




TSC-GCN: A Face Clustering Method Based on GCN

Jinmin Xue¹, Shengdong Qu¹, Jingxian Li¹, Yan Chu¹ ,
and Zhengkui Wang²

¹ Harbin Engineering University, Harbin 150001, China
{xuejinmin, lijingxian, chuyan}@hrbeu.edu.cn

² Singapore Institute of Technology, Singapore, Singapore
zhengkui.wang@singaporetech.edu.sg

Abstract. For face clustering, the density of each cluster distribution in the feature space is different. Too high similarity pruning will lead to the sparse clustering not being divided and reducing the recall ratio, while too low similarity pruning will lead to the decline of clustering accuracy. We propose the Two-Stage Clustering Method Based on Graph Convolutional Neural Network (TSC-GCN), in which the clustering size are set to measure the sparse degree of clustering and pruning with a low similarity degree. The clustering with sparse distribution is screened out, then the requirements for nodes similarity are improved. At the same time, the number of neighbor nodes is set to prevent the clustering core from deviating from the aggregation of nodes, and the clustering with dense distribution is screened out. The experimental results show that TSC-GCN can give a good consideration of the accuracy and recall ratio both, and achieve better clustering effectiveness than the state-of-the-art methods.

Keywords: Graph convolutional network · Facial clustering · Pruning screening

1 Introduction

Recently, there have been a large amount of face images generated daily due to the popularity of the cameras. This has unfortunately incurred big challenges for the face image management systems. Face clustering has become a very common approach to manage the face images for the purpose of face recognition or face labelling, etc. [1–4]. Traditional clustering methods rely on specific assumptions. For example, K-Means [5] requires the data set to be convex-shaped, and spectral clustering [6] requires different clusters with similar sizes. They all lack of the capability to deal with face images with complex data distributions. In order

Supported by Fundamental Research Funds for the Central Universities under Grant No. 3072022TS0601 and National Key Laboratory Foundation of Underwater Measurement and Control Technology.

to improve the adaptability of complex data distributions, recent research has proposed clustering methods to learn cluster patterns based on GCN link prediction, and show great performance improvement over traditional methods in terms of accuracy [2, 7]. These link prediction-based methods improve the performance of face clustering and have fewer requirements on the data distribution. They all share some common operations, including generation of the subgraph for each instance as central node first followed by predicting the linkage probability between the central node and its neighbors. In the end, they organize the instance into clusters by the linkage. However, though these approaches are effective enough to identify more accurate clusters, they have resulted in high computation overheads. The main computation bottleneck of the approaches is the need to get individual subgraphs or sub-networks from its local context for each image instance. This has unfortunately resulted in a huge number of subgraphs generations and led to further heavy learning tasks as well. We have observed that the generated subgraphs are usually highly overlapping. A central node could be a neighbor node in other subgraphs which results in excessive redundant calculation costs, and at the same time, the inference speed is reduced.

The contributions of our paper are summarized as follows:

- (1) We propose a new GCN-based face clustering TSC-GCN, which is both time-efficient and effective.
- (2) We conducted extensive experiments to evaluate TSC-GCN against existing methods in IJB-B face datasets.

Section 2 shows the related work about the graph convolutional neural networks and face clustering. Section 3 introduces the details of the proposed TSC-GCN. Section 4 and 5 provide the experimental evaluations and conclusions respectively.

2 Related Work

Graph Convolutional Neural Network has an excellent performance in processing graph data without regular spatial structure, so it is widely used in classification and link prediction tasks. Yao et al. [9] used an entire corpus to manually construct large heterogeneous graphs and learned them by GCNs to greatly improve text classification performance. Chu et al. [10] used Bilinear Convolutional Neural Network to fuse the high and low dimensional features. We use GCN to mine the complex relationships embedded between nodes and infer connectivity.

Unsupervised face clustering is to make face images similar within the class, and mutually exclusive between the classes. Traditional face clustering algorithms do not rely on machine learning and are implemented by manually designing clustering rules, such as K-means [5] and DBSCAN [11]. Vidal et al. [12] proposed Sparse Subspace Clustering, which utilized the bottom line subspace structure in data. Lin et al. [13] used linear support vector machines to design the nearest neighbor similarity measure based on data samples. Shi et al. [14]

proposed Conditional Pairwise Clustering, which defines the Clustering problem as a Conditional random field and uses Pairwise similarity between faces to complete Clustering.

In supervised face clustering, Tapaswi et al. [15] proposed ball cluster learning, which divided feature space into balls with equal radii, and each ball represents a cluster. Meanwhile, the relationship between balls is constrained. Wang et al. [2] proposed a graph convolution face clustering algorithm based on link prediction, which used GCN to capture the local context of face data and accurately judge the link possibility of face pairs. Lei et al. [7] designed two GCN modules, which were respectively used to detect high-quality cluster proposals and eliminate noise in them to obtain high-quality clustering results.

Though supervised face clustering has been proven to be more effective, it suffers from high computation overhead, as they all need to generate a lot of subgraphs for connectivity learning. In contrast, our proposed method divides face data by density and only predicts the connectivity of the low-density part. The high-density part is naturally connected, which speeds up the operation with good accuracy.

3 TSC-GCN

3.1 Problem Description

Due to the development of neural networks, feature extraction has been able to extract high-dimensional features. However, the clustering results are unsatisfactory in calculating the difference between the feature vectors of two face images. Thus, the similarity between face images is calculated by constructing subgraphs. In the node subgraph, the connection relationship between connected nodes is represented by an adjacency matrix. GCN can aggregate neighborhood information to obtain embeddings. Furthermore, the embeddings of the node subgraph are obtained and are used to obtain the similarity between nodes through the classification function. The larger similarity value means that they are more likely to belong to the same cluster.

When the similarity between two nodes is greater than or equal to a threshold, the two nodes are in the same cluster, and an edge can be added in. We evaluate algorithm results by using pseudo labels, which are not the real identity labels of nodes, but are temporarily assigned to nodes in the same cluster. As shown in Fig. 1, face images with the same pseudo-label are in the same cluster, and y_i is pseudo labels, where i represents the i -th cluster. When evaluating the clustering, the pseudo labels are compared with the real labels, and the PRECISION, RECALL, PAIRWISE F-score, and NMI are calculated.

3.2 Framework Overview

This method calculates the similarity through GCN and completes clustering in two stages, which is named as Two-Stage Clustering Method Based on Graph Convolutional Neural Network (TSC-GCN). The framework is shown in Fig. 2.

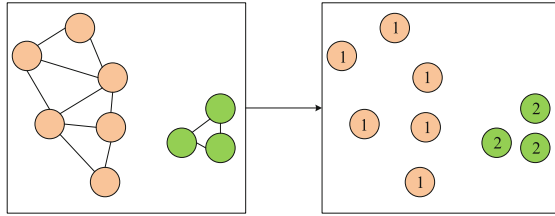


Fig. 1. Pseudo labels as the substitutions of the original labels

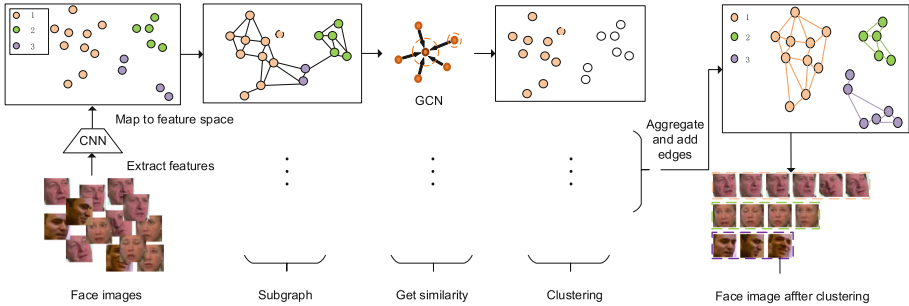


Fig. 2. The framework of TSC-GCN

Firstly, the features of the face images are extracted by the convolutional neural network, the embeddings of the face images are obtained, and then the obtained feature vector is mapped to the feature space. The face images in the feature space are regarded as nodes, and each node in the feature space is regarded as the center node. In order to use the topology of neighbor nodes to represent the center node, a subgraph corresponding to the center node is constructed according to the topology information of the neighbor nodes of the center node. Secondly, the node subgraph is used as the local data to be processed by the graph convolutional neural network, the embeddings of the node subgraph is implemented, and the similarity between the nodes is obtained through the classification. Using the similarity, we can get clusters in two stages. Finally, all the clustering results are merged and the nodes in a cluster are edged to obtain the classified face images.

3.3 Face Image Feature Representation

TSC-GCN uses the method provided in the ArcFaces face recognition model [20] to extract the features of the image. In the face alignment, the five key points of the face are detected by MTCNN, and the cropped 112×112 -dimensional face images are obtained after normalization. The features of the face images are extracted by the resnet50 model, and we can get the 512-dimensional embeddings through the fully connected layer. The training set of the model is the joint dataset of MS-Celeb-1M [21] and VGGFace2 [22]. IJB-B [8] is divided into three

datasets IJB-B-512, IJB-B-1024, IJB-B-1845. The images in the three data sets are input into the model to obtain the feature vector, so as to map the face images into the nodes of the feature space. Through feature extraction, the embeddings of the face image is obtained and mapped to the feature space as the face image node. In this way, the face images are transformed into a graph structure. In order to use the topology information of nodes and neighbor nodes in the feature space to construct the neighbor node subgraphs, we use GCN to calculate the embeddings of the node subgraphs. After classification, the prediction is done according to the similarity between nodes, so as to obtain the clusters.

3.4 Subgraph Representation

TSC-GCN takes each node in the feature space as the central node p , uses the topology information of the neighbor nodes of the central node p to construct the neighbor node subgraphs, and then obtains the adjacency matrix A_p and subgraph features X_p from the neighbor node subgraph. Subsequently we can get the similarity based on adjacency matrix A_p and subgraph features X_p . The construction of the neighbor node subgraph is divided into three steps: determining the subgraph nodes, constructing the subgraph features, and adding edges between the nodes.

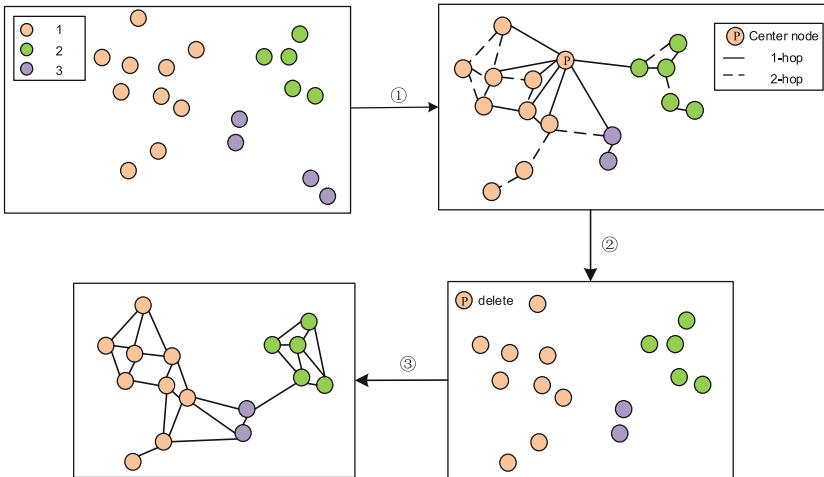


Fig. 3. The process of subgraph construction

Step 1: Identify subgraph nodes. Find the neighbor nodes of h -hop and less than h -hop of the center node p , and use all the neighbor nodes as subgraph nodes. Note that the center node p is not included in the subgraph node set. Figure 3 shows the search of 2-hop subgraph nodes. The solid lines are edges

between the 1-hop neighbor nodes and the dotted lines are edges between the 2-hop neighbor nodes.

Step 2: Build subgraph features. The subgraph features are constructed according to the face image features of the central node and the subgraph nodes. Concretely, the feature vectors of each subgraph node are subtracted from the feature vectors of the central node, and then these new feature vectors are formed as a row vector, which is the subgraph feature X_p corresponding to the central node p . The subgraph feature X_p is defined as:

$$X_p = [\dots, x_i - x_p, \dots]^T, \quad i \in V_p \quad (1)$$

where X_p is the subgraph feature, x_i is the feature of i th subgraph node, and x_p is the feature of the center node.

Step 3: Add edges between two nodes. In order to avoid large differences of each node in the subgraph. For a node a in the subgraph node set V_p , we record the U neighbor nodes closest to this node a which form a new node set V_u . If $b \in V_u \cap V_p$, we add a edge between node a and node b . Take the nodes in the subgraph node set V_p as the central node respectively, repeat the above process to add edges, and finally get the topology of the neighbor node subgraph. This topology is represented by an adjacency matrix A_p . The value of u is 3 in Fig. 3.

3.5 Similarity Estimator

Using the adjacency matrix A_p and subgraph features X_p , we can get the similarity on the graph convolution network. The input of the convolution layer is the adjacency matrix A_p and subgraph feature X_p , and the output is a transformed subgraph feature Y_p . The GCN is described as:

$$Y = \sigma([X||GX]W) \quad (2)$$

where $X \in R^{N \times d_{in}}$, $Y \in R^{N \times d_{out}}$, d_{in} , d_{out} are the input and output dimensions of the node feature respectively. $G = g(X, A)$ is a $N \times N$ dimensional aggregation matrix, where each row adds to 1. W is the weight matrix of the convolutional layer. $\sigma()$ is a nonlinear activation function. $G = g(X, A)$ adopts mean value aggregation, where $G = \Lambda^{-\frac{1}{2}} A \Lambda^{-\frac{1}{2}}$, A is the adjacency matrix, Λ is a diagonal matrix and $\Lambda_{ii} = \sum_j A_{ij}$.

3.6 Form Clusters

Pruning with too high similarity will cause sparse clusters to be unable to be divided, reducing RECALL while pruning with too low similarity will lead to the decline of clustering accuracy. TSC-GCN improves the division of dense clusters by pruning and aggregating nodes with a higher degree of similarity. In addition, aggregated nodes are filtered by setting nc (neighbor count) to prevent deviation from the cluster core when aggregating nodes. If the number of neighbor nodes is greater than or equal to nc , the node does not deviate from the clustering

core, and the node and its neighbor nodes are aggregated. TSC-GCN has two states, one is to limiting the size of clusters, and the other is to limit the number of neighbor nodes. The similarity inferred by GCN is used as the input, and the threshold of limiting the cluster size is firstly obtained to get a part of the final cluster that is relatively sparse and a part of the remaining node sets to enter the stage of limiting the number of neighbor nodes. After the stage of limiting the number of neighbor nodes, all nodes are assigned to the cluster, and the face image clustering is completed.

Limit the Cluster Size. We can set a threshold to prune the cluster. All nodes in the same cluster are connected together. However, simply using the similarity method will lead to mistakenly deleting a node that actually is in the same cluster during the pruning process, which will ultimately affect the clustering effect. Therefore, in the first stage, the pruning strategy based on the cluster set size threshold is adopted. At the same time, the similarity threshold of the nodes is gradually increased during the iterative clustering of the remaining nodes. In this way, the large set of clusters is pruned, and some clusters with less than optimal similarity representation ability are iteratively filtered out. For the remaining nodes, the value of similarity is higher, and we can take them to the stage of limiting the number of neighbor nodes for further screening.

Limiting the Number of Neighbor Nodes. In order to further distinguish which of these high similarity neighbor nodes are actually the same cluster as the given node. The remaining nodes after screening have a high similarity with the given node. In order to further distinguish between those neighbors with high similarity, which are actually in the same cluster as the given node, it is possible to introduce a variable: the number of neighbor nodes whose similarity with the given node is greater than or equal to Th_2 . The more neighbor nodes is, the higher the probability that a given node represents this cluster is. Add these qualified neighbor nodes and the given node to the cluster, store these neighbor nodes with a queue, which means check whether the number of neighbor nodes meet the requirements. If so, the neighbor nodes are added to the cluster, and these neighbor nodes are inserted into the tail of the queue. The above process is repeated for the nodes in the node queue until the queue is empty. At this time, a cluster will be obtained. Update the remaining node set and find the next cluster according to the above method. When the remaining node set is empty, the clustering ends.

The algorithm is described as Algorithm 1.

4 Experiments and Analysis

4.1 Dataset and Metrics

TSC-GCN is tested on the public face clustering benchmarks: IJB-B [8], which consists of seven different subtasks. The three subtasks with the largest number

Algorithm 1. Face Clustering Methods Based on GCN**Data:** *Imagedataset* V **Result:** *The final Clustering Results*Extract face image feature x ;Build neighbor node subgraphs, and get subgraph node set V_p ;Get subgraph features X_p using equation (1);Build adjacency matrix A_p , and add an edge between node a and node b , which $b \in V_u \cap V_P$, $A_{ab} = A_{ba} = 1$;

Calculate the similarity;

 $th1 \leftarrow q$; /* q is a predefined value */**while** *The number of nodes in set remain $\leq p \times$ the total number of nodes* **do**Put the nodes whose similarity $\geq th1$ into set S ;**if** *the number of nodes in $S \leq MAX_SIZE$* **then**Put set S into Result;Delete nodes in set S from set remain;**else**Clear set S ;Increase the value of $th1$ with a step;**end****end** $th2 \leftarrow r$; /* r is a predefined value */

Build a remaining node queue using the nodes in set remain;

while *remaining node queue is not empty* **do**Take the nodes in the remaining node queue as the queue head of the node queue and add it to the set Q ;**while** *node queue is empty* **do**Record the number of the neighbor nodes of the node whose similarity $\geq th2$;**if** *the number of the nodes $\geq nc$* **then**/* nc is a constant value */

Add the neighbor nodes to the node queue;

Add the neighbor nodes to set Q ;**end**

Get the next node in the node queue;

Add Q to set result;Delete nodes which is in Q from the remaining node queue;**end****end**

return result;

are selected. The numbers of identities included in the three subtasks are 512, 1024, and 1845 respectively, and the numbers of samples included are 18171, 36575, and 68195 respectively. We use the model trained on a random subset of CASIA [16] dataset for testing clustering. We use the most mainstream face

clustering evaluation metrics [17]: Bcubed F-score (denoted as F_B) and Pairwise F-score (denoted as F_P).

4.2 Parameter Setting

For the density selection threshold is determined to be 0.4 after many experiments. The initial learning rate of momentum and SGD is 0.1, and the weight decay value is $1e^{-5}$. The infrastructure/hardware (GPU/CPU/RAM) set up is (Tesla P100/Intel Xeon/16 GB).

4.3 Results and Analysis

Table 1 shows the comparison results of TSC-GCN and other base methods. The best results are shown in bold. The results in Table 1 show that our method is much better than traditional clustering methods in IJB-B compared to traditional methods. Specifically, TSC-GCN’s Pairwise F-score is about 4% higher than the existing method.

Table 1. Comparison on IJB-B data set.

Method	IJB-B-512		IJB-B-1024		IJB-B-1845	
	F_B	F_P	F_B	F_P	F_B	F_P
K-Means [5]	0.612	0.436	0.603	0.413	0.600	0.301
Spectral [6]	0.517	0.310	0.508	0.217	0.516	0.246
AHC [23]	0.795	–	0.797	–	0.739	–
AP [18]	0.494	–	0.484	–	0.477	–
DBSCAN [11]	0.753	–	0.725	–	0.695	–
ARO [19]	0.763	–	0.758	–	0.755	–
PAHC [13]	–	–	0.639	–	0.755	–
ConPAC [14]	0.656	–	0.641	–	0.755	–
DDC [24]	0.802	–	0.805	–	0.800	–
L-GCN [2]	0.833	0.843	0.833	0.853	0.814	0.677
TSC-GCN	0.834	0.913	0.835	0.890	0.811	0.712

4.4 Hyperparameter Analysis

We analyzed the hyperparametric similarity th1, th2, and neighbor count nc in both phases. We chose the Pairwise F-score as the evaluation metric for clustering effectiveness. Also, a comparison with L-GCN was made for reference and analysis.

The First Stage. In the first stage, i.e. the limiting clustering size stage, the value of the hyperparameter $th1$ affects the effect of clustering. As can be seen in Fig. 4, the TSC-GCN shows an overall upward trend in Pairwise F-score as the value of $th1$ increases. Once $th1$ is too high, some similar images will be excluded from the clusters.

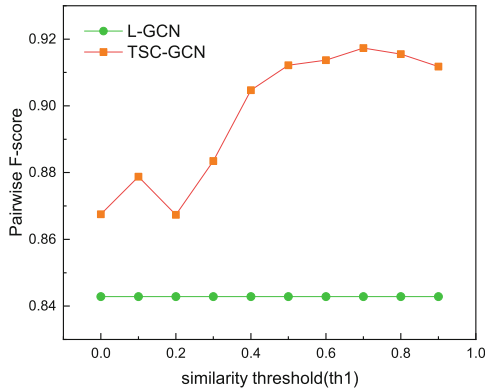


Fig. 4. Effect of $th1$ on Pairwise F-score

The Second Stage. The second stage is the combination of the two hyperparameters $th2$ and nc , which then filter the clusters. To address the shortcomings of $th1$, we exclude the interference of high similarity node pairs by setting a threshold for the number of neighboring nodes nc . Theoretically, the higher the number of surrounding neighboring nodes, the higher the confidence that the node can represent the cluster it is in. As can be seen from Fig. 5 and Fig. 6, the clustering results are relatively sensitive to the choice of the similarity threshold $th2$.

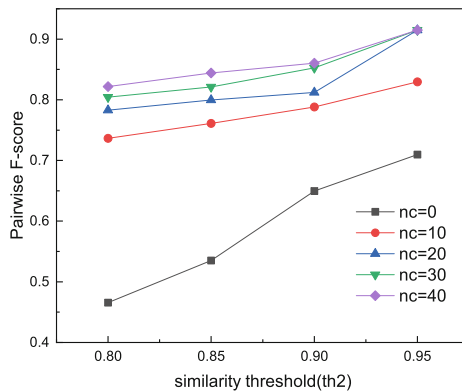


Fig. 5. Effect of $th2$ on Pairwise F-score

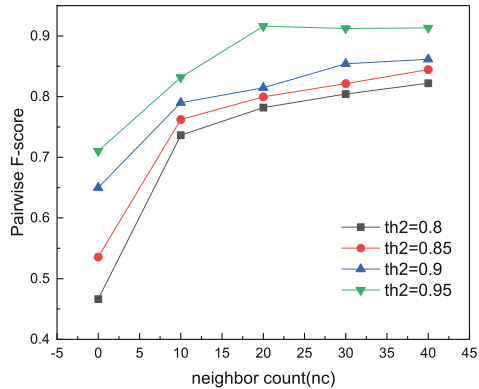


Fig. 6. Effect of nc on Pairwise F-score

5 Conclusions

We proposed a novel face clustering method TSC-GCN, which divided the process of obtained cluster results into two stages, limiting the size of clusters and the number of neighboring nodes. The purpose was to filter out the clusters that were more dispersed and were densely distributed in the feature space respectively. It was experimentally verified that the TSC-GCN could well balance precision and recall both, and achieved better clustering results when the runtime was at the same level as the existing methods.

References

1. Yang, L., Huang, Q., Huang, H., Xu, L., Lin, D.: Learn to propagate reliably on noisy affinity graphs. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020, Part XV. LNCS, vol. 12360, pp. 447–464. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58555-6_27
2. Wang, Z., Zheng, L., Li, Y., Wang, S.: Linkage based face clustering via graph convolution network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1117–1125 (2019)
3. Li, P., Zhao, H., Liu, H.: Deep fair clustering for visual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9070–9079 (2020)
4. Guo, S., Xu, J., Chen, D., Zhang, C., Wang, X., Zhao, R.: Density-aware feature embedding for face clustering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6698–6706 (2020)
5. Lloyd, S.: Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**(2), 129–137 (1982)
6. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905 (2000)
7. Yang, L., Zhan, X., Chen, D., Yan, J., Loy, C., Lin, D.: Learning to cluster faces on an affinity graph. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2298–2306 (2019)

8. Whitelam, C., et al.: IARPA Janus benchmark-b face dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 90–98 (2017)
9. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7370–7377 (2019)
10. Chu, Y., Wang, Z., Wang, L., Zhao, Q., Shan, W.: Fine-grained image classification based on target acquisition and feature fusion. In: Qiu, H., Zhang, C., Fei, Z., Qiu, M., Kung, S.-Y. (eds.) KSEM 2021, Part III. LNCS (LNAI), vol. 12817, pp. 209–221. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-82153-1_18
11. Ester, M., Kriegel, H., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231(1996)
12. Elhamifar, E., Vidal, R.: Sparse subspace clustering: algorithm, theory, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(11), 2765–2781 (2013)
13. Lin, W., Chen, J., Chellappa, R.: A proximity-aware hierarchical clustering of faces. In: The 12th IEEE International Conference on Automatic Face and Gesture Recognition, pp. 294–301. IEEE (2017)
14. Shi, Y., Otto, C., Jain, A.K.: Face clustering: representation and pairwise constraints. *IEEE Trans. Inf. Forensics Secur.* **13**(7), 1626–1640 (2018)
15. Tapaswi, M., Law, M., Fidler, S.: Video face clustering with unknown number of clusters. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5027–5036 (2019)
16. Yi, D., M., Lei, Z., Liao, S., Li, S.Z.: Learning face representation from scratch. arXiv preprint [arXiv:1411.7923](https://arxiv.org/abs/1411.7923) (2014)
17. Amigó, E., Gonzalo, J., Artilles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.* **12**(4), 461–486 (2009)
18. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. *Science* **315**(5814), 972–976 (2007)
19. Otto, C., Wang, D., Jain, A.: Clustering millions of faces by identity. *IEEE Trans. Pattern Anal. Mach. Intell.* **40**(2), 289–303 (2017)
20. Deng, J., Guo, J., Zafeiriou, S.: ArcFace: Additive angular margin loss for deep face recognition. arXiv reprint [arXiv:1801.07698](https://arxiv.org/abs/1801.07698) (2018)
21. Guo, Y., Zhang, L., Hu, Y., He, X., Gao, J.: MS-Celeb-1M: a dataset and benchmark for large-scale face recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016, Part III. LNCS, vol. 9907, pp. 87–102. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_6
22. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: VGGFace2: a dataset for recognising faces across pose and age. In: IEEE International Conference on Automatic Face and Gesture Recognition (FG) (2018)
23. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall Inc., Englewood Cliffs, New Jersey (1988)
24. Lin, W., Chen, J., Castillo, C., Chellappa, R.: Deep density clustering of unconstrained faces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8128–8137 (2018)