of GCNs does not require any fancy techniques, there are some important design choices. As Tabs. 3a, 3b and 3c indicate, the pooling method has large influence on the F-score. Both mean pooling and sum pooling impair the clustering results compared with max pooling. For sum pooling, it is sensitive to the number of vertices, which tends to produce large proposals. Large proposals result in a high recall(80.55) but low precision (40.33), ending up with a low F-score. On the other hand, mean pooling better describes the graph structures, but may suffer from the outliers in the proposal. Besides the pooling methods, Tabs. 3c and 3d show that lacking vertex feature will significantly reduce the GCNs' prediction accuracy. It demonstrates the necessity of leveraging both vertex feature and graph structure during GCN training. In addition, as shown in Tabs. 3c, 3e and 3f, widening the channels of GCNs can increase its expression power but the deeper network may drive the hidden feature of vertices to be similar, resulting in an effect like mean pooling.

4.3.3 GCN-S

In our framework, GCN-S is used as a de-nosing module after GCN-D. However, it can act as an independent module to combine with previous methods. Given the clustering results of K-means, HAC and CDP, we regard them as the cluster proposals and feed them into the GCN-S. As Fig. 6 shows, GCN-S can improve their clustering performances by discarding the outliers inside clusters, obtaining a performance gain around $2\% - 5\%$ for various methods.

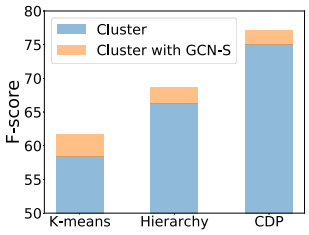


Figure 6: GCN-S

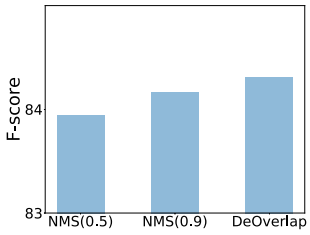


Figure 7: Post-processing

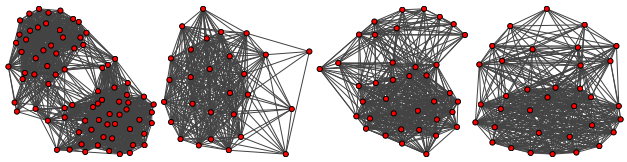


Figure 8: The figure shows 4 clusters pinpointed by our approach. All vertices in each cluster belong to the same class according to ground-truth annotation. The distance between vertices is inversely proportional to the similarity between vertices. It shows that our method can handle clusters with complex intra-structure, *e.g.* clusters with two sub-graphs inside, clusters with both dense and sparse connections.

4.3.4 Post-process strategies

NMS is a widely used post-processing technique in object detection, which can be an alternative choice of de-overlapping. With a different threshold of IoU, it keeps the proposal with highest predicted IoU while suppressing other overlapped proposals. The computational complexity of NMS is $O(N^2)$. Compared with NMS, de-overlapping does not suppress other proposals and thus retains more samples, which increases the clustering recall. As shown in Fig. 7, de-overlapping achieves better clustering performance and can be computed in linear time.

5. Conclusions

This paper proposes a novel supervised face clustering framework based on graph convolution network. Particularly, we formulate clustering as a detection and segmentation paradigm on an affinity graph. The proposed method outperforms previous methods on face clustering by a large margin, which consequently boosts the face recognition performance close to the supervised result. Extensive analysis further demonstrate the effectiveness of our framework.

Acknowledgement This work is partially supported by the Collaborative Research grant from SenseTime Group (CUHK Agreement No. TS1610626 & No. TS1712093), the Early Career Scheme (ECS) of Hong Kong (No. 24204215), the General Research Fund (GRF) of Hong Kong (No. 14236516, No. 14203518 & No. 14241716), and Singapore MOE AcRF Tier 1 (M4012082.020).

References

[1] Clustering millions of faces by identity. *TPAMI*, 40(2):289–303, 2018. 1, 2, 6

[2] Jingyu Cui, Fang Wen, Rong Xiao, Yuandong Tian, and Xiaoou Tang. Easyalbum: an interactive photo annotation system based on face clustering and re-ranking. In *SIGCHI*. ACM, 2007. 2

[3] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*, 2018. 1, 5

[4] 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. 6

[5] Thomas Finley and Thorsten Joachims. Supervised clustering with support vector machines. In *ICML*. ACM, 2005. 2

[6] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 3

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 3

[8] 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. 5

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017. 2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*. IEEE, 2017. 2

[11] Jeffrey Ho, Ming-Hsuan Yang, Jongwoo Lim, Kuang-Chih Lee, and David Kriegman. Clustering appearances of objects under varying illumination conditions. In *CVPR*, 2003. 1, 2

[12] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010. 2

[13] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009. 2

[14] Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *CVPR*, 2016. 5

[15] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 2

[16] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, and Anil K Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In *CVPR*, 2015. 5

[17] Wei-An Lin, Jun-Cheng Chen, Carlos D Castillo, and Rama Chellappa. Deep density clustering of unconstrained faces. In *CVPR*, 2018. 2

[18] Wei-An Lin, Jun-Cheng Chen, and Rama Chellappa. A proximity-aware hierarchical clustering of faces. In *FG*. IEEE, 2017. 2

[19] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 1, 6

[20] Daniel Müllner et al. fastcluster: Fast hierarchical, agglomerative clustering routines for r and python. *Journal of Statistical Software*, 53(9):1–18, 2013. 6

[21] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*. IEEE, 2014. 5

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 1

[23] 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. 2

[24] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973. 6

[25] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *NeurIPS*, 2014. 1

[26] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *stat*, 1050:7, 2017. 2

[27] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Zhifeng Li, Dihong Gong, Jingchao Zhou, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018. 1

[28] Lior Wolf, Tal Hassner, and Itay Maoz. *Face recognition in unconstrained videos with matched background similarity*. IEEE, 2011. 5

[29] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 2

[30] 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. 1, 2, 6

[31] 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. 1

[32] Ming Zhao, Yong Wei Teo, Siliang Liu, Tat-Seng Chua, and Ramesh Jain. Automatic person annotation of family photo album. In *ICIVR*. Springer, 2006. 1