一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

An Effective De-noising Algorithm for Making A Large Celebrity Face Dataset with A High Purity

Zheng Ge† Quan Cui† Rong Xu‡ Masahiro Imai‡ Yoshie Osamu†

†Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan

‡Artificial Intelligence Research Div., Datasection Inc., Tokyo, Japan

E-mail: †jokerzz@fuji.waseda.jp, †cui-quan@toki.waseda.jp, ‡{jo.ei, masahiro.imai}@datasection.co.jp, †yoshie@waseda.jp.

Abstract In recent years, face recognition has been greatly improved by the development of CNN such as DeepID, FaceNet, and so on. However, the performance of those trained models is not satisfactory when applied on Asian face recognition because those models are almost trained on western face datasets. In order to solve such a problem, we create a large Chinese celebrity face dataset, including 11,289 celebrities and 395,130 face images, and then we can make a fine tune on those trained models for Asian face recognition by our created Chinese celebrity dataset.

In this paper, we make a scheme to collect celebrity names and images from unstructured data by internet. Then we propose an effective de-noising algorithm to improve the quality of dataset, and the purity of our data can reach 97.7% from original 65.9% after the de-noising. Meanwhile, the de-noising operation on MS-Celeb-1M has been realized for evaluating the proposed method, and the purity of the tested fine part of MS-Celeb-1M has been improved from 73.0% to 98.7%. Therefore, the experiments on our created Chinese celebrity face dataset and MS-Celeb-1M indicate that the proposed de-noising algorithm has achieved excellent performance for improving the quality of dataset.

Keywords CNN, Face dataset, Denoising algorithm

1. Introduction

1.1. Introduction

Face recognition has been a hot topic in computer vision for dozens of years. In order to solve this problem, many researchers have proposed various methods to improve the accuracy of face recognition. With the development of deep convolutional neural network (CNN), face recognition approaches mainly focus on how to implement a classifier based on CNNs to distinguish face identities extracted from large face datasets. Obviously, face recognition is not only to tell who the identities in images are, but also there are several extending sub-problems, such as face verification [10, 11], face alignment [6, 12] and so on.

When we plan to apply deep learning methods on Asian face recognition, we found that the models pre-trained by some popular western datasets (e.g., CASIA [1], MS-Celeb-1M[2] and so on) cannot work as good as described in those related papers based on practice. Intuitively, the performance of the pre-trained models depends on what face dataset they learn. However, most public face datasets are mainly composed of American and European people, where Asian faces are just a small part of the datasets. It inspires us to build a clean dataset for Asian. Based on such an Asian face dataset, we can make a well fine-tuned model to perform better on Asian face recognition than those trained on western face datasets. As the first step, herein we focus on how to create and de-noise a novel Chinese celebrity face dataset.

In this paper, we proposed a practical idea to create a face dataset and a noise filter algorithm to improve the purity of the collected dataset. The final purpose of our research is to improve the face recognition performance on Asian faces. Our key objectives in this research are: 1. “As specific as enough”, we want to build a dataset that has some information which other datasets might not have. We will notify the specific name and nationality of each kind of face. The reason for this is that we hope our dataset can provide as much information as possible. 2.” As pure as enough”, we try our best to improve the purity of our dataset, because a dataset with a high purity can improve the performance of the deep learning methods. For these purposes, we propose a noise filter algorithm for purifying face datasets based on a feature extraction

convolutional neural network and a graphic cluster algorithm.

We have collected 395,130 images of 11,289 Chinese celebrities in our dataset based on the proposed algorithm in this paper. Once the dataset is applied for fine-tuning a face recognition model, the performance of Asian face recognition can be greatly enhanced. Our main contributions in this work are summarized as follows:

- Propose a novel Chinese celebrity face dataset to solve Asian face recognition problems.
- Propose a feasible idea to collect the celebrities' names and images.
- Propose a de-noising algorithm to create a face images dataset with a high purity.

1.2. Related Work

In the CASIA-WebFace database, they collect images from IMDB and chose one 'main photo' for each celebrity, then extract feature by a pre-trained face recognition engine, finally calculate similarities between photos in gallery and the selected main photo to de-noise. UMDFace[3] also use CNN to extract features, and treat 'fc7' as a new representation and calculate cosine distance, then pick up ten pairs with the maximum distance between those features and sum those ten distances. If the sum value is smaller than a certain threshold, there is no noise in that celebrity. Otherwise pick out the face image that appears most frequently in the ten pairs as a noise image. VGGface[4] treats top 50 images of one celebrity as positive samples, others are negative, and trained a one-versus-all SVM[14] to detect noise images. All of those methods use manual checking as a final noise filter step.

The rest of this paper is organized as follows. In Section 2, we talk about data collection and some simple de-noising tricks. In Section3, we present details of the proposed de-noising algorithm. In Section 4, we report the experiments, and in Section 5 we get a conclusion and discuss some future work.

2. Data Collection

In this section, we describe an automatic procedure to collect Chinese celebrity name list and images from web. In order to get data with high quality and purity, some simple filter rules are applied for promising the validity of the name list. Finally, we make a Chinese celebrity name list ranked with 'popularity', which contains 11,289 names, and 515,450 images totally, where there are still much noise in image dataset.

2.1. Name List Collection

Unlike CASIA or MS-Celeb-1M, they got celebrity names from structural website like IMDB or Freebase that are well organized. In Japan, 'talent-dictionary.com' also contains structural information about celebrities. As far as we know, there is no such public website or database in China. Thus, we develop a name list collection scheme. At first, we collect information of celebrity names by searching some wiki categories on Wikipedia and Weibo like 'category: Chinese female singers', 'category: Chinese film actress', 'category: Chinese politicians', and then use spider to get these names. About 15 categories containing 6000 names are crawled by this step. Next, we use the names collected in the first step as seeds to put in 'Baidubaike' [5] search box. Baidu's knowledge base 'BaiduZhixin' will give back some items that are relative to the seeds, and these relative items are further used as new seeds for searching more relative items. The search process is repeated 4 times, and we can obtain about 100,000 celebrity names by removing duplicate ones. In Baidubaike webpage, we also obtain additional information like 'Birthday', 'Nationality', etc. As shown in Fig. 1, we utilize several simple rules to filter names:

1. Items must have 'Birthday' column.
2. Items must be born after 1940.
3. Items must have 'Nationality' column and 'Nationality' must be 'China' or 'Chinese'.

Figure 1. Sample pages of Baidubaike.

After applying these rules, we collect 27,232 celebrity names. Considering that some celebrities in this name list are not famous enough, it is difficult to find enough images of them from Internet, thus we filter the name list by 'Popularity' level of each celebrity. In Baidubaike, there is a number in each item's page about 'how many time this page has been reviewed'. We consider this number as the degree of 'Popularity', and rank the name list according to popularity, and only choose celebrities

whose degree of popularity is above a certain threshold. At last, 11,289 names are selected and composed as the final name list.

2.2. Images Collection

Using python 'requests' library, we crawled images from two famous search engines (Baidu, Bing). For each celebrity, 30 and 40 images are crawled respectively from Baidu and Bing, i.e. totally 70 images for each celebrity.

2.3. Detection, Alignment and Simple De-noising

We use MTCNN model for face detection and alignment, which is proposed by Kaipeng Zhang [6]. We firstly detect all face images by MTCNN, and then apply our de-noising algorithm for all detected face images. In experiments, we find that the de-noising algorithm cannot work very well for some low-quality images, and those low-quality images have the following attributes:

1. More than 5 detected faces in one image.

2. Height or width of detected faces are less than 40 pixels.

Therefore, during the detection and alignment stage, the face images detected from the low-quality images with the above conditions will be rejected. After this stage, we obtain a less noisy dataset containing 515450 images.

3. De-noising Algorithm

In this section, we will introduce the de-noising algorithm behind the filter in detail. 3.1 will describe a whole image of our algorithm. 3.2 and 3.3 will introduce two loss which will make our feature extractor network perform better. 3.4 will introduce a cluster algorithm to detect ourliers in the mapped embeddings.

Here we define the feature extraction network as $F(d|\theta)$, where d is the input sample, θ is the parameters of this network.

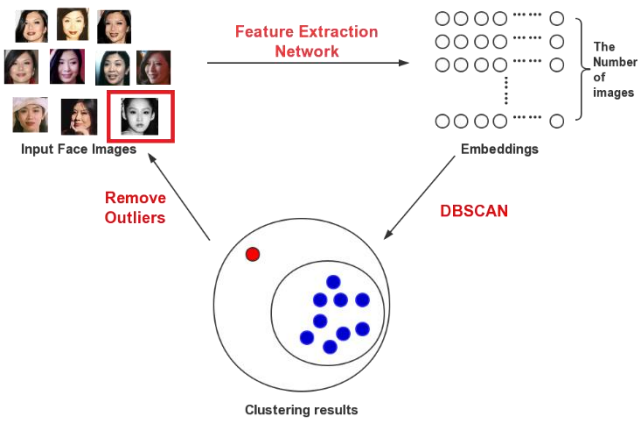

Figure 2. How the de-noising algorithm works

3.1. De-noising Algorithm

To de-noise a dataset, we need to choose an effective approach to map input images into embeddings. By learning a mapping from face images to a compact Euclidean space, we can calculate the similarity of faces through various distances among feature vectors (embeddings).

The de-noising algorithm is shown in Figure 2. The pipeline of our algorithm can be represented as follows:

1. *For each batch:*
2. $\boldsymbol{d^m} = Preprocess(input)$;
3. $\boldsymbol{e^m} = F(\boldsymbol{d^m}|\theta)$;
4. $z^m = DBSCAN(e^m; \varepsilon, MinPts)$;
5. *Remove the outliers*
6. *End*

Here d^m is the input of feature extraction CNN, e^m is the embeddings of inputs after feature extraction, z^m is the outliers detected using DBSCAN.

In our algorithm, we define a criterion recall by

$$Recall = \frac{number\ of\ z^m}{Total\ number\ of\ outliers} \quad (1)$$

If the recall is large, it means our algorithm can filter most outliers, i.e., its performance is good. Otherwise, our algorithm doesn't work well.

3.2. Center Loss

Traditional convolutional neural networks solving face recognition problems usually define soft-max loss (cross-entropy loss) as their loss function. Often defined as below,

$$Softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad (2)$$

But there are some drawbacks: 1. From the view of clustering, intra-class variances for the features in the same cluster are not small enough, and result in that the features of different classes cannot be well separated; 2. Each class occupies too much room in the Hilbert space, causing, with the increasing of classes, the Hilbert space can be disordered. What we expect is that each class occupies only a limited space, so that we can keep the universal property of features. 3. The softmax loss leads to the definite results, i.e., the results of the softmax loss are 1 or 0. It results in that some feature points at bound should be outliers, but the softmax method classifies this point as one class with quite strong determination. Thus, it is hard to apply reliability measurements by the softmax loss for clustering.

In this work, we combine the soft-max loss with center loss [7], instead. The center loss focuses on solving the

clustering problem, because it simultaneously considers intra-class compactness and inter-class separability.

The center loss is defined as

$$CenterLoss = \frac{1}{2N}\sum_{i=1}^{N}|d_i - c|_2^2 \quad (3)$$

where, c is the class center, d_i is the i_{th} sample's feature point.

Usually, the center loss and the soft-max loss are combined by

$$L = L_s + \lambda L_C \quad (4)$$

where, L_s is the softmax loss, L_c is the center loss, λ is a constant parameter.

3.3. Triplet Loss

Recently, learning to rank (L2R) is widely used in face recognition, like Google's FaceNet[8]. The core task of learning to rank is to find a good similarity function, where triplet loss is a kind of popular similarity function.

A triplet is composed of three elements. One element is a sample chosen randomly from a training set, called Anchor d^a. The second element is a sample chosen randomly from the class Anchor belongs to, called Positive d^p. The last element is a random sample from other classes, called Negative d^n.

The purpose of the triplet loss is to keep the distance of Positive and Anchor as small as possible, but the distance of Negative and Anchor as large as possible. In addition, there is a minimal margin α between these two distances. We can present triplet loss by

$$|F(d^a) - F(d^p)|_2^2 + \alpha < |F(d^a) - F(d^n)|_2^2 \quad (5)$$
$$,\forall(F(d^a), F(d^p), F(d^n)) \in S$$

, and the corresponding loss function is represented by

$$\sum_{i=1}^{N}\max\{[|F(d^a) - F(d^p)|_2^2 - |F(d^a) - F(d^n)|_2^2 + \alpha], 0\} \quad (6)$$

If the distance of Anchor and Negative is larger than the distance of Anchor and Positive plus a margin α, there will be a loss. Otherwise, the loss is zero.

3.4. DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise) [13] is a clustering algorithm proposed in 1996. Given a dot set in some domain, this algorithm can cluster neighbor dots into a class, at the same time, noting dots of other class with low density.

This algorithm requires two parameters, ε and $MinPts$. ε is the radius to define the neighborhood domain, and $MinPts$ is defined by the minimal number of points that a high-density domain should contain. It starts with a random unvisited point, then search the ε neighborhood domain of this point. If there are enough points in this neighborhood, a new cluster will be created, otherwise, this point will be considered as an outlier. What we should notice is that if a point belonging to one cluster is clustered into another cluster, these two clusters will be merged into one cluster. Repeating this processing until all points are clustered.

Here we represent DBSCAN as

$$z^m = DBSCAN(e^m;\ \varepsilon, MinPts) \quad (7)$$

where, z^m are the outliers of the m_{th} person, $e^m = F(d^m)$ are the embeddings of the m_{th} person extracted by the feature extractor.

4. Experiments

In this section, firstly, we make a brief introduction about our experiment details. Secondly, the way to choose parameters of DBSCAN is introduced. Thirdly, filter results on our dataset and MS-Celeb-1M are evaluated and compared. Finally, we make a discussion about the noise filter quality including advantages, drawbacks and failed example analysis.

4.1. Training Feature Extraction Network

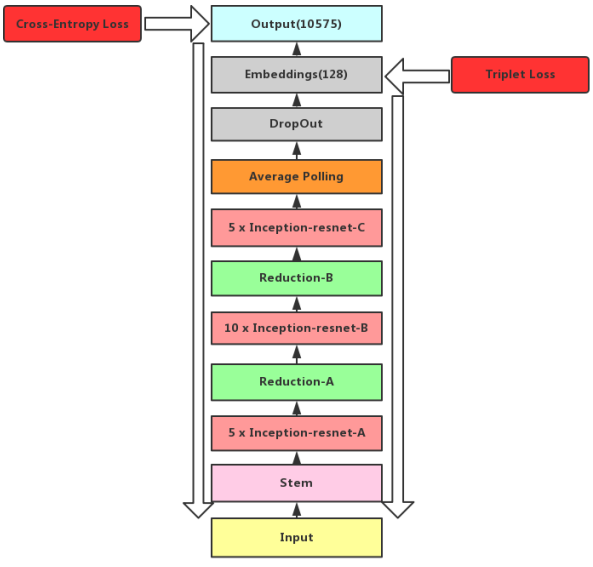

Figure 3. The architecture of the training network. It's mainly Inception-Resnet-V1. We trained with cross-entropy loss first, then fine-tuned with triplet loss.

In this part, we will give an explanation about training a convolutional neural network for feature extraction.

Our code is based on Tensorflow with GPU. CNN model is running on 2 GTX 1080 Ti GPU with totally 24GB memory. CASIA-WebFace is used as a training set, which has 10,575 identities and 494,414 images totally.

Table 1. Parameter experiment result of DBSCAN

ε	0.60	0.65	0.70	0.72	0.74	0.75	**0.76**	0.77	0.80	0.85	0.90	0.95	1.00
Recall	0.402	0.611	0.819	0.893	0.912	0.919	**0.921**	0.909	0.886	0.819	0.765	0.699	0.582

Table 2. The performance of de-noising algorithm

Dataset	Before Filter			After Filter			Precision
	Total	Outliers	Purity	Total	Outliers	Purity	
Our Data (1000)	49780	16976	65.9%	32624	742	97.7%	94.6%
MS-Celeb-1M-fine (390)	32346	8584	73.5%	20182	267	98.7%	68.3%
MS-Celeb-1M-bad (110)	7906	-	-	5077	-	-	-

Moreover, MTCNN is applied to detect and get aligned face with the size of 180x180 from the training set. Furthermore, face images will be randomly cropped to 160x160 and flipped left and right for data augmentation. Learning rate is set as 10^{-2} initially, and decreased by factor 10 after running each 100 epochs until reaching 10^{-4}. Training process and code is highly inspired by Facenet open source [9] and [4], cross entropy loss with 10,575 categories was trained first to accelerate convergence of network. Then the top classification layer is removed, and the rest model is fine-tuned based on the triplet loss. Model structure can be seen in Figure 3.

4.2. Parameters of DBSCAN

DBSCAN has two parameters: ε and *MinPts*. We need to find an ideal ε so that the recall of outliers is high. To get an ideal ε, we manually label images of 300 celebrities for de-noising test. For each celebrity m, the corresponding image d^m is fed into well trained feature extraction network F and the corresponding embeddings e^m can be calculated. Then, we put ε, e^m and *MinPts* into DBSCAN and select the noisy image $z^m$of celebrity m. After filtering all 300 celebrities, we can get the 'Recalls of outliers with its corresponding parameters. Repeating above procedure with different ε, then the relationship between ε and the Recalls of outliers can be found in Table1. According to Table 1, we set $\varepsilon = 0.76$ in the following experiments, which produces the highest Recall.

MinPts is usually set as 4. Given that each celebrity has around 70 images, experimentally 4 is an ideal number for capturing cluster structure.

4.3. Evaluation of our method

We randomly chose 1000 celebrities from our data set, 500 celebrities from MS-Celeb-1M for evaluating the performance of the proposed method. The ground truth is manually created. Detailed statistics is listed in Table 2.

For our data, the purity of the original dataset is 65.9%, but it can reach to 97.7% after the de-noising filter.

It's a great improvement which means the proposed de-noising algorithm works very well. Besides, 'Precision' is also a very important indicator. We define precision by:

$$Precision = \frac{n_{deleted\ noise\ imgs}}{n_{total\ deleted\ imgs}} \quad (8)$$

The higher precision we have, the less valid images we will mis-delete. Precision = 94.6% means when the algorithm deletes 100 images, and 94 of them are real noise images, which is also an exciting result.

For MS-Celeb-1M data set, we divide the result into 2 group: fine group and bad group. For fine group, the de-noising algorithm's performance is even better than that of our data, in which the purity reaches 98.7%. But there still are two problems:

1.Precision (68.3%) is not good, which means when the algorithm deletes 100 images, only 68 of them are real noise images. The reason is that the selected ε in Section 4.2 is not good enough for filtering MS-Celeb-1M data. Thus, we believe it's because there is variance between two datasets, i.e. MS-Celeb-1M cover celebrities from all over the world while our data only covers Chinese celebrities.

2.There are 110 celebrities that the algorithm fails to de-noise ('fail' means even after de-noising, it remains more than 1 cluster and there is no way we can know which one is main cluster). The reason can be found in Fig. 4 and Fig. 5. For relatively clean data like ours, main cluster is easy to locate. But for extremely noisy data like MS-Celeb-1M, high level noise will make embeddings loosely. Connectivity is crucial for DBSCAN, the looser connectivity data has, the harder for DBSCAN to make a clear boundary between different clusters. That's why the proposed our algorithm doesn't always work well for MS-Celeb-1M.

5. Conclusion

In this work, we present a reasonable way to collect celebrity name list and their images from web engine.

Then we train a deep convolutional neural network to map those images into Euclidean space. After that, we use DBSCAN to find the spatial structure to detect noise images. The way to choose parameters of DBSCAN is also illustrated. The performance of the proposed de-noising algorithm on our data and MS-Celeb-1M are compared, and we analyze the reason why algorithm works very well on our data while a little struggle on MS-Celeb-1M. And we believe such a drawback can be avoided by using different parameters in DBSCAN. We will consider such a solution as future work. Consequently, we have created a new face dataset containing 395,130 images of 11,289 Chinese celebrities, which will be applied for improving Asian face recognition in the near future.

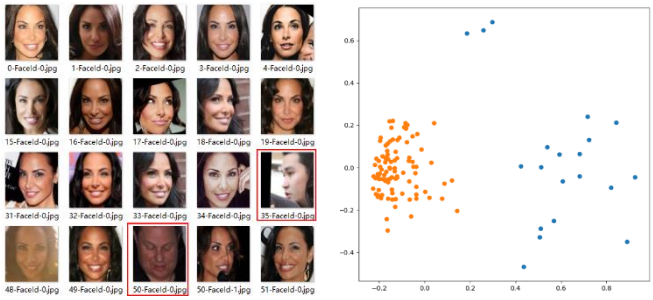

Figure 4. An example of well de-noised celebrity. For image size concerned, we only crop face parts here. In the right scatter image, we can find that embeddings in main cluster stay very close from each other.

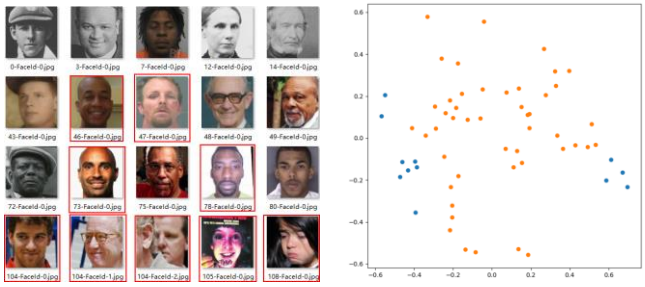

Figure 5. A failed example. Because of too noisy data, all embeddings are connected loosely. In this case, DBSCAN works badly.

6. Acknowledgement

This work is supported by our supervisor, Professor Yoshie Osamu, and DataSection, Inc., Japan. Thanks for their guidance so that this paper can be finished in time.

Reference

[1] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. Technical report, arXiv:1411.7923, 2014.

[2] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In European Conference on Computer Vision, pages 87–102. Springer, 2016.

[3] A. Bansal, A. Nanduri, R. Ranjan, C. D. Castillo, and R. Chellappa. Umdfaces: An annotated face dataset for training deep networks. arXiv:1611.01484, 2016.

[4] Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: Proceedings of the British Machine Vision, vol. 1, no. 3, p. 6, 2015

[5] https://baike.baidu.com/

[6] Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multi-task cascaded convolutional networks. arXiv preprint, 2016.

[7] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao, “A discriminative feature learning approach for deep face recognition,” in European Conference on Computer Vision. Springer, 2016, pp. 499–515.

[8] Florian Schroff, Dmitry Kalenichenko, and James Philbin, “Facenet: A unified embedding for face recognition and clustering,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.

[9] https://github.com/davidsandberg/facenet

[10] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In Proc. NIPS, 2014.

[11] Y. Sun, D. Liang, X. Wang, and X. Tang. DeepID3: Face recognition with very deep neural networks. arXiv preprint arXiv:1502.00873, 2015.

[12] Liu F, Zeng D, Zhao Q, Liu X.: Joint face alignment and 3D face reconstruction. In: ECCV 2016.

[13] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231.

[14] R. Rifkin and A. Klautau, "In defence of one-vs-all classification", Journal of Machine Learning Research, vol. 5, pp. 101-141, 2004.