Density-Aware Feature Embedding for Face Clustering

Senhui Guo[1*] Jing Xu[1*] Dapeng Chen[1 †] Chao Zhang[2] Xiaogang Wang[3] Rui Zhao[1]
[1] SenseTime Group Limited [2] Samsung Research China - Beijing(SRC-B) [3] CUHK

Abstract

Clustering has many applications in research and industry. However, traditional clustering methods, such as K-means, DBSCAN and HAC, impose oversimplifying assumptions and thus are not well-suited to face clustering. To adapt to the distribution of realistic problems, a natural approach is to use Graph Convolutional Networks (GCNs) to enhance features for clustering. However, GCNs can only utilize local information, which ignores the overall characteristics of the clusters. In this paper, we propose a Density-Aware Feature Embedding Network (DA-Net) for the task of face clustering, which utilizes both local and non-local information, to learn a robust feature embedding. Specifically, DA-Net uses GCNs to aggregate features locally, and then incorporates non-local information using a density chain, which is a chain of faces from low density to high density. This density chain exploits the non-uniform distribution of face images in the dataset. Then, an LSTM takes the density chain as input to generate the final feature embedding. Once this embedding is generated, traditional clustering methods, such as density-based clustering, can be used to obtain the final clustering results. Extensive experiments verify the effectiveness of the proposed feature embedding method, which can achieve state-of-the-art performance on public benchmarks.

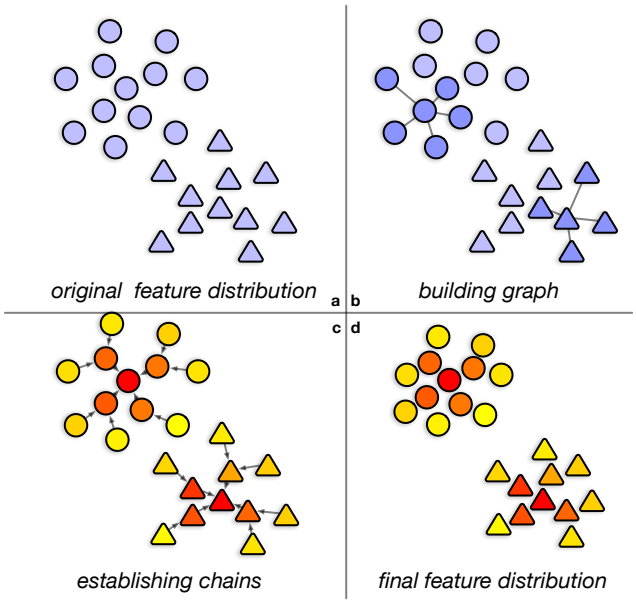


Figure 1. Distribution of face images using Density-Aware Feature Embedding. Circles denote face images of one cluster. Triangles denote face images of another cluster. By establishing density chain from low-density faces to high-density faces, our DA-Net aims to capture relevant information from long-range neighbors for obtaining better feature embedding.

1. Introduction

Thanks to the advances in face detection and recognition, face images can be conveniently collected from the internet or surveillance cameras and further be represented by robust feature vectors. In this situation, there arises a requirement to analyze the face features automatically, and clustering is a practical tool with a wide range of applications. Previously developed non-deep clustering methods use certain global hyper-parameters to determine clustering behavior. For example, DBSCAN relies on a threshold to determine if two nodes should belong to the same cluster, hierarchical clustering methods also employ some criteria to decide when the cluster splitting or merging should be stopped. However, these methods do not perform well on face image datasets with complex distributions, because the variation of the boundary condition of each person's image distribution makes it impossible to cluster face images well enough at the same time. Therefore, a method that can learn to deal with each cluster individually is required. Existing deep learning-based methods for face clustering mainly focus on classifying inter-sample relationships. For example, the approach in [23, 28] learns to recognize the pair-wise relationship of whether they should belong together, and the approach in [27] learns to classify whether several samples belong to the same cluster.

Existing non-deep learning and deep learning approaches use visual features from the feature extraction module directly, which introduces a number of issues. Specifically, as the images of one person can be very differ-

*Co-first authors guosenhui@sensetime.com, xjintw@hotmail.com
†Corresponding authors chendapeng@sensetime.com

ent from the images of another person in terms of variation on pose, illumination, and camera settings, the distributions of intra-class features are usually different for different people, and visual features learned or hand-crafted are still not robust to this variation in distribution, making these clustering approaches easily include outlier samples into clusters. Fig. 1(a) shows an example where the face images of two identities are so close that it is hard to cluster them simply based on the original features. In this paper, we aim to learn context-aware features embedding of an image by using the information from its neighboring images. Better learned features facilitate face clustering by simplifying the inter-class feature distribution.

Inspired by the success of graph convolutional networks (GCNs) over graph-structured data [9, 22, 7, 20], features of an image can be refined using correlated images, by treating each image as a node in the graph. However, standard GCN usually relies on the affinities between the first order neighbors, and the resulting features still lack the information from features at longer distances. To take features at longer distances into consideration, we have to stack GCN layers, whose number of nodes involved, however, would increase geometrically, and this method is found ineffective in previous literature [11]. For example, suppose a sample has 10 neighbors, then 10 steps of message passing along the edges in the graph may correspond to 10^{10} samples/nodes involved in the GCN, which requires too much memory and computation to be practical for training or inference.

Since it is not a desirable option to include the whole graph in the GCN framework, we need to select the nodes with the most important information as the input for our network. The question now is which nodes should be chosen. Due to photo-taking habits and face detection models, faces captured under good conditions, such as from a front-on angle with suitable lighting and a neutral facial expression, are more likely to be detected and registered to the dataset, which we call *model faces*. These faces are supposed to be the most favorable faces we want to pick out and contain the most relevant part of non-local information. As shown in Figure 2, we can see that the yaw angle of most faces lies near 0, specifically, the angles of 93.58% of all faces are within 25 degrees. Thus the problem remains of how to find these features with higher probability density.

Even though it is impractical to estimate the distribution itself, we can still manage to find features with higher densities, and thus we can extract the more important part of the graph to represent the non-local information. Based on the motivations above to locally enhance features and represent non-local information by the model feature, we propose a Density-Aware Feature Embedding Network (DA-Net) to aggregate relevant information from both local and non-local neighbors. To be more specific about the non-local part, we will use the term "long range" to refer to it in the rest of the paper. The DA-Net has two sub-networks. The first sub-network is the local clique network based on GCNs, aiming to learn feature embeddings from the contextual information contained in the local neighborhood. The second is the long-range chain network based on CNNs. It aggregates the knowledge along the chain as a learning path, which starts from the considered sample and gradually moves to the nearby density peak/model feature. Examples of density chains are shown in the Fig. 1(c).

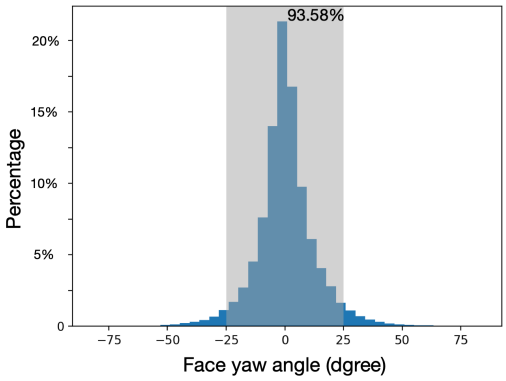


Figure 2. Distribution of face angle in MS-Celeb-1M dataset

As discussed above, features with higher densities are features of faces with better conditions. Thus, a face image and its nearby higher density face image have a high probability of being the same person. Therefore, the chain tends to describe consistent identities, which is the most important part of the graph for each feature on the chain. Generating the feature embedding on the identity-consistent chain is more likely to improve the representation. Such a method is inspired by the assumption in [16], where cluster centers are characterized by a density peak that has a higher density than their neighbors and is distant from other cluster centers. Considering the nearby density peak for a sample can better discover long-range and density-aware information. By simplifying the relationship from the graph to the chain, this designed network only needs to update the feature along the chain, significantly accelerating the training and inference procedure.

In summary, (1) different from conventional face clustering methods that focus on classifying inter-sample relationships, we improve the face clustering problem by learning context-aware feature embeddings, which simplifies the inter-class feature distribution and is scalable on large-scale datasets. (2) The feature embedding is based on a density-guided receptive field. It captures both local neighborhood information and long-range cluster-level information. Superior clustering results are achieved on MS-Celeb-1M [6], YouTube Faces DB [26] and IJB-B [24].

2. Related Work

Face clustering is a challenging task due to its massive data scale, complex intra-class feature distribution, and the ambiguous feature border of different classes. Traditional clustering methods like K-Means[12], spectral clustering[17], and DBSCAN[3] all rely on certain assumptions on data distribution, such as that the clusters are of convex shape, similar size or the same density, and can achieve optimum results when the actual distribution lines up with the expected one. However, these methods ignore the contextual information for clustering, which are not suitable for real-world face clustering.

Context-based Face Clustering. Contextual information helps to cluster faces. Shi *et al.* [18] built a conditional random field in the neighborhood, named Conditional Pairwise Clustering, to facilitate the face clustering. Lin *et al.* [10] learned the minimal covering spheres of neighborhoods and estimated the inter-sample similarities by a new density-based strategy. Zhan *et al.* [28] encoded the pairwise relationships in the neighborhood into a feature vector, then learned to determine whether two nodes belong to the same class. Compared with these methods, GCN is a more efficient tool for graph-structured data and can be applied to face clustering. Wang *et al.* [23] proposed a linkage-based GCN to predict the probability of the connection between a pivot node and its neighbors. Yang *et al.* [27] learned to cluster in a detection-segmentation paradigm, where GCN-Detection selected high-quality clusters from proposals, and GCN-Segmentation identified the outliers. However, these methods use visual features from the feature extraction module directly and only capture short-range local information. Our approach for updating features is complementary to these approaches. As long-range neighbors may still contain consistent semantic meanings to the concerned sample, we propose two sub-networks to aggregate relevant information from both short-range and long-range neighbors.

GNN-based Feature Learning. A Graph Neural Network(GCN) naturally makes use of local graph structure and can learn more discriminative features for the task like classification or linkage prediction. Deep Walk [15] is proposed to generate graph embeddings by combining the SkipGram model [13] with graph random walk. Similar approaches like node2vec and LINE [5] also have achieved excellent performance. Hamilton *et al.* introduced GraphSage [7] to compute node representations in an inductive manner. It sampled a fixed-size neighborhood for each node and then performed simple feature aggregation such as mean pooling, max pooling, and LSTM[8]. In these methods, the nodes involved in the feature learning merely depended on the graph topology, while our method additionally exploits cluster-level feature distribution by defining the density-aware graph, and thus is more effective and efficient for face clustering.

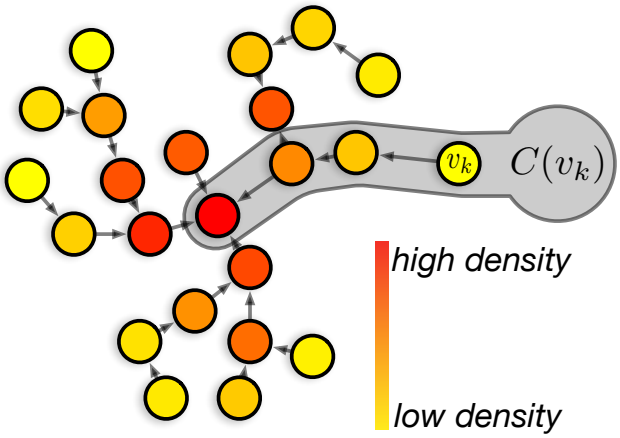


Figure 3. Chain graph. The gray paths, $C(v_k)$ is the chain from node v_k to its nearby density peak. And the chain stops growing once the next nearest higher density neighbor is too far away.

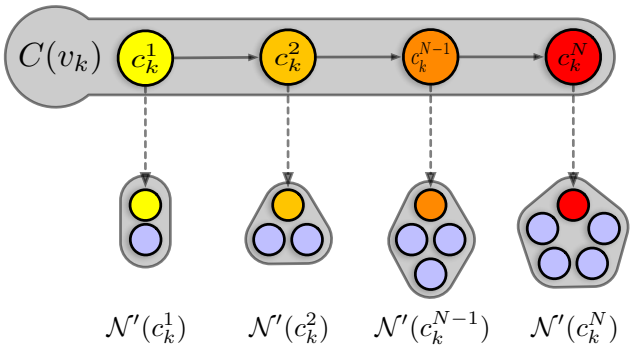


Figure 4. Clique finding. We find a clique for each node in the chain.

Self-supervised learning. Our method makes use of the distribution of clusters and can be regarded as a self-supervision method[1, 2] that employs density information to improve the general representation power.

3. Methodology

In our approach, we first build a density-aware local graph for each sample, then utilize the DA-Net (Density-Aware Feature Embedding Network) to exploit the information within the constructed graph for clustering. The DA-Net consists of a local clique sub-network and a long-range chain sub-network and outputs an enhanced feature for each sample. The obtained feature embeddings of all samples are fed to a density-based clustering strategy to produce the final clusters.

3.1. Data Density

We model the face images in a KNN graph, where each image is represented by a node, and is connected to its K nearest neighbours. For node v_i, its K nearest neighbours are represented by $\mathcal{N}(v_i)$. To utilize non-local information, it is essential to compress the whole graph and extract the most important nodes. Let $f(v_i)$ be the CNN feature of node v_i normalized by its L^2 norm, we believe there will be a distribution for each person in feature space.

Given the probability density function for person l to be $p_l(\cdot)$, the value $p_l(v_i)$ reflects the probability of the person l's picture of node v_i being taken. In practice, it is almost impossible to find out the exact distribution. Based on the observation that the feature distributions of different persons are non-overlap at most of the time, we utilize the data density $\rho(v_i)$, which approximates the probability of the person of interest. With appropriate definition of ρ, we have asymptotic property that

$$\rho(v_i) \to p_l(v_i) \ \text{ for } \ |\mathcal{D}| \to \infty \tag{1}$$

where $\mathcal{D}$ is the data collected for the person. In this way, the feature with high data density tend to have a high probability to be him, thus it is the feature containing the important facial information of that person.

Consider the node v_i, the data density $\rho(v_i)$ is computed based on the neighbours of v_i on the affinity graph. *i.e.*, $\mathcal{N}(v_i)$, which is:

$$\rho(v_i) = \sum_{v_j \in \mathcal{N}(v_i)} \langle f(v_i), f(v_j) \rangle, \tag{2}$$

where their inner-product $\langle f(v_i), f(v_j) \rangle$ measures the similarity between the nodes v_i and v_j.

3.2. Density-aware Graph Construction

The DA-Net utilizes contextual information by constructing a density-aware graph. The graph captures the cluster-level structure, which gradually relates the consider image to its non-local neighbours with higher density. For node v_k, we first generate a chain graph from v_k to the nearby density peak. The chain graph grows in multiple steps, and each step sequentially adds one node to the chain. For notation clarity, we define the chain as $C(v_k) = \{c_k^1, c_k^2, ..., c_k^N\}$, where c_k^i is the node on the chain and c_k^1 is the very node v_k. Suppose the added node at step i is c_k^i, then the node to be added c_k^{i+1} is the nearest neighbor with higher density:

$$c_k^{i+1} = \arg\max_v \{\langle f(c_k^i), f(v) \rangle, \ v \in \{u \mid \rho(u) > \rho(c_k^i)\}. \tag{3}$$

The chain grows until the inner product between the node to be added and the last node is lower than a pre-defined value. As illustrated in Figure 3, the chain gradually moves from the concerned node to the most related density peak. As mentioned in Section 3.1, neighboring samples that are similar to c_k^1 with a higher density are more likely to be a model feature and correspond to the same person. Therefore, these samples with higher data density are more useful in representing the cluster structure and guiding the update of the features for node c_k^1.

Given the chain graph $\{c_k^i\}_{i=1}^N$, we then extend it by attaching a clique to each node in the chain. The nodes in the clique include c_k^i and the nearest neighbors $\mathcal{N}(c_k^i)$ of c_k^i. To alleviate the influence of irrelevant neighbors, we prune the clique by performing a threshold choosing scheme, and the pruned nodes are represented by:

$$\mathcal{N}'(c_k^i) = \{v| \quad \langle f(v), f(c_k^i) \rangle > \tau, v \in \mathcal{N}(c_k^i)\}, \tag{4}$$

where τ is an empirical value based on the original features, and $\tau = 0.6$ in our experiments.

3.3. Local Clique Network

The local clique network is based on GCN, aiming to update the feature of every node in the chain. As shown in Figure 4, for each node, it takes the corresponding clique as input. Given the clique $\mathcal{N}'(c_k^i)$, we define the affinity matrix $A(c_k^i) \in \mathbb{R}^{|\mathcal{N}'(c_k^i)| \times |\mathcal{N}'(c_k^i)|}$, whose elements are computed by the inner product. The initial feature matrix concatenates all the original features in the clique, denoted by $F^0(c_k^i) \in \mathbb{R}^{|\mathcal{N}'(c_k^i)| \times d}$. In each layer of GCN, we update the feature matrix by:

$$\begin{aligned} F^{l+1}(c_k^i) = \sigma(&\alpha \cdot F^l(c_k^i) \\ &+ (1-\alpha) \cdot D^{-1}(c_k^i) A(c_k^i) F^l(c_k^i) W^l), \end{aligned} \tag{5}$$

where $F^l(c_k^i)$ is the updated features of the l-th GCN layer for all nodes belonging to the clique of c_k^i, $D(c_k^i)$ is the diagonal matrix with $D_{i,i}(c_k^i) = \sum_j A_{i,j}(c_k^i)$, σ is the *ReLU* function, and α is a learnable parameter balancing the importance of the updated features and its contexts. Intuitively, this formula expresses a procedure of taking weighted average of the original features in the clique, transforming them with W^l, combining with the previous feature $F^l(c_k^i)$, and then going through a nonlinear activation. This is similar to a CNN module but is operated on a graph with arbitrary topology. Our method significantly accelerates the training and inference by decomposing a large graph into multiple cliques. With L layers of GCN, we obtain the embedded feature of the node c_k^i from the corresponding row in $F^L(c_k^i)$, which is denoted by $\phi(c_k^i)$.

3.4. Long-range Chain Network

The long-range chain network is essentially a CNN-based attention network. For node v_k, the chain features yielded by the local clique network are represented by $\{\phi(c_k^1), \phi(c_k^2), ..., \phi(c_k^N)\}$. Inspired by the "scaled dot-product attention"[21], we use the transformer architecture to further update the node feature as shown in Figure 5.

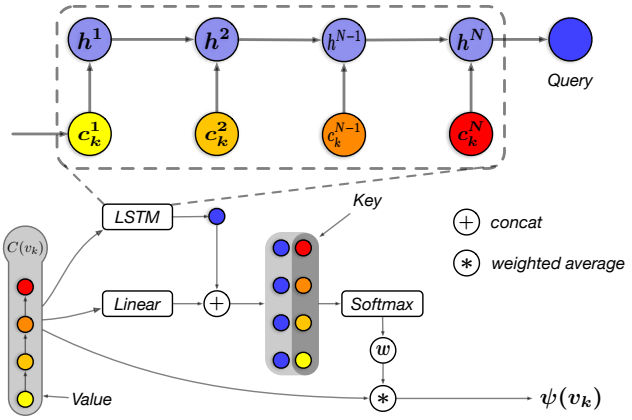


Figure 5. Long-range chain network. The feature chain is processed by an LSTM network to produce a query feature. The inner product between the query and the key of each node produces a weight for that node. The weights on the chain are normalized by a softmax function and are used to summarize the final feature embedding.

The transformer uses triplets in the form of query, key, and value. It firstly estimates a query feature, and the query feature usually indicates what we expect to extract from the given samples. The query is then matched against a list of keys where each key has a value. The final value is then returned as the sum of all the values weighted by the affinities between the keys and the query.

In our method, the query feature should encode the structure of the nearby feature distributions. The constructed chain and the associated features should be robust to the density variations of the neighborhood. We, therefore, apply an LSTM network along the chain from c_k^1 to c_k^N, which consists of the following updating procedure: $\boldsymbol{h}^{i+1} = \text{LSTM}(\phi(c_k^{i+1}), \boldsymbol{h}^i)$, where the LSTM unit takes the features $\phi(c_k^{i+1})$ and $\boldsymbol{h}^i$ as the input and outputs the hidden state $\boldsymbol{h}^{i+1}$ corresponding to the node c_k^{i+1}. The hidden state at the last step $\boldsymbol{h}^N$ is used as the query $\boldsymbol{q}(v_k)$. The keys and values are associated with each node in the chain, which are projected from the features $\phi(c_k^i)$ for $i = 1, \ldots, N$. The key and value corresponding to c_k^i are represented by $\boldsymbol{k}(c_k^i)$ and $v(c_k^i)$ respectively, and the key set and the value set for the chain $C(v_k)$ are respectively denoted by $\mathcal{K}(v_k)$ and $\mathcal{V}(v_k)$.

For feature aggregation, we obtain $\boldsymbol{w}(c_k^i)$ by computing the inner product between the query and the key. The attention weight $\bar{\boldsymbol{w}}(c_k^i)$ is obtained by normalizing $\boldsymbol{w}(c_k^i)$ within the chain using the softmax operation, *i.e.*,

$$\bar{\boldsymbol{w}}(c_k^i) \propto \exp\Big(\frac{\boldsymbol{q}(v_k)^\top \boldsymbol{k}(c_k^i)}{\sqrt{d_k}}\Big), \tag{6}$$

where d_k is the dimension of $\boldsymbol{k}(c_k^i)$. The attention weights

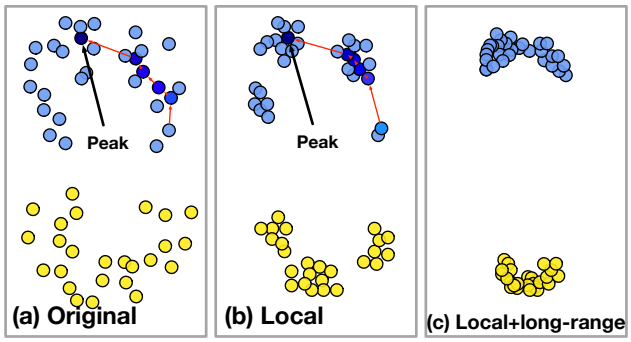


Figure 6. Function of density-aware chain

are then used to compute the final embedding for node v_k:

$$\psi(v_k) = \sum_{i=1}^{N} \bar{\boldsymbol{w}}(c_k^i) \cdot v(c_k^i), \tag{7}$$

which is exactly the proposed density-aware feature embedding for node v_k

Discussion The long-range chain using the density peak essentially connects the samples that are more likely from the same class. In this way, the feature learning over the chain can effectively pull the features of the same class together, which is desired for clustering. Figure 6 illustrates the effects of long-range chain from a boundary sample to the density peak. Figure 6(b) shows that the local clique network can make very similar features closer, forming several clear sub-patterns that might result in wrong clustering results. Our long-range chain associates the samples in different sub-patterns with the local density peak, obtains more compact feature distribution for each class, and facilitates more accurate clustering.

3.5. Connected Graph Generation

We apply simple density-based clustering method to generate clusters. As shown in Algorithm 1, our clustering method consists of two steps. The first step is to generate edges for adjacent nodes, which has a time complexity of $\mathcal{O}(n \log n)$. The second step is to find the clusters by connecting edges. It is dependent on the speed of finding the connected graph, and the time complexity of finding cluster is $\mathcal{O}(\log n)$ while using [4]. Moreover, our feature is also compatible with other clustering methods, *e.g.*, K-Means, DBSCAN.

4. Experiments

Datasets and evaluation metric. Our proposed method is evaluated on three public face clustering benchmarks, MS-Celeb-1M [6], YouTube Faces DB [26] and IJB-B [24]. MS-Celeb-1M consists of 100K identities and about 10M images. We divide it into training and testing sets using the same setting as [27]. YouTube Faces DB contains 3,425

Algorithm 1 Density-based Clustering

Input: feature set Ψ, threshold θ, the number of neighbors k

Output: clusters $\mathbb{C}$

procedure CLUSTERING
  $\mathbb{E}$ = GENERATING EDGES(Ψ, θ, k)
  $\mathbb{C}$ = Use Union-Find Algorithm [4] to find connected graphs on $\mathbb{E}$.
  return $\mathbb{C}$
end procedure

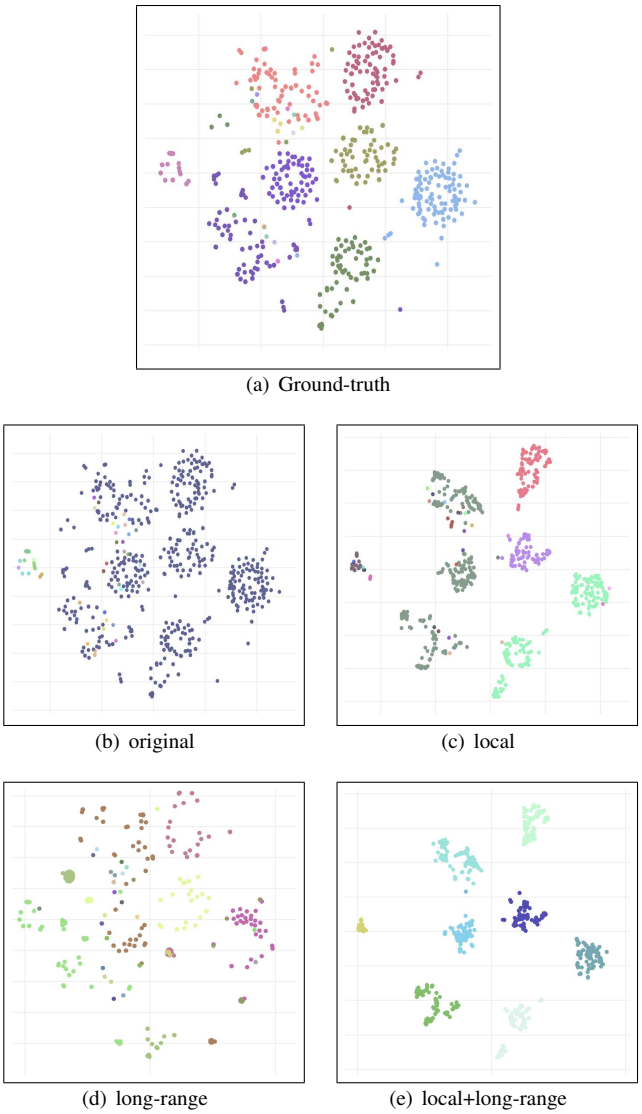
function GENERATING EDGES(Ψ, θ, k)
  $\mathbb{E} = \emptyset$
  for all $\psi(v_i)$ in Ψ **do**
    Find its k nearest neighbors $v_j \in \mathcal{N}(v_j)$
    if $\langle \psi(v_i), \psi(v_j) \rangle > \theta$ **then** $\mathbb{E} = \mathbb{E} \cup \{\{v_i, v_j\}\}$
    end if
  end for
  return $\mathbb{E}$
end function

videos and 1,595 identities, where we use 159 identities for training and the other 1,436 for testing. IJB-B consists of three sets, which include 512, 1,024, 1,845 identities, and 18,171, 36,575, 68,195 samples, respectively, and we use the model trained on MS-Celeb-1M for testing clustering. For a fair comparison, we use the features provided by [27] as the original feature when evaluating on MS-Celeb-1M and YouTube Faces DB, and use IJB-B feature provided by [23]. For MS-Celeb-1M and YouTube Faces DB, we adopted *pairwise* recall, precision and *F-score* to measure the clustering performance, as in [27]. For the IJB-B dataset, we follow its official protocol [24] and adopt the B-Cubed F-score for evaluation.

Implementation Details. In the DA-Net, the local clique network employs a two-layer GCN, and the long-range chain network employs a two-layer LSTM for query feature, two-layer linear transformations for keys and values. We train the DA-Net in an end-to-end fashion. The local clique network takes the density-aware graph as input to update the features on the chain. The long-range chain network summarises the chain to obtain the final density-aware feature embedding. We impose a softmax classification loss over the feature embedding. The network is trained with a learning rate of 0.01 and the SGD optimizer. Most parameters are learnable, except in the long-range chain network, where we use $K = 256$ nearest neighbors to find the local density.

4.1. Ablation Analysis

Our ablation study is conducted on MS-Celeb-1M. To better evaluate our method, we utilize four kinds of features. Specifically, **original** indicates original features; the local clique network generates **local only** features; the long-range chain network outputs **long-range only** feature when taking the original features as input; and the overall framework produces **local+long-range** features.



(a) Ground-truth (b) original (c) local (d) long-range (e) local+long-range

Figure 7. Feature distribution visualization on t-SNE. Several identities are illustrated. Nodes of the same color indicates faces of the same identity in (a) and predicted clusters in (b) (c) (d) and (e). (a) is the ground-truth label for (b). We can observe that most of the samples in (b) are mis-classified into one cluster. After generating local and long-rang network, the features are more compact, thus the predicted result is more accurate.

Local clique network and long-range chain network. We evaluate the main components of our method in two folds: (1) clustering performance; (2) the discriminative power of the feature embedding.

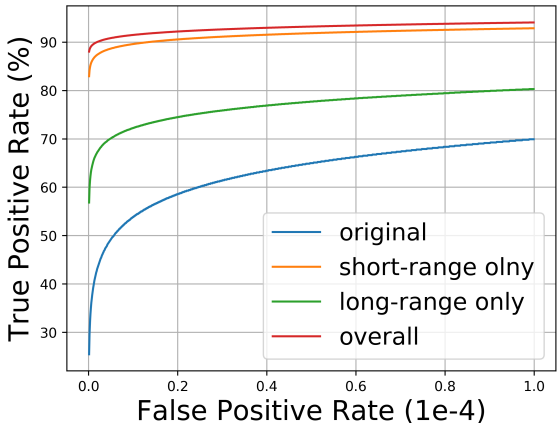


Figure 8. ROC curve of MS-Celeb-1M. Under the same false positive rate, true positive rate of local+long-range feature significantly outperforms the original one.

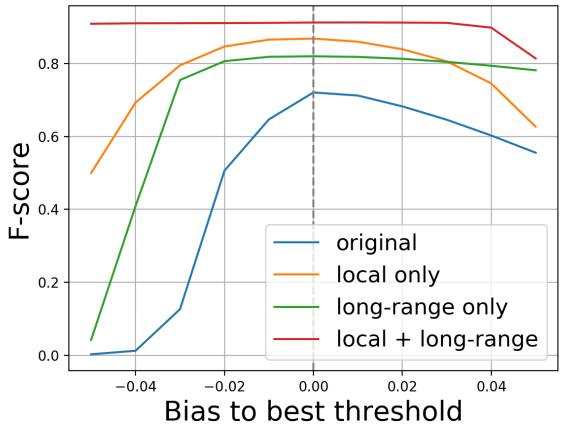


Figure 9. Sensitivity analysis on MS-Celeb-1M. The vertical line in the middle represents the best threshold for clustering.

Clustering performance. First, we evaluate the clustering performance from different settings of features when the connected graph generation module is used for clustering. As shown in Tables 1 and 2, in the MS-Celeb-1M dataset, we can see that both the short-range and the long-range features outperforms original features by 14.47% and 9.68% in F-score, respectively. Furthermore, combining local neighbor structure with long-range chain structure, the short+long-range features achieve 90.60% in F-score.

Feature discriminative power. We utilize the ROC curve to illustrate the discriminative power of our feature embedding. As shown in Figure 8, under the same false-positive rate, the true-positive rate of the short+long feature is much higher than that of other features. On the other hand, we visualize the evolution of the data distribution when applying different components.

Network depth study. As shown in Table 5, the deeper structure the GCN and LSTM they have, the better results their will achieve. We believe a better base network (though not our focus) could further improve the performance.

Feature distribution analysis. After our DA-Net, the features will be more identity-relevant. In other words, the distribution in the embedding space is more compact and discriminative. To see how the distribution is optimized, we first select 10 identities that are close and put their original features, local only features and local+long-range features all together into the t-SNE mapping training to visualize their distribution shown the Figure 7, the local clique network tends to gather local neighbors together. However, it is unable to recognize long-range relationships, thus combines distinct classes into one cluster. On the other hand, the long-range chain network captures distant neighbors but leaves a sparse cluster distribution. After utilizing the strengths from both local and long-range relationships, features are more inner-class compact and inter-class discriminative. Experiments on the other two datasets in Tables 2 and 3 also validate our assumption.

As mentioned, we use a simple labeling scheme that uses a global threshold to decide if two features are from the same class. However, as long as the image distribution remains various, it is impractical to prevent merging different classes and splitting samples of the same class simultaneously. This is reflected by the sensitivity of the clustering performance. As shown in Figure 9, the local+long-range features are more robust to the clustering threshold. This is because of the shrinkage of the intra-class feature distribution, which significantly simplifies the clustering task.

Compatibility. Our method improves face clustering by learning a density-aware feature embedding. This embedding is compatible with other clustering methods. As shown in Table 4, clustering with local+long-range features outperforms using original features in pairwise precision, recall, and F-score.

4.2. Method Comparison

We compare our method with two sets of clustering approaches. One is the conventional approaches, including K-Means [12], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [3], Hierarchical Agglomerative Clustering (HAC) [19], Approximate Rank Order (ARO) [14]. The other is learning based methods, including CDP [28], Learning to Cluster (LearnClust) [27] and linkage-GCN [23].

The experimental results are presented in Tables 1 to 3. These show that our method outperforms other approaches on all datasets. Specifically, in MS-Celeb-1M dataset, our

Methods	Precision	Recall	F-score	Time
K-Means [12]	52.52	70.45	60.18	13h
DBSCAN [3]	72.88	42.46	53.50	100s
HAC [19]	66.84	70.01	6.839	18h
ARO [14]	81.10	7.30	13.34	250s
CDP [28]	80.19	70.47	75.01	350s
LearnClust [27]	98.24	75.93	85.66	2200s
original	85.63	62.17	72.04	30s
local only	93.50	80.50	86.51	178s
long-range only	94.94	71.73	81.72	181s
local+long-range	95.88	85.87	**90.60**	329s

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

Methods	Precision	Recall	F-score
K-Means [12]	60.65	59.60	60.12
DBSCAN [3]	72.88	42.46	53.5
HAC [19]	99.64	87.31	93.07
ARO [14]	81.1	7.3	13.34
CDP [28]	98.32	89.84	93.89
LearnClust [27]	96.75	92.27	94.46
original	98.25	86.57	92.04
local only	99.47	89.86	94.42
long-range only	97.67	88.30	92.75
local+long-range	97.44	92.86	**95.10**

Table 2. Comparison on YouTube Faces DB.

Methods	F_{512}	F_{1024}	F_{1845}
K-Means [12]	0.612	0.603	0.600
DBSCAN [3]	0.753	0.725	0.695
ARO [14]	0.763	0.758	0.755
GCN-A [23]	0.833	0.833	0.814
original	0.785	0.765	0.758
local only	0.827	0.807	0.793
long-range only	0.806	0.797	0.802
local+long-range	**0.834**	**0.833**	**0.828**

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

method outperforms the state-of-the-art method LearnClust [27] by 5%, and is substantially better than conventional methods. In YouTube Faces DB and IJB-B-1845, our method outperforms the state-of-the-art by 0.64% and 1.4% respectively. Our approach generates a context-aware feature, which makes it compatible with other clustering methods, such as K-Means and LearnClust.

Comparison on efficiency and complexity. In our work, the computational complexity arises from three parts: the density-aware graph construction, the network inference, and the cluster generation. The graph construction is

Methods	Precision	Recall	F-score
K-Means+original	52.52	70.45	60.18
K-Means+local +long-range	89.28	89.78	**89.53**
DBSCAN+original	72.88	42.46	53.50
DBSCAN+local +long-range	97.17	79.79	**87.63**

Table 4. Compatibility analysis.

Layer settings	Precision	Recall	F-score
2 layer GCN	93.50	80.50	86.51
2 layer GCN + 2 layer LSTM	95.88	85.87	90.60
4 layer GCN	97.12	78.46	86.80
4 layer GCN + 4 layer LSTM	95.96	86.71	**91.10**

Table 5. Increase parameters on MS-Celeb-1M.

the most time-consuming part with a time complexity of $\mathcal{O}(n^2)$. With approximate nearest neighbor search [25], the searching complexity reduces to $\mathcal{O}(n \log n)$. Then, the graph is only constructed based on the local neighbors, hence the number of graph grows linearly with the number of data, thus the network inference cost is $\mathcal{O}(n)$. Besides, the complexity of cluster generation module is $\mathcal{O}(n)$ as mentioned in Section 3.5. Consequently, the overall complexity is $\mathcal{O}(n \log n)$, which means it is efficient and scalable.

From Table 1, it is clear that our method is much faster compared to other network-based methods and is even comparable with the fastest traditional methods. Specifically, our approach only costs 15% of the time when compared with related GCN-based method LearnClust [27], because the graph in LearnClust constructed is based on the set of nodes generated by cluster proposal, which may involve much more nodes for GCN than the nodes involved using our density-aware graph construction.

5. Conclusion

This paper has presented a density-aware feature embedding framework. It uses GCN to obtain local context and uses a density-aware long-range chain to capture cluster-level information effectively. An LSTM network is applied along to the density-aware chain to aggregate different local information to a unified feature embedding for each node. Efficient clustering algorithm is performed over the improved features. Extensive experimental results on three benchmark datasets and different clustering approaches validate the effectiveness of the proposed DA-Net.

References

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *The European Conference on Computer Vision (ECCV)*, September 2018. 3

[2] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 3

[3] 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*, volume 96, pages 226–231, 1996. 3, 7, 8

[4] Michael Fredman and Michael Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354. ACM, 1989. 5, 6

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. 3

[6] 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*, pages 87–102. Springer, 2016. 2, 5

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017. 2, 3

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 3

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[10] Wei-An Lin, Jun-Cheng Chen, Carlos D Castillo, and Rama Chellappa. Deep density clustering of unconstrained faces. In *CVPR*, pages 8128–8137, 2018. 3

[11] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*, 2018. 2

[12] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 3, 7, 8

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprintarXiv:1301.3781*, 2013. 3

[14] Charles Otto, Dayong Wang, and Anil K Jain. Clustering millions of faces by identity. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):289–303, 2018. 7, 8

[15] Bryan Perozzi, Rami Alrfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 2014. 3

[16] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014. 2

[17] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000. 3

[18] 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. 3

[19] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973. 7, 8

[20] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018. 2

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4

[22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 2

[23] Zhongdao Wang, Liang Zheng, Yali Li, and Shengjin Wang. Linkage Based Face Clustering via Graph Convolution Network. In *CVPR*, 2019. 1, 3, 6, 7, 8

[24] Cameron Whitelam, Emma Taborsky, Austin Blanton, Brianna Maze, Jocelyn Adams, Tim Miller, Nathan Kalka, Anil K Jain, James A Duncan, Kristen Allen, et al. Iarpa janus benchmark-b face dataset. In *CVPR Workshops*, pages 90–98, 2017. 2, 5, 6

[25] P. Wieschollek, O. Wang, A. Sorkine-Hornung, and H. P. A. Lensch. Efficient large-scale approximate nearest neighbor search on the gpu. In *CVPR*, 2016. 8

[26] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR*. IEEE, 2011. 2, 5

[27] Lei Yang, Xiaohang Zhan, Dapeng Chen, Junjie Yan, Chen Change Loy, and Dahua Lin. Learning to Cluster Faces on an Affinity Graph. In *CVPR*, 2019. 1, 3, 5, 6, 7, 8

[28] Xiaohang Zhan, Ziwei Liu, Junjie Yan, Dahua Lin, and Chen Change Loy. Consensus-driven propagation in massive unlabeled data for face recognition. In *ECCV*, pages 568–583, 2018. 1, 3, 7, 8