# Learning to Cluster Faces via Confidence and Connectivity Estimation

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

## Abstract

*Face clustering is an essential tool for exploiting the unlabeled face data, and has a wide range of applications including face annotation and retrieval. Recent works show that supervised clustering can result in noticeable performance gain. However, they usually involve heuristic steps and require numerous overlapped subgraphs, severely restricting their accuracy and efficiency. In this paper, we propose a fully learnable clustering framework without requiring a large number of overlapped subgraphs. Instead, we transform the clustering problem into two sub-problems. Specifically, two graph convolutional networks, named GCN-V and GCN-E, are designed to estimate the confidence of vertices and the connectivity of edges, respectively. With the vertex confidence and edge connectivity, we can naturally organize more relevant vertices on the affinity graph and group them into clusters. Experiments on two large-scale benchmarks show that our method significantly improves clustering accuracy and thus performance of the recognition models trained on top, yet it is an order of magnitude more efficient than existing supervised methods.*

## 1. Introduction

Thanks to the explosive growth of annotated face datasets [19, 11, 17], face recognition has witnessed great progress in recent years [31, 27, 33, 7, 40]. Along with this trend, the ever-increasing demand for annotated data has resulted in prohibitive annotation costs. To exploit massive unlabeled face images, recent studies [14, 39, 35, 38] provide a promising clustering-based pipeline and demonstrate its effectiveness in improving the face recognition model. They first perform clustering to generate "pseudo labels" for unlabeled images and then leverage them to train the model in a supervised way. The key to the success of these approaches lies in an effective face clustering algorithm.

Existing face clustering methods roughly fall into two categories, namely, unsupervised methods and supervised methods. Unsupervised approaches, such as K-means [22] and DBSCAN [9], rely on specific assumptions and lack the capability of coping with the complex cluster structures in real-world datasets. To improve the adaptivity to different data, supervised clustering methods have been proposed [35, 38] to learn the cluster patterns. Yet, both accuracy and efficiency are far from satisfactory. In particular, to cluster with the large-scale face data, existing supervised approaches organize the data with numerous small subgraphs, leading to two main issues. First, processing subgraphs involves heuristic steps based on simple assumptions. Both subgraph generation [38] and prediction aggregation [35] depend on heuristic procedures, thus limiting their performance upper bound. Furthermore, the subgraphs required by these approaches are usually highly overlapped,

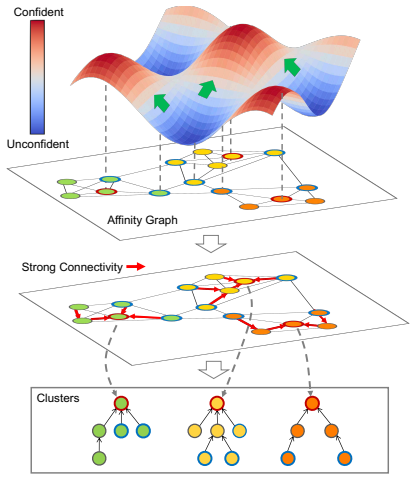


Figure 1: The core idea of our approach. Vertices with different colors represent different classes. Previous methods group all vertices in the box into a cluster as they are densely connected, while our approach, learning to estimate the confidence of belonging to a specific class, is able to detect unconfident vertices that lie among multiple classes. With the estimated vertex confidence, we further learn to predict the edge connectivity. By connecting each vertex to a neighbor with higher confidence and strongest connection, we partition the affinity graph into trees, each of which naturally represents a cluster.

incurring excessive redundant computational costs.

We therefore seek an algorithm that learns to cluster more *accurately* and *efficiently*. For higher accuracy, we desire to make all components of the framework learnable, moving beyond the limitations of heuristic procedures. On the other hand, to reduce the redundant computations, we intend to reduce the number of required subgraphs. Previous works [39, 35, 38] have shown that clusters on an affinity graph usually have some structural patterns. We observe that such structural patterns are mainly originated from two sources, namely *vertices* and *edges*. Intuitively, connecting each vertex to a neighbor, which has higher confidence of belonging to a specific class, can deduce a number of trees from the affinity graph. The obtained trees naturally form connected components as clusters. Based on this motivation, we design a fully learnable clustering approach, without requiring numerous subgraphs, thus leading to high accuracy and efficiency.

Particularly, we transform the clustering problem into two sub-problems. One is to estimate the confidence of a vertex, which measures the probability of a vertex belonging to a specific class. The other is to estimate the edge connectivity, which indicates the probability of two vertices belonging to the same class. With the vertex confidence and edge connectivity, we perform clustering in a natural way, *i.e*., each vertex is connected to a vertex with higher confidence and strongest connectivity. As Figure 1 illustrates, each vertex finds an edge connected to a vertex with higher confidence, and the vertices that finally connected to the same vertex belong to the same cluster.

Two learnable components, namely, a *confidence estimator* and a *connectivity estimator* are proposed to estimate the vertex confidence and edge connectivity, respectively. Both components are based on a GCN to learn from the data, denoted by GCN-V (for vertex confidence) and GCN-E (for edge connectivity). Specifically, GCN-V takes the entire graph as input and simultaneously estimates confidence for all vertices. GCN-E takes the graph constructed from a local candidate set as input and evaluates the possibility of two vertices belonging to the same class.

The experiments demonstrate that our approach not only significantly accelerates the existing supervised methods by an order of magnitude, but also outperforms the recent state of the art [38] under two F-score metrics on $5M$ unlabeled data. The main contributions lie in three aspects: (1) We propose a novel framework that formulates clustering as an estimation of confidence and connectivity, both based on learnable components. (2) Our approach is an order of magnitude faster than existing learning-based methods. (3) The proposed method achieves state-of-the-art performance on both large-scale face clustering and fashion clustering. The discovered clusters boost the face recognition model to a level that is comparable to its supervised counterparts.

## 2. Related Work

**Unsupervised Face Clustering.** With the emergence of deep learning, recent works primarily adopt deep features from a CNN-based model and focus on the design of similarity metrics. Otto *et al*. [1] proposed an approximate rank-order metric. Lin *et al*. [20] introduced minimal covering spheres of neighborhoods as the similarity metric. Besides methods designed specifically for face clustering, classical clustering algorithms can also be applied to face clustering. Density-based clustering is the most related approach. DBSCAN [9] computed empirical density and designated clusters as dense regions in the data space. OPTICS [3] adopted similar concepts and addresses the ordering of data points.

The proposed method shares common intuitions with the density-based clustering, *i.e*., computing the "density" for each sample [9] and focusing on the relative order between samples [3]. Yet, our method differs substantially with all the unsupervised methods above: all components in our framework are *learnable*. This allows us to learn to capture the intrinsic structures in face clusters.

**Supervised Face Clustering.** Recent works have shown that the introduced supervised information in face clustering leads to considerable performance gains. Zhan *et al*. [39] trained a MLP classifier to aggregate information and thus discover more robust linkages. Wang *et al*. [35] further improved the linkage prediction by leveraging GCN to capture graph context. Both methods obtained clusters by finding connected components with dynamic threshold. Yang *et al*. [38] devised a partitioning algorithm to generate multi-scale subgraphs and proposed a two-stage supervised framework to pinpoint desired clusters therefrom.

Whereas the proposed method adopts the idea of supervised clustering, it differs from two key aspects: (1) Unlike previous supervised methods [39, 35, 38], it does not rely on heuristic algorithms for pre-processing or post-processing. Instead, all components of the proposed framework are learnable and can potentially achieve higher accuracy. (2) It is more efficient in design. Existing methods rely on a large number of subgraphs for pinpointing clusters. [35] predicted all connections around each vertex, where two nearby vertices are likely to have highly overlapped neighborhoods, and thus there are redundant computational costs. [38] produced multi-scale subgraphs for detection and segmentation, the number of which is usually several times larger than the number of clusters. In contrast, the proposed method adopts an efficient subgraph-free strategy to estimate the vertex confidence and concentrates on a small portion of neighborhoods for connectivity prediction.

**Graph Convolutional Networks.** Graph Convolutional Networks (GCNs) [18] have been successfully applied to various tasks [18, 12, 32, 37, 36]. Some recent efforts extend GCN to handle large-scale graphs. GraphSAGE [12]

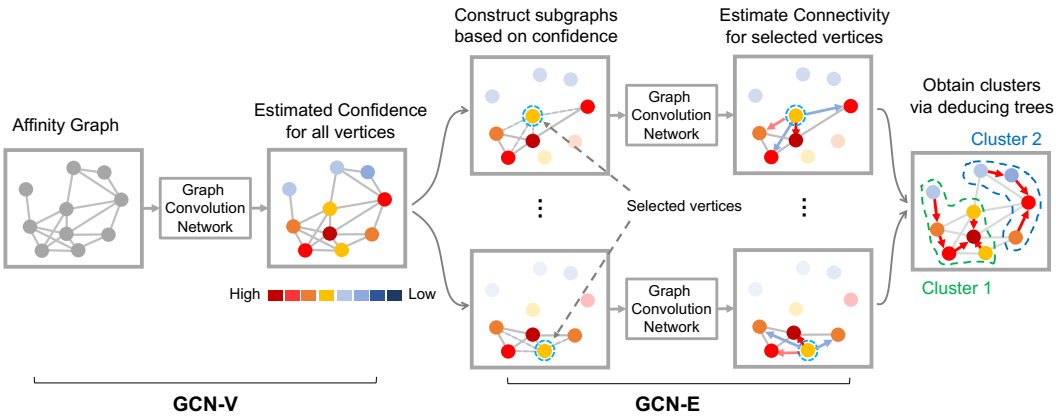


Figure 2: Overview of the proposed clustering framework.

sampled a fixed number of neighbors in each layer for aggregation. FastGCN [4] further reduced computational cost by sampling vertices rather than neighbors. In this paper, we draw on the strong expressive power of graph convolutional networks, to learn vertex confidence on the massive affinity graph and edge connectivity on the local subgraphs.

## 3. Methodology

In large-scale face clustering, supervised approaches demonstrate their effectiveness in handling complex cluster patterns, but their accuracy is limited by some hand-crafted components and their efficiency suffers from the requirement of numerous highly overlapped subgraphs. Therefore, how to cluster accurately and efficiently remains a problem. To address the challenge, we propose an efficient alternative in which all components are learnable. Specifically, we formulate clustering as a procedure of estimating vertex confidence and edge connectivity on an affinity graph, and then partition the graph into clusters by connecting each vertex to neighbors with higher confidence and connectivity.

### 3.1. Framework Overview

Given a dataset, we extract the feature for each image from a trained CNN, forming a feature set $\mathcal{F} = \{\mathbf{f}_i\}_{i=1}^{N}$, where $\mathbf{f}_i \in \mathbb{R}^D$. $N$ is the number of images and $D$ denotes the feature dimension. The affinity between sample $i$ and sample $j$ is denoted as $a_{i,j}$, which is the cosine similarity between $\mathbf{f}_i$ and $\mathbf{f}_j$. According to the affinities, we represent the dataset with a $K$NN affinity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each image is a vertex belonging to $\mathcal{V}$ and is connected to its $K$ nearest neighbors, forming $K$ edges belonging to $\mathcal{E}$. The constructed graph can be expressed as a vertex feature matrix $\mathbf{F} \in \mathbb{R}^{N \times D}$ and a symmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where $a_{i,j} = 0$ if $v_i$ and $v_j$ are not connected.

To perform clustering by learning the structural patterns from vertices and edges, we decompose the clustering into two sub-problems. One is to predict the *confidence* of the vertices. The confidence is to determine whether a vertex belongs to a specific class. Intuitively, a vertex with high confidence usually lies in the place where the vertices are densely distributed and belong to the same class, while the vertices with low confidence are likely to be on the boundary among several clusters. The other is sub-problem to predict the *connectivity* of the edges. The edge with high connectivity indicates the two connected samples tend to belong to the same class. With the vertex confidence and the edge connectivity in the affinity graph, clustering can be performed in a simple way by finding a directed path from vertices with lower confidence to those with higher confidence. This process naturally forms a number of trees isolated from each other, thus readily partitioning the graph into clusters. We refer to this process as *tree-based partition*.

The key challenge for the proposed method remains in how to estimate vertex confidence and edge connectivity. As shown in Figure 2, our framework consists of two learnable modules, namely *Confidence Estimator* and *Connectivity Estimator*. The former estimates the vertex confidence based on GCN-V, while the latter predicts the edge connectivity based on GCN-E. Specifically, GCN-V takes the entire affinity graph as input and simultaneously estimates confidence for all vertices. GCN-E takes the graph constructed from a candidate set as input and evaluates the confidence of two vertices belonging to the same class. According to the output of these two modules, we perform our tree-based partition to obtain clusters.

### 3.2. Confidence Estimator

Similar to anchor-free methods in object detection [41, 8], where they use heatmap to indicate the possibility that an object appears in the corresponding area of an image, the confidence estimator aims to estimate a value for each vertex, thereby indicating whether there is a specific class

on the corresponding area of an affinity graph.

As real-world datasets usually have large intra-class variations, each image may have different confidence values even when they belong to the same class. For an image with high confidence, its neighboring images tend to belong to the same class while an image with low confidence is usually adjacent to the images from the other class. Based on this observation, we can define the confidence $c_i$ for each vertex based on the labeled images in the neighborhood:

$$c_i = \frac{1}{|\mathcal{N}_i|} \sum_{v_j \in \mathcal{N}_i} (\mathbb{1}_{y_j = y_i} - \mathbb{1}_{y_j \neq y_i}) \cdot a_{i,j}, \tag{1}$$

where $\mathcal{N}_i$ is the neighborhood of $v_i$, $y_i$ is the ground-truth label of $v_i$, and $a_{i,j}$ is the affinity between $v_i$ and $v_j$. The confidence measures whether the neighbors are close and from the same class. Intuitively, vertices with dense and pure connections have high confidence, while vertices with sparse connections or residing in the boundary among several clusters have low confidence. We investigate some different designs of confidence in Sec. 4.3.1.

**Design of Confidence Estimator.** We assume that vertices with similar confidence have similar structural patterns. To capture such patterns, we learn a graph convolutional network [18], named GCN-V, to estimate confidence of vertices. Specifically, given the adjacency matrix $\mathbf{A}$ and the vertex feature matrix $\mathbf{F}$ as input, the GCN predicts confidence for each vertex. The GCN consists of $L$ layers and the computation of each layer can be formulated as:

$$\mathbf{F}_{l+1} = \sigma\left(g(\tilde{\mathbf{A}}, \mathbf{F}_l)\mathbf{W}_l\right), \tag{2}$$

where $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-1}(\mathbf{A} + \mathbf{I})$ and $\tilde{\mathbf{D}}_{ii} = \sum_j (\mathbf{A} + \mathbf{I})_j$ is a diagonal degree matrix. The feature embedding of the input layer $\mathbf{F}_0$ is set with the feature matrix $\mathbf{F}$, and $\mathbf{F}_l$ contains the embeddings at $l$-th layer. $\mathbf{W}_l$ is a trainable matrix to transform the embeddings into a new space. $\sigma$ is a nonlinear activations (*ReLU* in this work). To leverage both input embeddings and embeddings after neighborhood aggregation to learn the transformation matrix, we define $g(\cdot, \cdot)$ as the concatenation of them:

$$g(\tilde{\mathbf{A}}, \mathbf{F}_l) = [(\mathbf{F}_l)^\top, (\tilde{\mathbf{A}}\mathbf{F}_l)^\top]^\top. \tag{3}$$

Such definition has been proven to be more effective than simply taking weighted average of the embedded feature of neighbors around each vertex [35]. Based on the output embedding of the $L$-th layer, *i.e.*, $\mathbf{F}_L$, we employ a fully-connected layer to predict the confidence of the vertices.

$$\mathbf{c}' = \mathbf{F}_L\mathbf{W} + \mathbf{b}, \tag{4}$$

where $\mathbf{W}$ is trainable regressor and $\mathbf{b}$ is trainable bias. The predicted confidence of $v_i$ can be taken from the corresponding element in $\mathbf{c}'$, denoted by $c'_i$.

**Training and Inference.** Given a training set with class labels, we can obtain the ground-truth confidence following Eq. 1 for each vertex. Then we train GCN-V, with the objective to minimize the *mean square error(MSE)* between ground truth and predicted scores, which is defined by:

$$\mathcal{L}_V = \frac{1}{N} \sum_{i=1}^{N} |c_i - c'_i|^2 \tag{5}$$

During inference, we use the trained GCN-V to predict the confidence of each vertex. The obtained confidence is used in two ways. First, they are used in the next module to determine whether the connectivity of an edge needs to be predicted, thus significantly reduces the computational cost. Furthermore, they are used in the final clustering to provide partial orders between vertices.

**Complexity Analysis.** The main computational cost lies in the graph convolution (Eq. 2). Since the built graph is a $K$NN graph with $K \ll N$, the affinity matrix $\mathbf{A}$ is a highly sparse matrix. Therefore, the graph convolution can be efficiently implemented as the sparse-dense matrix multiplication, yielding a complexity $\mathcal{O}(|\mathcal{E}|)$ [18]. As the number of edges $|\mathcal{E}|$ of the sparse matrix is bounded by $NK$, the inference complexity is linear in the number of vertices as $K \ll N$. This operation can be scaled to a very large setting by sampling neighbors or sampling vertices [12, 4]. Empirically, a 1-layer GCN takes $37G$ CPU Ram and $92s$ with 16 CPU on a graph with $5.2M$ vertices for inference.

### 3.3. Connectivity Estimator

For a vertex $v_i$, neighbors with confidence larger than $c_i$ indicate they are more confident to belong to a specific class. To assign $v_i$ to a specific class, an intuitive idea is to connect $v_i$ to neighbors from the same class with larger confidence. However, neighbors with larger confidence do not necessarily belong to the same class. We therefore introduce the connectivity estimator, named GCN-E, to measure the pairwise relationship based on the local graph structures.

**Candidate set.** Given the predicted vertex confidence, we first construct a candidate set $\mathcal{S}$ for each vertex.

$$\mathcal{S}_i = \{v_j | c'_j > c'_i, v_j \in \mathcal{N}_i\}. \tag{6}$$

The idea of candidate set is to select edges connected to neighbors more confident to belong to a cluster, and $\mathcal{S}_i$ only contains the vertices with higher confidence than the confidence of $v_i$.

**Design of Connectivity Estimator.** GCN-E shares similar GCN structures with GCN-V. The main difference lies in: (1) Instead of operating on the entire graph $\mathcal{G}$, the input of GCN-E is a subgraph $\mathcal{G}(\mathcal{S}_i)$ containing all vertices in

$\mathcal{S}_i$; (2) GCN-E outputs a value for each vertex on $\mathcal{G}(\mathcal{S}_i)$ to indicate how likely it shares the same class with $v_i$.

More specifically, the subgraph $\mathcal{G}(\mathcal{C}_i)$ can be represented by the affinity matrix $\mathbf{A}(\mathcal{S}_i)$ and the vertex feature matrix $\mathbf{F}(\mathcal{S}_i)$. We subtract $\mathbf{f}_i$ from each row of the feature matrix $\mathbf{F}(\mathcal{S}_i)$ to encode the relationship between $\mathcal{S}_i$ and $v_i$, and the obtained feature matrix is denoted by $\bar{\mathbf{F}}(\mathcal{S}_i)$. The transformation in GCN-E can be therefore represented by:

$$\bar{\mathbf{F}}_{l+1} = \sigma\left(g(\tilde{\mathbf{A}}(\mathcal{C}_i), \bar{\mathbf{F}}_l(\mathcal{C}_i))\mathbf{W}'_l\right), \tag{7}$$

where $\sigma$, $g(\cdot)$ and $\tilde{\mathbf{A}}(\mathcal{S}_i)$ are defined similar to those in Eq. 2. $\mathbf{W}'_l$ is the parameter of GCN-E in the $l$-th layer. Based on the output embedding of the $L$-th layer, we obtain the connectivity for each vertex in $\mathcal{S}_i$ by a fully-connected layer. As the connectivity reflects the relationship between two vertices, we use $r'_{i,j}$ to indicate the predicted connectivity between $v_i$ and $v_j$.

**Training and Inference.** Given a training set with class labels, for a vertex $v_i$, if a neighbor $v_j$ shares the same label with the $v_i$, the connectivity is set to 1, otherwise it is 0.

$$r_{i,j} = \begin{cases} 1, & y_i = y_j \\ 0, & y_i \neq y_j \end{cases}, v_j \in \mathcal{C}_i, \tag{8}$$

We aim to predict the connectivity that reflects whether two vertices belong to the same class. Similar to Eq. 5 in GCN-V, we also use vertex-wise MSE loss to train GCN-E.

$$\mathcal{L}_E(\mathcal{C}_i) = \sum_{v_j \in \mathcal{C}_i} |r_{i,j} - r'_{i,j}|^2 \tag{9}$$

To accelerate the training and inference procedures, we only apply GCN-E to a small portion of vertices with large estimated confidence, as they potentially influence more successors than vertices with small confidence do. We denote the portion of vertices using GCN-E as $\rho$. For other vertices, they simply connect to their $M$ nearest neighbors in the candidate set, indicating they connect to neighbors with top-$M$ largest similarities and higher confidence. $M = 1$ leads to the tree-based partition strategy, while $M > 1$ produces directed acyclic graphs as clusters. Empirical results indicate that $M = 1, \rho = 10\%$ can already bring considerable performance gain (see Sec. 4.3.2).

**Complexity Analysis.** The idea of connectivity estimator shares similar spirits to [35], where they evaluated how likely each vertex on a subgraph connects to the center vertex. Although the complexity of [35] is linear with $N$, applying a GCN on the neighborhood of each vertex incurs excessive computational demands. The proposed GCN-E has two key designs to be much more efficient: (1) We only predict linkages in the candidate set, an effort that potentially involves fewer neighbors for each vertex and does not need to manually select the number of hops and the number of neighbors for each hop. (2) With the estimated vertex confidence, we are able to focus on a small portion of vertices with high confidence. With these two important designs, we achieve a speedup over [35] by an order of magnitude.

# 4. Experiments

## 4.1. Experimental Settings

**Face clustering.** MS-Celeb-1M [11] is a large-scale face recognition dataset consisting of $100K$ identities, and each identity has about 100 facial images. We adopt the widely used annotations from ArcFace [7], yielding a reliable subset that contains $5.8M$ images from $86K$ classes. We randomly split the cleaned dataset 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.

**Fashion clustering.** We also evaluate the effectiveness of our approach for datasets beyond the face images. We test on a large subset of DeepFashion [21], namely In-shop Clothes Retrieval, which is very *long-tail*. Particularly, we mix the training features and testing features in the original split, and randomly sample $25,752$ images from $3,997$ categories for training and the other $26,960$ images with $3,984$ categories for testing. Note that fashion clustering is also regarded as an open set problem and there is no overlap between training categories and testing categories.

**Face recognition.** We evaluate face recognition model on MegaFace [17], which is the largest benchmark for face recognition. It includes a probe set from FaceScrub [25] with $3,530$ images and a gallery set containing $1M$ images.

**Metrics.** We assess the performance on both *clustering* and *face recognition*. Face clustering is commonly evaluated by two metrics [29, 35, 38], namely *Pairwise F-score* and *BCubed F-score* [2]. The former emphasizes on large clusters as the number of pairs grows quadratically with cluster size, while the latter weights clusters according to their cluster size. Both metrics are the harmonic mean of precision and recall, referred to as $F_P$ and $F_B$, respectively. Face recognition is evaluated with *face identification* benchmark in MegaFace. 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.

**Implementation Details.** To construct the $K$NN affinity graph, we set $K = 80$ for MS1M and $K = 5$ for DeepFashion. Since GCN-V operates on a graph with millions of vertices, we only use 1-layer GCN to reduce the computational cost. For GCN-E, it operates on a neighborhood with no more than $K$ vertices, and thus we use 4-layer GCN to increase its expressive power. For both datasets, momentum SGD is used with a start learning rate 0.1 and weight decay $1e^{-5}$. To avoid the situation where there is no correct

Table 1: Comparison on face clustering with different numbers of unlabeled images. (MS-Celeb-1M)

| #unlabeled | 584K | | 1.74M | | 2.89M | | 4.05M | | 5.21M | | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method / Metrics | $F_P$ | $F_B$ | $F_P$ | $F_B$ | $F_P$ | $F_B$ | $F_P$ | $F_B$ | $F_P$ | $F_B$ | |
| K-means [22, 28] | 79.21 | 81.23 | 73.04 | 75.2 | 69.83 | 72.34 | 67.9 | 70.57 | 66.47 | 69.42 | 11.5h |
| HAC [30] | 70.63 | 70.46 | 54.4 | 69.53 | 11.08 | 68.62 | 1.4 | 67.69 | 0.37 | 66.96 | 12.7h |
| DBSCAN [9] | 67.93 | 67.17 | 63.41 | 66.53 | 52.5 | 66.26 | 45.24 | 44.87 | 44.94 | 44.74 | **1.9m** |
| ARO [1] | 13.6 | 17 | 8.78 | 12.42 | 7.3 | 10.96 | 6.86 | 10.5 | 6.35 | 10.01 | 27.5m |
| CDP [39] | 75.02 | 78.7 | 70.75 | 75.82 | 69.51 | 74.58 | 68.62 | 73.62 | 68.06 | 72.92 | 2.3m |
| L-GCN [35] | 78.68 | 84.37 | 75.83 | 81.61 | 74.29 | 80.11 | 73.7 | 79.33 | 72.99 | 78.6 | 86.8m |
| LTC [38] | 85.66 | 85.52 | 82.41 | **83.01** | 80.32 | **81.1** | 78.98 | 79.84 | 77.87 | 78.86 | 62.2m |
| **Ours (V)** | 87.14 | 85.82 | 83.49 | 82.63 | 81.51 | 81.05 | 79.97 | 79.92 | 78.77 | **79.09** | 4.5m |
| **Ours (V + E)** | **87.55** | **85.94** | **83.73** | 82.7 | **81.83** | **81.1** | **80.22** | **79.93** | **79.04** | 79.08 | 11.5m |

Table 2: Performance on DeepFashion clustering.

| Methods | #clusters | $F_P$ | $F_B$ | Time |
|---|---|---|---|---|
| K-means [22] | 3991 | 32.86 | 53.77 | 573s |
| HAC [30] | 17410 | 22.54 | 48.77 | 112s |
| DBSCAN [9] | 14350 | 25.07 | 53.23 | 2.2s |
| MeanShift [5] | 8435 | 31.61 | 56.73 | 2.2h |
| Spectral [15] | 2504 | 29.02 | 46.4 | 2.1h |
| ARO [1] | 10504 | 26.03 | 53.01 | 6.7s |
| CDP [39] | 6622 | 28.28 | 57.83 | **1.3s** |
| L-GCN [35] | 10137 | 28.85 | 58.91 | 23.3s |
| LTC [38] | 9246 | 29.14 | 59.11 | 13.1s |
| **Ours (V)** | 4998 | 33.07 | 57.26 | 2.5s |
| **Ours (V + E)** | 6079 | **38.47** | **60.06** | 18.5s |

neighbor for connection, we set a threshold $\tau$ to cut off the edges with small similarities. $\tau$ is set to 0.8 for all settings.

## 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 [22],** the commonly used clustering algorithm. For $N \geq 1.74M$, we use mini-batch K-means, yielding a comparable result but significantly shorted running time.
**(2) HAC [30],** the method hierarchically merges close clusters based on some criteria in a bottom-up manner.
**(3) DBSCAN [9]** extracts clusters based on a designed density criterion and leaves the sparse background as noises.
**(4) MeanShift [6]** pinpoints clusters which contain a set of points converging to the same local optimal.
**(5) Spectral [24]** partitions the data into connected components based on spectrum of the similarity matrix.
**(6) ARO [1]** performs clustering with an approximate nearest neighbor search and a modified distance measure.
**(7) CDP [39],** a graph-based clustering algorithm, which exploits more robust pairwise relationship.
**(8) L-GCN [35],** a recent supervised method that adopts GCNs to exploit graph context for pairwise prediction.
**(9) LTC [38],** another recent supervised method that formulates clustering as a detection and segmentation pipeline.
**(10) Ours (V),** the proposed method that applies GCN-V on the entire graph and obtains clusters through connecting each vertex to its nearest neighbor in the candidate set.
**(11) Ours (V + E),** the proposed method that employs GCN-E on top of GCN-V to estimate the connectivity and obtain clusters by connecting each vertex to the most connective neighbors in the candidate set.

**Results** For all methods, we tune the corresponding hyper-parameters and report the best results. The results in Table 1 and Table 2 show: (1) Given the ground-truth number of clusters, K-means achieves a high F-score. However, the performance is influenced greatly by the number of clusters, making it hard to employ when the number of clusters is unknown. (2) HAC does not require the number of clusters but the iterative merging process involves a large computational budget. Even using a fast implementation [23], it takes nearly 900 hours to yield results when $N$ is $5.21M$. (3) Although DBSCAN is very efficient, it assumes that density among different clusters is similar, which may be the reason for severe performance drop when scaling to large settings. (4) MeanShift yields a good result on fashion clustering but takes a long time to converge. (5) Spectral clustering also performs well but solving eigenvalue decomposition incurs large computation and memory demand, thus limiting its application. (6) The performance of ARO depends on the number of neighbors. With a reasonable time budget, the performance is inferior to other methods in MS1M. (7) CDP is very efficient and achieves a high F-score on different datasets with different scales. For a fair comparison, we compare with the single model version of CDP. (8) L-GCN surpasses CDP consistently but it is an order of magnitude slower than CDP. (9) As a recent approach to cluster face in a supervised manner, LTC shows its advantage in large-scale clustering. However, relying on the itera-

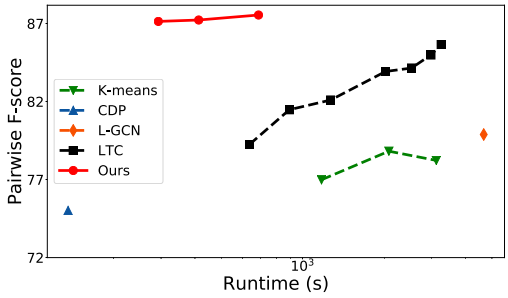


Figure 3: Pairwise F-score *vs*. the runtime of different methods. Note that x-axis is in *log-scale*.

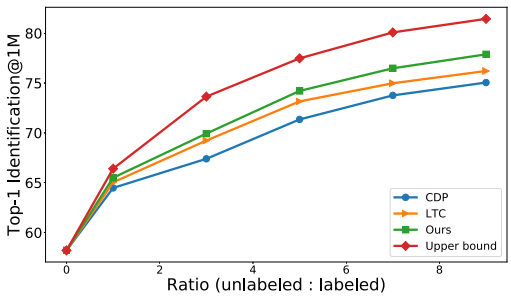


Figure 4: MegaFace top-1 Identification@1M.

tive proposal strategy, the performance gain is accompanied by a large computational cost. (10) The proposed GCN-V outperforms previous methods consistently. Although the training set of GCN-V only contains $580K$ images, it generalizes well to $5.21M$ unlabeled data, demonstrating its effectivenesss in capturing important characteristics of vertices. Besides, as GCN-V simultaneously predicts the confidence for all vertices, it is an order of magnitude faster than previous supervised approaches. (11) We apply GCN-E to $20\%$ vertices with top estimated confidence. It brings further performance gain, especially when applied to DeepFashion. This challenging dataset contains noisy neighborhoods, and thus it is required to select connectivity more carefully.

**Runtime Analysis** We measure the runtime of different methods with ES-2640 v3 CPU and TitanXP. For MS-Celeb-1M, we measure the runtime when $N = 584K$. All the compared approaches, except K-means and HAC, rely on the $K$NN graph. To focus on the runtime of algorithms themselves, we use 1 GPU with 16 CPU to accelerate the search of $K$NN [16], which reduces the time of finding 80 nearest neighbors from 34min to 101s. For all the supervised methods, we analyze their inference time. As shown in Table 1, the proposed GCN-V is faster than L-GCN and LTC by an order of magnitude. GCN-E takes more time to predict the connectivity in the candidate sets, but it is still several times more efficient than L-GCN and LTC. Figure 3 better illustrates the trade-off between accuracy and efficiency. For LTC and Mini-batch K-means, we control the number of proposals and batch size respectively, to yield different runtime and accuracy. In real practice, we can leverage the idea of super vertex in LTC to further accelerate GCN-V, and parallelize GCN-E to estimate connectivity for different vertices simultaneously.

#### 4.2.2 Face Recognition

Following the pipeline of [39, 38], we apply the trained clustering model to assign pseudo labels to unlabeled data, and leverage them to enhance the face recognition model. As Sec. 4.1 introduces, we split the dataset into 10 splits and randomly select 1 split to have the ground-truth labels, denoted as $\mathcal{S}_L$. Particularly, the face recognition experiments involve 4 steps: (1) use $\mathcal{S}_L$ to train a face recognition model $\mathcal{M}_r$; (2) use $\mathcal{M}_r$ to extract face features on $\mathcal{S}_L$ and train the clustering model $\mathcal{M}_c$ with extracted features and corresponding labels in $\mathcal{S}_L$; (3) use $\mathcal{M}_c$ to assign pseudo labels to unlabeled images; (4) use $\mathcal{S}_L$ and unlabeled data with pseudo labels to train the final face recognition model in a multi-task manner. Note that $\mathcal{S}_L$ is used to train both initial face recognition model and face clustering model.

Different from previous work [39, 38], where the unlabeled data are assumed to be obtained sequentially and clustering is performed 9 times on 9 splits separately, we directly perform clustering on $5.21M$ unlabeled data, which is more practical and challenging. The upper bound is trained by assuming all unlabeled data have ground-truth labels. As Figure 4 indicates, all the three methods benefit from an increase of the unlabeled data. Owing to the performance gain in clustering, our approach outperforms previous methods consistently and improves the performance of face recognition model on MegaFace from 58.21 to 77.88.

### 4.3. Ablation Study

To study some important design choices, we select MS-Celeb-1M($584K$) and DeepFashion for ablation study.

#### 4.3.1 Confidence Estimator

**Design of vertex confidence.** We explore different designs of confidence. As the confidence is related to the concept of "density" described in Sec. 2, we first adopt two widely used unsupervised density as the confidence [9, 3, 26]. Given a radius, the first one is defined as the number of vertices and the second one is computed by the sum of edge weights, denoted them as $u^r_{num}$ and $u^r_{weight}$, respectively as shown in Table 3. Note that for these unsupervised definitions, the confidence is directly computed without the learning process. On the other hand, we can define various supervised confidence based on the ground-truth labels. $s_{avg}$ is defined as the average similarity to all vertices with

Table 3: Design choice of vertex confidence. The confidence metrics are defined in Sec. 4.3.1. $\mathbf{F}_L$ denotes the output feature embeddings of $L$-th GCN layer in Sec. 3.2.

| Metric | $\mathbf{F}_L$ | MS1M-584K | | DeepFashion | |
|---|---|---|---|---|---|
| | | $F_P$ | $F_B$ | $F_P$ | $F_B$ |
| $u^r_{num}$ | × | 61.65 | 64.8 | 19.42 | 45.85 |
| $u^r_{weight}$ | × | 81.78 | 80.47 | 29.31 | 52.81 |
| $s_{avg}$ | × | 82.37 | 83.32 | 30.11 | 56.62 |
| $s_{center}$ | × | 82.55 | 83.46 | 31.81 | 56.48 |
| $s_{nbr}$ | × | 82.76 | 83.61 | 32.24 | 57.11 |
| $s^F_{nbr}$ | ✓ | 87.14 | 85.82 | 33.07 | 57.26 |

the same label. $s_{center}$ is defined as the similarity to the center, which is computed as the average feature of all vertices with the same label. $s_{nbr}$ is defined as Eq. 1. $s^F_{nbr}$ indicates using the top embedding $\mathbf{F}_L$ to rebuild the graph. To compare different confidence designs, we adopt the same connectivity estimator by setting $\rho = 0$ and $M = 1$. In this sense, the connectivity estimator directly selects the nearest neighbors in the candidate set without learning.

As shown in Table 3, two unsupervised density definitions achieve relatively lower performance. The assumption that high data density indicates high probability of clusters may not necessarily hold for all the situations. Besides, the performance is sensitive to the selected radius for computing density. Table 3 shows that the supervised confidence outperforms the unsupervised confidence without the need to manually set a radius. Among these three definitions, $s_{nbr}$ achieves better performance than $s_{avg}$ and $s_{center}$. As $s_{nbr}$ is defined on neighborhood, the learning of GCN may be easier compared to $s_{avg}$ and $s_{center}$ which are defined with respect to all samples in the same cluster. In real practice, similar to saliency map fusion in saliency detection [10, 13], we can ensemble the outputs from different confidences to achieve better performance.

**Transformed embeddings.** Comparison between $s_{nbr}$ and $s^F_{nbr}$ indicates that using the transformed features to rebuild the affinity graph leads to performance gain in both datasets. This idea shares common concepts to Dynamic graph [34] where they rebuild the $K$NN graph after each graph convolutional layer. However, on a massive graph with millions of vertices, constructing $K$NN graph per layer will incur prohibitive computational budget. The experiments indicate that only using the top embedding to rebuild the graph can product reasonably well results.

#### 4.3.2 Connectivity Estimator

**The Influence of $\rho$.** We vary $\rho$ from 0 to 1 with a step 0.1. As shown in Figure 5, focusing only on 10% of vertices with high confidence can lead to considerable performance gain while adding very little computational cost. As $\rho$ increases, more vertices benefit from the prediction of GCN-E and thus $F_P$ increases. There is a slight drop when applying GCN-E to all vertices, since connections between unconfident vertices are often very complex, and it may be hard to find common patterns for learning.

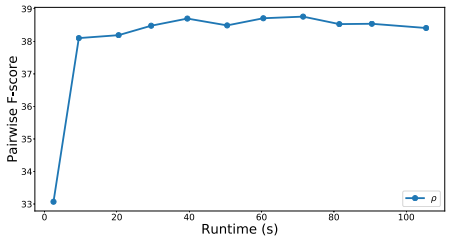


Figure 5: Influence of $\rho$ on DeepFashion. The leftmost point ($\rho = 0$) indicates the result without GCN-E, while the rightmost point ($\rho = 1$) employs GCN-E to all vertices.

**The Influence of $M$.** In Table below, $M = -1$ indicates applying GCN-E without using the candidate set. It includes unconfident neighbors, thus increasing the difficulty of learning and leading to performance drop.

| $M$ | -1 | 1 | 2 | 3 |
|---|---|---|---|---|
| $F_P$ | 29.85 | 38.47 | 1.19 | 0.31 |
| $F_B$ | 56.12 | 60.06 | 56.43 | 52.46 |

When $M = 1$, each vertex connects to its most connective neighbor in the candidate set. When $M > 1$, unconfident vertices will possibly connect to two different clusters. Although it increases the recall of obtained clusters, it may severely impair the precision.

## 5. Conclusion

This paper has proposed a novel supervised face clustering framework, eliminating the requirement of heuristic steps and numerous subgraphs. The proposed method remarkably improves accuracy and efficiency on large-scale face clustering benchmarks. Besides, the experiments indicate the proposed approach generalizes well to a testing set 10 times larger than the training set. Experiments on fashion datasets demonstrate its potential application to datasets beyond human faces. In the future, an end-to-end learnable clustering framework is desired to fully release the power of supervised clustering.

**Acknowledgement** This work is partially supported by the SenseTime Collaborative Grant on Large-scale Multi-modality Analysis (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. 2, 6

[2] 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. 5

[3] 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. 2, 7

[4] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018. 3, 4

[5] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995. 6

[6] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1197–1203. IEEE, 1999. 6

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

[8] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Object detection with keypoint triplets. *arXiv preprint arXiv:1904.08189*, 2019. 3

[9] 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. 1, 2, 6, 7

[10] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. Context-aware saliency detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(10):1915–1926, 2011. 8

[11] 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. 1, 5

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017. 2, 4

[13] Junwei Han, Hao Chen, Nian Liu, Chenggang Yan, and Xuelong Li. Cnns-based rgb-d saliency detection via cross-view transfer and multiview fusion. *IEEE transactions on cybernetics*, 48(11):3171–3183, 2017. 8

[14] 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. 1

[15] Jeffrey Ho, Ming-Hsuan Yang, Jongwoo Lim, Kuang-Chih Lee, and David Kriegman. Clustering appearances of objects under varying illumination conditions. In *CVPR*, 2003. 6

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017. 7

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

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 2, 4

[19] 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. 1

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

[21] 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. 5

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

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

[24] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002. 6

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

[26] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014. 7

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

[28] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010. 6

[29] 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. 5

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

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

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

[33] 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

[34] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019. 8

[35] 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. 1, 2, 4, 5, 6

[36] Sijie Yan, Zhizhong Li, Yuanjun Xiong, Huahan Yan, and Dahua Lin. Convolutional sequence generation for skeleton-based action synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4394–4402, 2019. 2

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

[38] 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. 1, 2, 5, 6, 7

[39] 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, 7

[40] 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

[41] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 3