ISBN: 978-981-18-7950-0
2023 the 13th International Workshop on Computer Science and Engineering (WCSE 2023)
doi: 10.18178/wcse.2023.06.018

Learning to Cluster Faces via Hypergraph Convolution with Transformer on Large Graph

Dengdi Sun [1,4], Zhizhong Huang[2], Bin Luo[2] and Zhuanlian Ding [3+]

[1] School of Artificial Intelligence, Anhui University, Hefei, 230601, China.

[2] School of Computer Science and Technology, Anhui University, Hefei, 230601, China.

[3] School of Internet, Anhui University, Hefei, 230039, China.

[4] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei, 230026, China.

Abstract. Face clustering is a very useful method in image annotation, image retrieval and other real-world applications. The main challenge is that with the increasing data scale, the large graph constructed by KNN is difficult to train due to out of memory. At the same time, the image features of the previous face clustering methods are extracted through CNN, which is not conducive to obtaining the global features of the image. In this paper, we propose a method that uses transformer to extract image features. Then, we use the extracted image features to construct a large graph by KNN, and we randomly partition the large graph into several non-overlapping subgraphs through METIS. In addition, we use hypergraph convolution to learn deeper high-order graph structure data. Experiments on both MS-Celeb-1M and DeepFashion show that our method achieves state-of-the-art performance, eg.90.08% in pair-wise F-score on MS-Celeb-1M.

Keywords: Face clustering, Transformer, Hypergraph convolution network, Large graph.

1. Introduction

In recent years, we have witnessed remarkable progress in the performance of face clustering and it has been successfully applied in many real-world application scenarios. Large-scale well-labeled data is the key to the success of high performance face recognition systems. However, well-labelled data increasingly relies on large-scale data. Face clustering is a very effective method to solve a large number of unlabeled data.

So far, there are many face clustering methods, these methods can be roughly divided into two categories, respectively supervised face clustering and unsupervised face clustering. Unsupervised methods mostly rely on certain assumptions, for example, K-means [1]requires that the distribution of clusters be convex, and DBSCAN [2] requires that the density of clusters must be greater than a certain threshold. Thus, unsupervised methods lack the ability of deal with complex cluster structures. The research work of clustering algorithm based on deep learning shows that supervised face clustering is a promising method, which can effectively improve the clustering effect and significantly improve the performance. The existing supervised face clustering methods almost all benefit from the representation ability of graph convolution and obtain better performance in large-scale face clustering data baseline. CDP [3] proposes a supervised Metric for face clustering to partially solve the problems of unannotated data with complex internal structure, dependence on specific metrics, and lack of Outlier control. L-GCN [4] converted face clustering into linkage prediction problem for processing, and the whole graph was divided into some connected components, and each connected component was a cluster of clustering results. Lei et al. [5] proposed GCN-V+E clustering algorithm, GCN-V+E uses two GCN to complete face clustering, one GCN to estimate the confidence of nodes, the other to measure whether there is a connection between nodes. However, firstly, previous face clustering methods (both supervised and unsupervised) extract features through common convolutional neural networks. The features extracted by this method are difficult to obtain the global information of the image. Inspired by the success of ViT (Vision Transformer) [6] models. We design a simple ViT model to extract facial image features. Secondly, it is worth noting that when the scale of training

+ Corresponding author. Tel:+86 551 65108582.
E-mail address: dingzhuanlian@163.com.

data is large, the graph constructed by KNN will be very large, and the input of this large graph into GCN [7] or HGCN model will be out of memory. L-GCN is a typical method to train with subgraphs, but this method will produce many overlapping subgraphs. To this end, we designed an algorithm to partition a large graph into subgraphs. This algorithm not only allows the model to be well trained, but also the partitioned subgraphs are non-overlapping. Finally, graph convolutional neural network (GCN) is a model that applies deep neural network to graph structure data. To get deeper feature embedding, you have to stack multiple layers of GCNS, but stacking more layers into GCNs will encounter the vanishing gradient and network degradation problem, finally the features of graph nodes converges to the same value [8]. Therefore, in order to learn the graph structure data of high-order deep embedding more effectively, instead of graph convolution network (GCN), we use hypergraph convolution network (HGCN) [9] to obtain higher order graph embedding data.

The main contributions are in three aspects: (1) We design a simple ViT model to extract facial image features, which is able to capture the global features of the image. (2) An algorithm is proposed to partition the large graph into non-overlapping subgraphs. (3) Instead of graph convolution network, hypergraph convolution network is used to embed higher-order graph structure data.

2. Related Works

2.1. Unsupervised Face Clustering

We first briefly introduce face clustering methods, then give a brief introduction to graph convolutions, hypergraph convolutions, and transformer.

Early face clustering generally adopts LBP (Local Binary Pattern), HOG (Histogram of Oriented Gradient) and other methods to manually make features, which is out of date and out of the scope of our discussion. For example, Cui et al [10]. use LBP features. In recent years, deep learning has become a hot research direction due to its high performance, and face clustering in the era of deep learning generally uses features extracted from convolutional neural networks (CNN) [11].Existing unsupervised face clustering including K-means [1], DBSCAN [2], Spectral [12], Hierarchical Agglomerative Clustering(HAC) [13], Rand Order [14], etc. are based on features extracted from CNN for clustering. K-means clustering, spectral clustering and DBSCAN usually rely on making different assumptions on the data, which are difficult to meet the complex application scenarios in the real world. K-means assumes a spherical distribution within intra class, and that each class has the same variance. DBSCAN calculates empirical density, designating a cluster as a dense area in the data space. However, DBSCAN cannot reflect high-dimensional data well, and if the density of data sets is uneven or the cluster spacing is very different, the clustering quality is poor. Spectral cluster is widely used in clustering due to its excellent performance. Compared with other unsupervised clustering, Spectral cluster has no requirements on data structure assumptions. However, Spectral cluster is sensitive to parameter selection. Hierarchical Agglomerative Clustering (HAC) algorithm is a good clustering algorithm, but it tends to form chain clusters and is sensitive to outliers, which affects the clustering effect. Rank-order proposed a relational measure based on local context, but the time complexity of this clustering algorithm is high and its time complexity is $O(N^2)$. The Approximate Rank-Order clustering algorithm (ARO) [11] proposed an approximate order measure to reduce operating costs. Compared with other clustering algorithms, ARO is much more efficient because its computational complexity is only $O(kN)$.

2.2. Supervised Face Clustering

With the development of graph convolution network and its successful application in the field of computer vision [15], face clustering algorithm based on graph convolution network has become the main research trend at present. Firstly, the graph is constructed according to the similarity of the sample spatial data, and then the constructed graph is used as the input of the graph convolution model. Therefore, the clustering problem is transformed into the graph problem, and finally the clustering result is formed. CDP [3] proposes a supervised Metric for face clustering to partially solve the problems of unannotated data with complex internal structure, dependence on specific metrics, and lack of Outlier control. L-GCN [4] proposed a link-based GCN to predict the connection between the central axis node and its neighbors through the

embedding of enhanced edges. GCN-V+E [5] uses two stacked GCNs to estimate vertex confidence and construct edges from these high confidence levels. DA-GCN [16] uses GCN to aggregate local features, and then uses density chains (from low density to high density face chains) to combine non-local information. In addition, some methods involve new clustering frameworks. After constructing clustering results with traditional methods, additional post-processing models are used to obtain more accurate clustering results. DS-GCN [17], a typical work of this approach, designed a partitioning algorithm to generate multiscale subgraphs and proposed a two-stage supervised framework to determine the desired clusters from them.

2.3. Graph Convolution Network

Graph is a classical non-European data structure that is very common in the real world. For example, the reference relationship between the representation papers can be represented by a graph, the relationship between the representation users can be represented by a graph, and the network of protein molecules in biology can be represented by a graph. Graph neural network is the most classical and common method to deal with this kind of non-Euclidian structure data. Graph convolutional network is a kind of graph neural network. It can be regarded as extending the convolution operation of CNN to deal with graph structured data. Graph Convolutional Networks have also been successfully applied to different tasks [18]. Graph Convolutional Networks (GCNS) [7] has achieved excellent performance in Graph structure data. According to the definition of graph data convolution, GCNs can be divided into spectral method and spatial method. Spectral GCNs mainly uses graph Fourier transform to carry out convolution generalization [22], while the core idea of spatial method is to aggregate information of adjacent nodes[24].Some recent work has extended GCNs to work with large graph. Hamilton et al. [19] proposed GraphSAGE algorithm, the main idea of which is to sample a fixed number of neighbors in each layer for aggregation. FastGCN proposed by Chen et al. [25] further reduces the computational cost by sampling vertices instead of neighbors. Cluster-GCN [26] is a training algorithm based on subgraph, which is constructed by clustering on the whole graph.

2.4. Hypergraph Convolution Network

Graph neural network has attracted wide attention in various research fields and has made remarkable achievements. Most algorithms assume that the nodes in the graph are paired, that is, an edge can only connect two nodes. However, in many practical applications, the relationship between objects is high order, beyond the pair relationship, which may lead to uneven data distribution, unable to accurately describe the relationship between objects and other problems. We call the edge of the hypergraph a hyperedge. A hyperedge can contain any number of vertices. Hypergraph learning is the first introduction proposed in a pioneering work [27]. and has been introduced for encoding high-order correlation in many applications, such as image retrieval, image segmentation, 3D model classification. Inspired by the remarkable success of GCNs Models, many hypergraph neural networks are proposed to obtain the representation of hypergraphs. HGNN [28] first used Laplacian hypergraph operator to represent hypergraph from the spectral perspective. HyperGCN [29] trains GCNs on hypergraphs using the spectral theory of hypergraph. HNHN [30] designed a convolutional network combining hypernodes and Hyperedge features, which was based on the normalization of hypergraphs with cardinality. Zhang et al. [31] proposed a learnable Hypergraph Lapalcian module for updating the hypergraph topology during training.

2.5. Transformer and Its Variants

Transformer is a classic NLP model introduced by the Google team in 2017[32].The Transformer model uses the Self-Attention mechanism instead of RNN's sequential structure, which allows the model to be trained in parallel and to have global information. As ViT [6] has been successfully applied to image tasks, Transformer's development in image is becoming increasingly rapid, and the Transformer system in visual tasks is becoming rich increasingly. Although ViT has achieved considerable performance in image classification tasks, the performance of semantic segmentation for high-resolution (e.g. 800×1333) intensive image tasks such as object detection is not good, mainly because the feature output of each level is single-scale and low-resolution. PVT [33] and PVT-v2 [34] introduce different sizes of the patch embedding layer to ensure that the resolution of the feature map decreases successively.

3. Methodology

The framework of our proposed method for face clustering is mainly divided into four parts, including feature extraction, big graph partition, hypergraph convolution network and training. We will introduce different modules in details in this section.

3.1. Feature Extraction by ViT

As we all know, previous face clustering methods for face feature extraction are based on CNN. In this subsection, we introduce face features extracted by ViT in detail. Fig.1 also shows the general process of feature extraction. Suppose the size of the input each image is $I^{H\times W\times C}$. H,W and C represent the height, width and number of channels of the picture respectively. First, the patch embedding divides each picture into patches of fixed size. The patch size is $p\times p$. Therefore, each image will generate ${}^{H\times W}/_{p^2}$ patches, which can be represented as T and the dimension of each patch is $p\times p\times C$,which is represented as D. The dimension of the linear projection layer is D, so the dimensions of the input through the linear projection layer is still $T\times D$. In addition, a special character need be added in to $T\times D$, and the final dimension is $(T+1)\times D$. Positional encoding is standard learnable $1D$ position embeddings. Position encoding can better capture the information between each patch. After passing through the positional encoding module, the dimension of output is unchanged, and still is $(T+1)\times D$.

$$Attention(Q,K,V)=Softmax\left(\frac{(QK^T)}{\sqrt{d}}\times V\right) \tag{1}$$

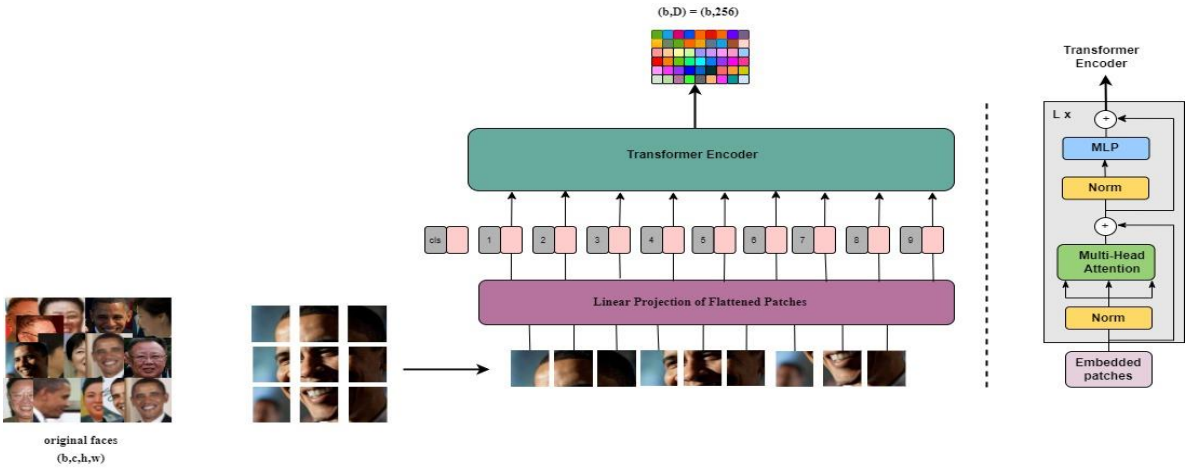

Fig. 1: The process of the ViT extract face image features.

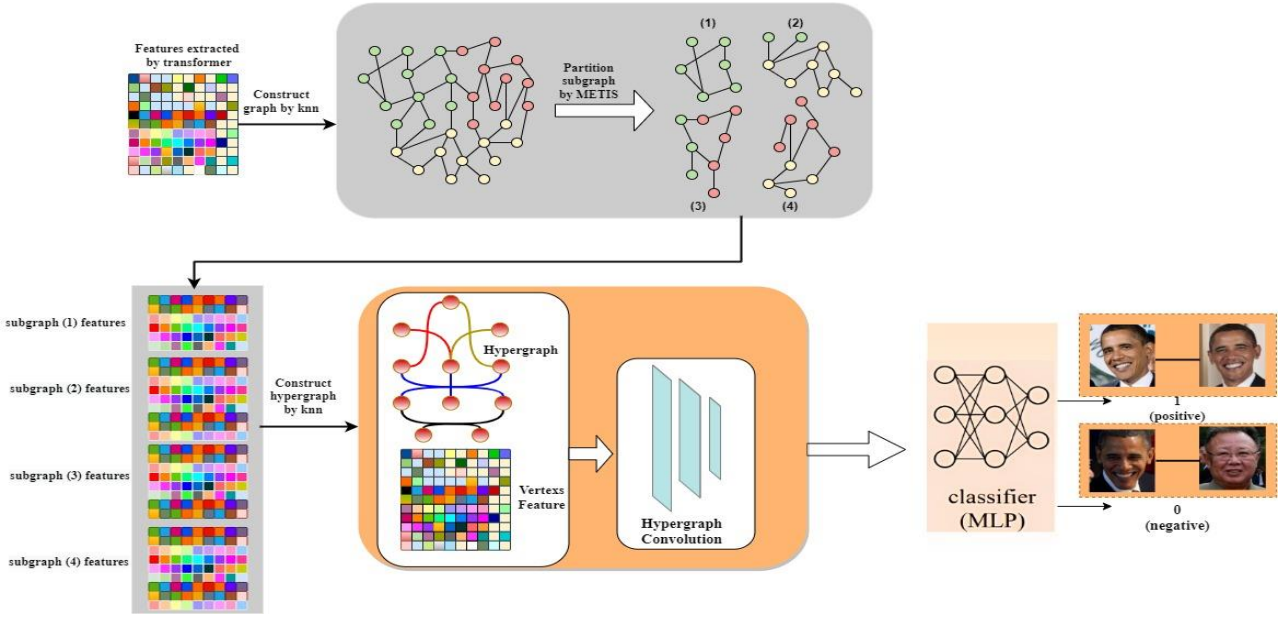

Fig. 2: The process of the ViT extract face image features.

If there's a head, the dimension of the input to Q, K, V is $(T+1) \times D$, according to Equation 1. The dimension of the final output is also $(T+1) \times D$. For multi-head attention, we suppose the number of heads is h, so we map the input dimension $(T+1) \times D/h$ to Q, K, V. Finally, h groups of q, k *and* v outputs are concated together, and the output dimension is $(T+1) \times D$, according to Equation 2.

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)$$
$$wherehead_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (2)$$

We need to reshape the feature of the multi-head attention output into a one-dimensional tensor. It is worth noting that since the dimensions of the reshaped features (i.e the one-dimensional tensor) are very large, we need to use MLP to narrow the features to get the features we want, according to Equation 3.

$$X = MLP\left(Reshape\left(MuliHead(Q,K,V)\right)\right) \quad (3)$$

3.2. Partition Large Graph

For the global-based method [5], the memory threshold of GPU may be exceeded due to the excessively large affinity graph. Although local based methods [4] [35] can reduce the storage burden of GPU through local operations on graphics, their dependence on a large number of overlapping subgraphs seriously affects the efficiency and accuracy. Therefore, we use Metis [36] to divide the large graph into non-overlapping subgraphs.

We use ViT to extract feature $X \in \mathbb{R}^{N \times F}$ of face image, and use KNN to construct a large graph G_L.We partition the graph G_L into t parts by using METIS. Subgraphs are represented as $[g_1, g_2, \dots, g_t]$. The advantage of using METIS is better to capture the clustering and community structure of the graph [26]. For example, as shown in Fig.2, the graph we constructed with KNN is divided by METIS into four non-overlapping parts.

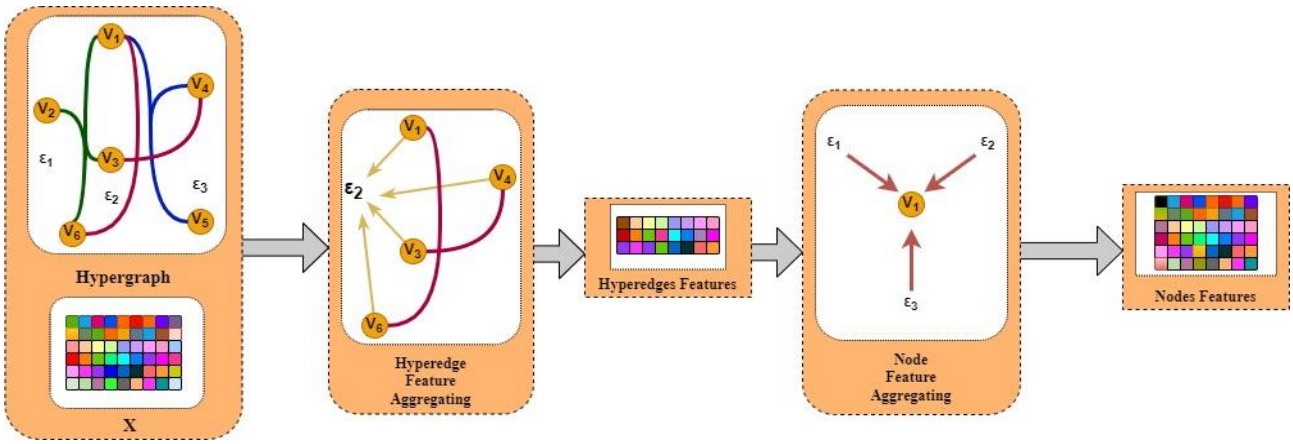

Fig. 3: The illustration of the hyperedge convolution layer.

3.3. Subgraph-Based Hyper Graph Convolution

The core work of graph convolutional networks is to aggregate the embedded information of neighbor nodes, and are widely used to analyse structured data. However, in many practical applications, the relationship between objects is high order and goes beyond the pairwise relationship, which a graph be not competent to model. So it limits the scalability of GCN for more complex structures, while simply stacking GCNS can lead to serious over-smoothing problems, making the effect of related tasks degrade sharply [8]. To this end, hypergraph convolution [28] was introduced to encode higher-order relationships between nodes. Different from graphs, in a hypergraph, a hyperedge is generally composed of any number of nodes, and Hypergraph convolution aggregates node information through node aggregation and hyperedge aggregation operations. In this paper, after obtaining the subgraph G by Metis and its corresponding initial features from the subgraph, we can construct its hypergraph H. Specifically, for each node v in the subgraph G, k-1 nearest neighbors of each node v are calculated separately through node features. These neighbour's vertices, along with the vertex v, form a hyperedge of the hypergraph H. In this way, the hypergraph H consists of the same set of nodes as the graph G and the set of hyperedges constructed by the KNN method on these nodes. We

can use the mathematical incidence matrix $A \in \mathbb{R}^{N\times M}$ to represent a hypergraph, and each entry in the incidence matrix A can be defined as:

$$A_{i,j} = \begin{cases} 1, if\ \ node\ \ v_i\ \ \in\ hyperedge\ \ \varepsilon_j \\ 0, if\ node\ \ \ v_i\ \ \notin\ hyperedge\ \varepsilon_j \end{cases} \tag{4}$$

Convolution operations can be performed on the constructed hypergraph structure. The reason why the features obtained after hypergraph convolution have higher-order correlation characteristics of data is that HGCN can convert the node features into hyperedge features, and then aggregate the hyperedge features into node features. Specifically, as shown in Fig.3, curves of different colour represent different hyperedges. Taking hyperedge ε_2 as an example, in the t-th layer of convolution, the features of the four nodes connected by the red curve (e.i. ε_2) are first aggregated as the features of ε_2 by multiplying the transpose of the incidence matrix $A \in \mathbb{R}^{N\times M}$with the input features $X_h^{(l)} \in \mathbb{R}^{N\times F^{(l)}}$. The process of this aggregation operation can be represented as $A^T X_h^{(l)}$ by mathematical expression. Then the features of the three hyperedges connected to the node v_1 are re-aggregated into the representation of v_1 by multiplying again by the incidence matrix A. It is worth noting that the weight matrix $\Lambda \in \mathbb{R}^{M\times M}$ must be added between the two multiplication operations to give different confidences to the hyperedges, i.e., $A\Lambda A^T X_h^{(l)}$. Therefore, through the above aggregation operations, vertex v_1 has the relevant features of the other five nodes. In general, the hyperedges with larger weights deserve more confidence in such an aggregation. Thus, one step of hypergraph convolution is defined as:

$$X_h^{(l+1)} = \sigma\left(A\Lambda A^T X_h^{(l)} W_h^{(l)}\right) \tag{5}$$

where $\sigma(\cdot)$ represents nonlinear activation function, and $W_h^{(l)}$ is the trainable weight matrix in the *l*-th layer convolution. Considering the scale change of $X_h^{(l)}$, same as [9], we adopt the following symmetric normalization form:

$$X_h^{(0)} = X, X_h^{(l+1)} = \sigma\left(D_h^{-1/2} A\Lambda P^{-1} A^T D_h^{-1/2} X_h^{(l)} W_h^{(l)}\right) \tag{6}$$

First of all, when $l = 0$, the node features of hypergraph $X_h^{(0)}$ are set to equals to the initial feature matrix $X \in \mathbb{R}^{N\times F}$. When $l \neq 0$, where $D_h \in \mathbb{R}^{N\times N}$ is degree matrix of diagonal nodes with $D_{h_{i,i}} = \sum_{j=1}^{M} \Lambda_{jj} H_{jj}$, and $P \in \mathbb{R}^{M\times M}$ is the degree matrix of the hyperedge, which can be calculated by $P_{jj} = \sum_{i=1}^{N} H_{ij}.D$ and P play a role of normalization in Equation 6.

Table 1: Data sets trained and tested on three datasets, MS-Celeb-1M, DeepFashion and IJB-B.

	datasets	**instances**	**classes**
Train	MS-Celeb-1M	0.57M	8.3k
Test	MS-Celeb-1M	0.58M	8.5k
Train	DeepFashion	25752	3997
Test	DeepFashion	26960	3984
Train	CASIA	0.45M	10k
Test	IJB-B-512	18171	512
	IJB-B-1024	36575	1024
	IJB-B-1845	68195	1845

Table 2: Comparison on MS-Celeb-1M with images count of 58.4k.

Methods/Metrics	**F_p**	**F_b**	**NMI**
K-means[1]	79.21	81.23	-
DBSCAN[2]	70.63	70.46	-
HAC[13]	67.93	67.17	-
ARO[11]	13.6	17	-
CDP[3]	75.02	78.70	94.69
DS-GCN[17]	85.66	85.52	97.04
L-GCN[4]	78.68	84.37	96.12
GCN-V[5]	87.14	85.82	96.37
GCN-V+E[5]	87.55	85.94	96.41
Ours(H-GCN)	**90.08**	**88.64**	**97.16**

3.4. Model Training

For model training, we input the set of partitioned subgraphs, the features set corresponding to the subgraphs, constructing the hypergraph function, the HGCN model, the MLP function, the cross-entropy loss function, and the ground-truth labels of the edges. In each iteration, we train only one subgraph at each time and iteration ends until all subgraphs in the set of subgraphs have been trained. In the HGCN module, we input the features corresponding to the subgraph into the function of constructing the hypergraph to obtain the hypergraph structure. We input the hypergraph structure and the corresponding node features into the HGCN model to obtain the output features of the HGCN. The output features of HGCN are input to the MLP function to predict the confidence of nodes with edge connections, and the confidence of nodes without edge connections. Finally, we use the cross-entropy loss to calculate the confidence of the predicted edge and the loss of the true label. Finally, back propagation is used to update the weights of the HGCN and the weights of the MLP. The specific details are described in Algorithm 1.

Table 3: BCubed F-score and NMI performance comparisons on IJB-B dataset, and the best performance is reported in bold.

Methods/Metrics	IJB-B-512		IJB-B-1024		IJB-B-1845	
	F_b	NMI	F_b	NMI	F_b	NMI
K-means[1]	61.2	85.8	60.3	86.5	60	86.8
ARO[11]	76.3	89.8	75.8	90.8	75.5	91.3
DBSCAN[2]	75.3	84.1	72.5	83.3	69.5	81.4
Spectral[12]	51.7	78.4	50.8	79.2	51.6	78.5
AHC[37]	79.5	91.7	79.7	92.5	79.3	92.3
AP[21]	49.4	85.4	48.4	86.4	47.7	86.9
DDC[20]	80.2	92.1	80.5	92.6	80	92.9
L-GCN[4]	83.3	93.6	83.3	94.2	81.4	93.8
Ours(H-GCN)	**85.6**	**94.5**	**83.9**	**94.3**	**83.7**	**94.1**

Table 4: Performance on DeepFashion dataset.

Methods/Metrics	F_p	F_b	NMI
K-means[1]	32.02	53.30	88.91
ARO[11]	22.54	52.77	88.71
DBSCAN[2]	25.07	53.23	90.75
CDP[3]	28.28	57.83	90.93
HAC[13]	22.54	48.77	90.44
MeanShift[23]	31.61	56.73	89.29
L-GCN[4]	30.7	60.13	90.67
DS-GCN[17]	32.52	57.52	89.54
GCN-V[5]	33.59	59.41	90.88
GCN-V+E[5]	38.47	60.06	90.50
Ours(H-GCN)	**39.30**	**61.69**	**91.19**

4. Experiment

4.1. Experiment Settings

Table 1 shows the details of these datasets. In this experiment, we use MS-Celeb-1M, DeepFashion and CASIA datasets for training of face clustering, respectively. the pictures of all our data sets are basically consistent with the ViT model, and the pre-training weights of ViT are used. In the training phase, for the MS-Celeb-1M data set, in order to construct the KNN affinity graph, we set the direct nearest neighbors's amount K = 80. In the process of partitioned graph, we set the number of partitions S = 50. When constructing the hypergraph with the features corresponding to the subgraphs, we set $\boldsymbol{K_h}$=80. For the HGCN, we set 1-layer HGCN in our experiment. For the DeepFashion dataset, we set K=5 to build a KNN affinity graph and in the process of graph partition we set the number of partitions S = 5. When constructing the hypergraph, we set $\boldsymbol{K_h}$=10. Different from MS-Celeb-1M, for HGCN, we use 1-layer HGCN. For the CASIA dataset, direct nearest neighbors's amount of constructing the KNN affinity graph and layers of HGCN is same as MS-Celeb-1M. In the graph partition stage, we set the number of partitions S = 40.

Algorithm 1: Subgraph-base Train for Face Clustering based Hyper-GCN.

Input: Subgraph Set; Subgraph Feature Set; HGCN model $M_h(\cdot)$; cross-entropy function *Loss* $(\cdot)$; Train iterator number Epochs; construct hypergraph function Chf$(\cdot)$; edges ground-truth label *y*; MLP model *MLP*$(\cdot)$;

Output: $W_h(\cdot)$ and W**;**

for g_i,f_i in Subgraph Set,Subgraph Features Set **do**
  $g_h \leftarrow$ Construct hypergraph by Chf(f_i);
  for each $i \in [1, Epochs]$ **do**
    $f = M_h(g_i, f_i)$;
    $Loss = Loss(MLP(f, y))$;
    update HGCN weight W_h and MLP weight W according to the gradient ∇_W loss;
  end for
end for
return $W_h(\cdot)$ and W;

4.2. Evaluation

The proposed face clustering approach is compared with supervised face clustering and unsupervised face clustering algorithms. We compare the clustering results of the above algorithms on different datasets using Pairwise F-score, BCubed F-score and NMI. The comparison results are shown in Table 2, Table 3 and Table 4 and the best results are marked in bold. The first four algorithms in Table 2 show the unsupervised clustering algorithm, while the last five algorithms show the supervised clustering algorithm. Table 2 shows the comparison results of the MS-Celeb-1M dataset, and the state-of-art performance is obtained by comparing with four classical unsupervised algorithms and five supervised classical algorithms respectively. In the unsupervised clustering algorithm, K-means shows obvious advantages compared with other unsupervised clustering algorithms. This is mainly because the K-means algorithm is based on too many assumptions that are difficult to satisfy in the real world. The last five in Table 2 are supervised face clustering. L-GCN can achieve good performance in supervised face clustering, but this method will produce many overlapping subgraphs. GCN-V+E has the highest accuracy compared with other four supervised clustering algorithms. However, GCN-V+E needs to input a large image directly into the GCN model during training, which is easy to cause out of memory. The method shown in the last row of Table 2 is our proposed method H-GCN. The experiments show that H-GCN achieves the best clustering results on MS-Celeb-1M dataset. Table 4 shows the results of experiments on the DeepFashion dataset. Our method is more accurate than the most advanced methods. The test results of IJB-512, IJB-1024 and IJB-B-1845 are shown in Table 3. We only used metrics of BCubed F-score and NMI to evaluate our results, and compared with the best method, our method can achieve better results. H-GCN uses ViT to extract image features. Different from CNN, the features extracted by ViT have global information, which can improve the clustering quality. At the same time, we use METIS to divide the large graph into non-overlapping subgraphs. It solves the training problem of out of memory. Finally, we use HGCN instead of GCN to aggregate deeper graph feature embeddings and produce good clustering results.

4.3. Ablation Study

To verify that the key modules work as expected and the validity of the model structure, and investigate some key design choices, we perform ablation experiments on the MS-Celeb-1M.

In order to better evaluate the role of each sub-module, we carried out end-to-end experiments using each sub-module separately. In Table 5, we present the results of the ablation experiment on the MS-Celeb-1M dataset. We maintain the consistency of HGCN model or GCN model, and use ViT and CNN respectively to extract features. From the experimental data, the feature extracted by ViT has higher accuracy. At the same time, we use the same feature extractor VIT or CNN, respectively using HGCN model and GCN to aggregate node features. Using HGCN model to cluster faces can achieve better clustering effect.

Table 5: Compatibility analysis on MS-Celeb-1M.

Dataset	MS-Celeb-1M		
Methods /Metrics	$\mathbf{F_p}$	$\mathbf{F_b}$	**NMI**
ViT HGCN	**90.08**	**88.64**	**97.16**
CNN HGCN	88.80	86.59	96.34
ViT GCN	83.66	82.46	96.01
CNN GCN	81.73	79.47	95.56

5. Conclusion

HGCN have proven to be useful tool in many pioneer works. In this paper, we try to improve the performance of face clustering, and propose a novel supervised face clustering framework. The proposed method is based on HGCN to better aggregate the feature embedding of deeper nodes and remarkably improves accuracy. At the same time, in order to eliminate the limitation of being unable to train due to memory overflow on the large graph, we use METIS to partition the large graph as subgraphs. We use HGCN to learn the deeper information from these subgraphs to do clustering. Besides, in order to further improve the performance of face clustering, we use the most advanced feature extractor (ViT) to extract face features. The effectiveness of the proposed H-GCN clustering approach is tested on benchmark datasets. Comprehensive experiments indicate that our method is superior to other popular clustering algorithms on large-scale face datasets.

Acknowledgements

This work was supported by NSFC (62076005, 61906002, U20A20398), NSF of Anhui Province (2008085MF191, 2008085QF306, 2008085UD07), and the University Synergy Innovation Program of Anhui Province (GXXT-2021-002).

References

[1] S. P. Lloyd, Least squares quantization in pcm, IEEE Trans. Inf. Theory 28 (1982) 129–136.

[2] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Knowledge Discovery and Data Mining, 1996.

[3] X. Zhan, Z. Liu, J. Yan,D. Lin, C. C. Loy, Consensus-driven propagation in massive unlabeled data for face recognition, ArXiv abs/1809.01407 (2018).

[4] Z. Wang, L. Zheng, Y. Li, S. Wang, Linkage based face clustering via graph convolution network, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 1117–1125.

[5] L. Yang, D. Chen, X. Zhan, R. Zhao, C. C. Loy, D. Lin, Learning to cluster faces via confidence and connectivity estimation, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 13366–13375. Title Suppressed Due to Excessive Length 15.

[6] A.Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, ArXiv abs/2010.11929 (2020).

[7] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, ArXiv abs/1609.02907 (2016).

[8] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: AAAI Conference on Artificial Intelligence, 2018.

[9] S. Bai, F. Zhang, P. H. S. Torr, Hypergraph convolution and hypergraph attention, Pattern Recognit. 110 (2019) 107637.

[10] J. Cui, F. Wen, R. Xiao, Y. Tian, X. Tang, Easyalbum: an interactive photo annotation system based on face clustering and re-ranking, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2007).

[11] C. Otto, D. Wang, A. K. Jain, Clustering millions of faces by identity, IEEE Transactions on Pattern Analysis and

Machine Intelligence 40 (2016) 289–303.

[12] J. Shi, J. Malik, Normalized cuts and image segmentation, Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1997) 731–737.

[13] W.-A. Lin, J.-C. Chen, R. Chellappa, A proximity-aware hierarchical clustering of faces, 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017) (2017) 294–301.

[14] C. Zhu, F. Wen, J. Sun, A rank-order distance based clustering algorithm for face tagging, CVPR 2011 (2011) 481–488.

[15] R. Li, S. Wang, F. Zhu, J. Huang, Adaptive graph convolutional neural networks, ArXiv abs/1801.03226 (2018).

[16] S. Guo, J. Xu, D. Chen, C. Zhang, X. Wang, R. Zhao, Density-aware feature embedding for face clustering, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 6697–6705.

[17] L. Yang, X. Zhan, D. Chen, J. Yan, C. C. Loy, D. Lin, Learning to cluster faces on an affinity graph, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 2293–2301.

[18] S. Yan, Z. Li, Y. Xiong, H. Yan, D. Lin, Convolutional sequence generation for skeleton-based action synthesis, 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (2019) 4393–4401.

[19] W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NIPS, 2017.

[20] W.-A. Lin, J.-C. Chen, C. D. Castillo, R. Chellappa, Deep density clustering of unconstrained faces, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018) 8128–8137.

[21] B. J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (2007) 972 – 976.

[22] J. Bruna, W. Zaremba, A. D. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, CoRR abs/1312.6203 (2013).

[23] D.Comaniciu, P. Meer, Mean shift analysis and applications, Proceedings of the Seventh IEEE International Conference on Computer Vision 2 (1999) 1197–1203 vol.2.

[24] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', Y. Bengio, Graph attention networks, ArXiv abs/1710.10903 (2017).

[25] J. Chen, T. Ma, C. Xiao, Fastgcn: Fast learning with graph convolutional networks via importance sampling, ArXiv abs/1801.10247 (2018). 16 F. Author et al.

[26] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019).

[27] D. Zhou, J. Huang, B. Sch ̈olkopf, Learning with hypergraphs: Clustering, classification, and embedding, in: NIPS, 2006.

[28] Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in: AAAI Conference on Artificial Intelligence, 2018.

[29] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, P. P. Talukdar, Hypergcn: A new method for training graph convolutional networks on hypergraphs, in: Neural Information Processing Systems, 2018.

[30] Y. Dong, W. Sawin, Y. Bengio, Hnhn: Hypergraph networks with hyperedge neurons, ArXiv abs/2006.12278 (2020).

[31] J. Zhang, Y. Chen, X. Xiao, R. Lu, S. Xia, Learnable hypergraph laplacian for hypergraph learning, ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2021) 4503–4507.

[32] A.Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, ArXiv abs/1706.03762 (2017).

[33] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, L. Shao, Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021) 548–558.

[34] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, L. Shao, Pvtv2: Improved baselines with pyramid vision transformer, ArXiv abs/2106.13797 (2021).

[35] C. Qi, J. Zhang, H. Jia, Q. Mao, L. Wang, H. Song, Deep face clustering using residual graph convolutional network, Knowl. Based Syst. 211 (2021) 106561.

[36] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1998) 359–392.

[37] A.K. Jain, R. C. Dubes, Algorithms for clustering data, 1988.