

Face Clustering: Representation and Pairwise Constraints

Yichun Shi, *Student Member, IEEE*, Charles Otto, *Member, IEEE*, and Anil K. Jain, *Fellow, IEEE*

Abstract—Clustering face images according to their latent identity has two important applications: (i) grouping a collection of face images when no external labels are associated with images, and (ii) indexing for efficient large scale face retrieval. The clustering problem is composed of two key parts: representation and similarity metric for face images, and choice of the partition algorithm. We first propose a representation based on ResNet, which has been shown to perform very well in image classification problems. Given this representation, we design a clustering algorithm, Conditional Pairwise Clustering (ConPaC), which directly estimates the adjacency matrix only based on the similarities between face images. This allows a dynamic selection of number of clusters and retains pairwise similarities between faces. ConPaC formulates the clustering problem as a Conditional Random Field (CRF) model and uses Loopy Belief Propagation to find an approximate solution for maximizing the posterior probability of the adjacency matrix. Experimental results on two benchmark face datasets (LFW and IJB-B) show that ConPaC outperforms well known clustering algorithms such as k -means, spectral clustering and approximate Rank-order. Additionally, our algorithm can naturally incorporate pairwise constraints to work in a semi-supervised way that leads to improved clustering performance. We also propose an k -NN variant of ConPaC, which has a linear time complexity given a k -NN graph, suitable for large datasets.

Index Terms—face clustering, face representation, Conditional Random Fields, pairwise constraints, semi-supervised clustering.

I. INTRODUCTION

CAMERAS are everywhere, embedded in billions of smart phones and hundreds of millions of surveillance systems. Surveillance cameras, in particular, are a popular security mechanism employed by government agencies and businesses alike. This has resulted in the capture of suspects based on their facial images in high profile cases such as the 2013 Boston Marathon Bombing [1]. But, getting to the point of locating suspects' facial images typically requires manual processing of large volumes of images and videos of an event. The need for automatic processing of still images and videos to assist in forensic investigations has motivated prior works on clustering large collections of faces by identity [2]. In [3], Nech et al. also used face clustering to help label face images and compiled MF2 face dataset.

In surveillance applications, the quality of available face images is typically quite low compared to face images in some of the public domain datasets such as the Labeled Faces in the Wild (LFW) [4]. The IARPA Janus project is pushing the

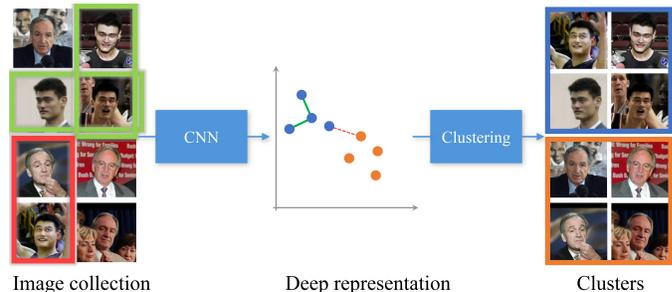


Fig. 1: Face clustering workflow. A deep neural network is trained to generate the representations for aligned face images. Given the representation and a similarity measure, goal of clustering is to group these unlabeled face images according to their identity. In the semi-supervised scenario, pairwise constraints are provided along with the unlabeled data. The red line here indicates a cannot-link pair and green lines indicate must-link pairs.

boundaries of unconstrained face recognition and has released a dataset, NIST IJB-B [5]¹, where many of the faces cannot be detected by off-the-shelf face detectors [7]. The face recognition problem posed by the Janus benchmark may therefore be closer to that encountered in forensic applications. We attempt to handle this more difficult category of faces by: (i) improving the face representation (through the use of large training sets, and new deep network architectures), (ii) developing an effective face clustering algorithm to automatically group faces in images and videos, and (iii) incorporating user feedback during the clustering process via a semi-supervised extension of the proposed clustering algorithm.

To develop a representation for face clustering, we leverage two public domain datasets: CASIA-Webface [8] and VGG-Face [9]. In terms of network architecture, we adopt deep residual networks, which have resulted in better performance than VGG-architecture on the ImageNet benchmark [10], and improved results over the architecture proposed in [8] on the BLUFR protocol [11].

Given a representation, we propose a face clustering method, called *Conditional Pairwise Clustering (ConPaC)* to group the face collection according to their hidden class (subject identity) using the pairwise similarities between face images. No assumption about the data, including the true number of identities (clusters), is used; only a threshold on similarity is specified to balance between the precision and recall rate. Instead of learning new similarity measures or representations and feeding them to a standard partitional or hierarchical clustering method, Conditional Pairwise Clustering directly treats the adjacencies between all pairs of faces as

Y. Shi and A. K. Jain are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48824. E-mail: shiyichu@msu.edu, jain@cse.msu.edu

C. Otto is with Noblis, Reston, VA. E-mail: ottochar@gmail.com

¹There is also an earlier version with smaller number of images, called IJB-A released in 2015 [6]

the variables to predict and look for a solution that maximizes the joint posterior probability of these variables given their corresponding pairwise similarities. To model this conditional distribution, we propose a triplet consistency constraint which reveals such a dependency between the output variables that a valid adjacency matrix must be transitive to represent a partitioned clustering. That means any two adjacent points should share exactly the same adjacent neighbors. The proposed model can dynamically determine the number of clusters, and also retain the similarity information. In particular, we model the problem as a Conditional Random Field (CRF) and employ Loopy Belief Propagation to arrive at a valid adjacency matrix. This model is easily extended to the semi-supervised clustering by accepting a set of pairwise constraints (either must-link, or cannot-link assignments) on the similarity matrix.

We perceive the following contributions in this work: (i) We propose a clustering algorithm (ConPaC) based on direct estimation of an adjacency matrix derived from pairwise similarities between faces using the learned representation from a Deep Residual Network; (ii) We evaluate the proposed method on two unconstrained face datasets: LFW and IJB-B; (iii) We show that the proposed method can be naturally applied to semi-supervised face clustering scenario; (v) We propose an approximate k -NN variant of the algorithm for efficient clustering of millions of face images.

II. BACKGROUND

A. Face Representation

Face images have been traditionally represented by appearance models or local descriptors [12] [13] [14] [15]. But as Deep Neural Networks (DNN) have shown their great potential in solving computer vision problems due to its representation learning ability [16] [17] [18], a number of DNN based methods have been proposed for face representation and recognition. The DeepFace [19] method trained a CNN on a dataset of four million facial images belonging to more than 4,000 identities. The training is based on minimizing classification error and the output of the last hidden layer taken as the face representation. DeepFace significantly surpassed the traditional methods in face recognition, especially for unconstrained face images. Sun et al. extended the work of DeepFace in their DeepId series [20] [21] [22] [23]. They proposed to use multiple CNNs with joint Bayesian framework [24] and added supervision to early convolutional layers. Schroff et al. [25], in their FaceNet work, abandoned the classification layer and instead introduced the triplet loss to directly learn an embedding space where feature vectors of different identities could be separated with Euclidean distance.

B. Face Clustering

Cluster analysis is an important topic widely studied in pattern recognition, statistics and machine learning [26]. It is useful for exploratory analysis by a preliminary grouping of a collection of unlabeled data. Due to potentially large and unknown number of identities in many large scale face collections, it is useful to tag the face images with the labels obtained from clustering. Otto et al. [2] provided a

brief review of face clustering. Most of the previous studies [27] [28] [29] [30] [31] [32] focused on learning a good similarity matrix or robust representations from non-discriminative low-level features and then partitioned the dataset with standard clustering algorithms such as spectral clustering. However, in practice, high-level features generated by deep neural networks today can give quite robust representation, and hence similarity, even with simple metric functions. Furthermore, many supervised metric or representation learning methods have been proposed, which are shown to be able to enhance the deep representations and have good generalizability [24] [33]. Therefore, as we will show in Section IV-B, similarity learning is relatively simple in practical face clustering problems; instead the partitioning algorithm plays a more important role.

Otto et al. [34] [2], based on the work of [31], made use of the assumption that homogeneous face images (images that belong to the same identity) usually have similar nearest neighbors and proposed the approximate Rank-order distance metric. They showed that by linking all the image pairs within a certain distance, they can achieve good clustering performance on challenging unconstrained face datasets.

C. Semi-supervised Clustering

Given the difficult nature of data clustering (choice of representation, similarity measure and number of clusters), one approach to improve clustering performance is to incorporate side-information. One common form of side-information is pairwise constraints, indicating that a pair of data points either must be placed in the same cluster (a “must-link” constraint), or they cannot be placed in the same cluster (a “cannot-link” constraint), as shown in Figure 1. Wagstaff et al. [35] first incorporated the pairwise constraints into k -means algorithm by forcing the cluster assignments to satisfy the constraints and showed that user-specified constraints could help to improve clustering results. Xing et al. [36] proposed to learn a Mahalanobis distance metric from the given constraints before applying k -means. Basu et al. [37] designed a probabilistic model for semi-supervised clustering with Hidden Markov Random Fields (HMRFs) and used EM algorithm to optimize the parameters. Research has also been conducted on incorporating pairwise constraints into hierarchical clustering [38] and spectral clustering [39] [40]. For a review of semi-supervised clustering, readers are referred to [41].

D. Conditional Random Fields (CRFs)

Conditional Random Fields (CRFs) are a type of undirected probabilistic graphical models first proposed by Lafferty et al. for predicting labels of sequential data [43] and later introduced to computer vision to model images [44] [45] [46]. The difference between CRFs and traditional Hidden Markov Models (HMMs) [47] lies in that CRF is a discriminant model which directly models the conditional distribution $p(Y|X)$ rather than joint distribution $p(Y, X)$ and predicts the labels



Fig. 2: Example face images from (a) LFW, (b) IJB-B, (c) CASIA-webface, and (d) VGG datasets.

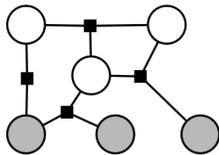


Fig. 3: An example factor graph of a general CRF model. Here, the filled nodes are input nodes and the white nodes are output nodes. Each factor (square) represents a potential function on a clique, encoding the constraints between nodes. There are constraints either between input and output or between output nodes. The figure is taken from [42].

Y by maximizing the posterior probability. Generally, a CRF can be formulated as:

$$p(Y|X) = \frac{1}{Z} \prod_{C_p \in C} \prod_{\psi_c \in C_p} \psi_c(X_c, Y_c; \theta_p), \quad (1)$$

where Z is the normalization factor, $C = \{C_1, C_2, \dots, C_P\}$ is the set of all cliques in the graph, ψ_c is a potential function defined on the variables (X_c, Y_c) in clique C_p , and θ_p is a set of parameters of the model [42]. We can represent undirected graphical models by a factor graph, where a factor node is there for each potential function and connects to every node in its clique, as shown in Figure 3. Usually, there are two types of potential functions in CRFs: (1) association potential that equals the local conditional distribution over observations $p(Y_c|X_c)$ and (2) interaction potential that encodes the dependencies between different output variables. Although both of them are originally defined as Gibbs distributions on features in [43], association potentials are often substituted by supervised discriminant classifiers such as neural networks [44] [48].

As for inference on the CRFs, any method for undirected graphical models can be applied, and one of these methods is Belief Propagation (BP) [49]. There are two types of BP algorithms: sum-product and max-sum. They are exact inference methods, respectively, for finding marginal probability and maximizing posterior probability on tree-like graphical models. But because they only involve local message updates, they can also be applied to graphs with loops, resulting in Loopy Belief Propagation. Although Loopy Belief Propagation is not guaranteed to converge, it has achieved success in a

variety of domains [50] [51] [52]. It has also been shown that the result of Loopy Belief Propagation corresponds to the stationary point of Bethe free energy and that it is related to variational methods [53]. Readers are referred to [54] for further information on CRFs and Loopy Belief Propagation.

III. FACE DATASETS

We leverage the CASIA-Webface [8], and VGG-Face [9] datasets to train networks for learning the representation to be used for clustering. We then evaluate the performance of our clustering algorithm on two benchmarks datasets, LFW [4], and IARPA Janus Benchmark-B (IJB-B). Some example images from these datasets are shown in Figure 2. As stated in [8] and [9], there is no overlap of identity between LFW and VGG-Face or LFW and CASIA-Webface. Similarly, IJB-B does not include overlapping identities with VGG-Face or CASIA-Webface [5].

A. LFW

The Labeled Faces in Wild (LFW) [4] contains 13, 233 face images of 5, 749 individuals; of those 5, 749 individuals, 4, 069 have only one face image each. The dataset was constructed by searching for images of celebrities and public figures, and retaining only those images for which an automatically detectable face via the off-the-shelf face detectors [7] was present. As a result, facial pose variations in LFW are limited.

B. IJB-B

The IJB-B dataset [5] is composed of 7 different clustering experiments, with increasing number of subjects. These experiments, respectively, involve 32, 64, 128, 256, 512, 1, 024, 1, 845 subjects with total of 1, 026, 2, 080, 5, 224, 9, 867, 18, 251, 36, 575 and 68, 195 images, respectively. Two protocols related to clustering are defined for IJB-B dataset: (i) clustering of detected faces and (ii) face detection + clustering. Since the focus of this work is face clustering, we will use the first protocol and assume faces have already been detected. The faces are aligned following the procedure in [55], using the bounding boxes provided in IJB-B as the starting point for landmark detection. Many images in the IJB-B datasets are in extreme poses or of low quality, making the clustering task



Fig. 4: An example showing the normalization of face images. We normalize all our images before feeding into the network according to the procedure in [55], where the figure is taken from.

more difficult for IJB-B than for LFW. Most of the images in the clustering protocols of IJB-B are from video frames, making it more related to the surveillance application.

C. CASIA-Webface

The CASIA-webface dataset [8] is a semi-automatically collected face dataset for pushing the development of face recognition systems. It contains 494,414 images of 10,575 subjects (mostly celebrities) downloaded from internet. However, we are unable to localize faces in some of the images with the face detector from the Dlib library². So we use a subset of CASIA-Webface with 404,992 face images of 10,533 subjects to train our network. This dataset has been popular for training deep networks.

D. VGG-Face

The VGG-Face dataset was released in 2016 as a set of 2.6 million URLs with corresponding face detection locations [9]. We could acquire only 2.2 million images of the original 2.6 million listed URLs due to broken links. Example VGG images are shown in Figure 2(d). Additionally, among the images we were able to download, we identified a number of exact duplicate files. On manual examination, these duplicates typically had mislabeled identities, or were placeholder images served by the image host when the original image was no longer available. After excluding duplicate images, we retained 1.7 million images from the VGG dataset. We combined these images with the CASIA-Webface dataset (and merged overlapping subject identities), to get a total of 2.1M images, of 11,326 distinct subjects.

IV. METHODS

A. Representation

He et al. [10] used a “deep residual network” architecture to achieve competitive results on the ImageNet object recognition dataset. So we directly adapt the architecture for face recognition (leveraging the Torch7 framework [56] and the implementation of residual networks released by Facebook AI Research fb.resnet.torch³). We have investigated the 50, and 101-layer architectures outlined in [10]. In terms of data augmentation, we scale our normalized face images following the alignment procedure proposed in [55] to 256×256 , as

shown in Figure 4, and randomly crop 224×224 regions during training. We additionally flip images during training, and use the scale and aspect-ratio augmentations from [57]. As for feature extraction, each image is aligned using the same procedure, then a 2048-dimension feature vector is extracted from the bottleneck layer⁴ with 10-crop strategy⁵ and is normalized into unit ℓ^2 norm. Because of the low quality of the face images in IJB-B benchmark, many images cannot be aligned using Dlib. In such cases, we crop a square region containing the ground-truth bounding box provided in the protocol. This is allowed in the test 7 of the benchmark, where the detection task is assumed to be already finished.

B. Motivation

Face clustering, like other clustering problems, attempts to partition data points into a number of groups based on their similarities or structure. However, in real-world problems of unconstrained face clustering, the situation could be quite different such that many popular clustering algorithms are not suitable. On one hand, most clustering algorithms are based on certain distribution assumptions of the data. For example, k -means assumes that the data points of different classes are close to the centroids of the classes and spectral clustering [58] [59] aims at finding a balanced partition of the dataset. But in fact, the data could be distributed in arbitrary shapes, depending on the representation, and the sizes of different clusters in a large face image collection could be very unbalanced. Besides, most algorithms require the number of clusters as an input parameter, which is usually unknown and quite large in real-world face clustering problems. On the other hand, the rapid development of deep neural networks makes it possible to learn highly robust representations for unconstrained face images. Even with simple metric functions, good pairwise similarities can be acquired, providing reliable evidence on pairwise homogeneity (whether a pair of face images belong to the same identity).

In Figure 5, by visualizing the similarity matrix from ResNet representations, we can see it is highly consistent with the ground-truth adjacency matrix. However, in comparison, the resulting adjacency matrices (by using the ground-truth number of classes) of k -means and spectral clustering are not only far from the ground-truth one, but also not similar to the input similarity matrix itself. Thus, we attempt to partition the face images merely based on the pairwise similarities; no other assumptions, including the number of identities is used.

C. Problem Formulation

Given a dataset X of size N , where each $X_i, i = 1, 2, \dots, N$ is a data point, we want to directly estimate an $N \times N$ adjacency matrix Y , where Y_{ij} is a binary variable indicating whether X_i and X_j are assigned to the same cluster. Assuming that we are given the pairwise conditional probability $p(Y_{ij}|X_i, X_j)$ for all pairs, the goal is to find the overall

⁴The last hidden layer. Because a ReLU layer is used as the activation function for the bottleneck layer, all the feature values are non-negative.

⁵Average the features extracted from 10 different sub-crops of size 224×224 (corners + center with/without horizontal flips).

²<http://dlib.net>

³<https://github.com/facebook/fb.resnet.torch>

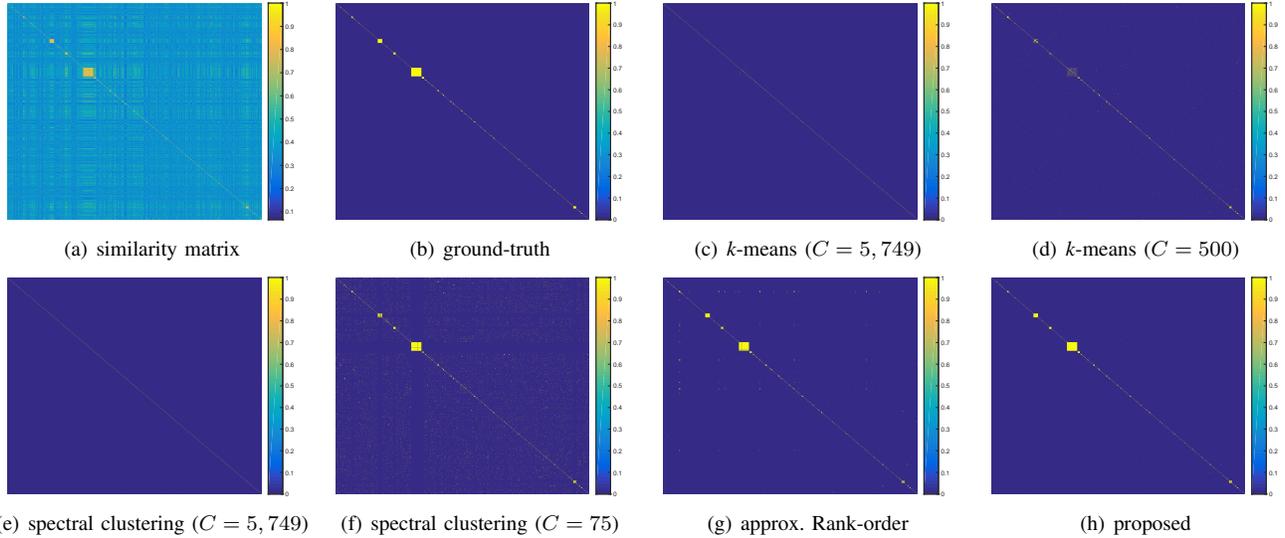


Fig. 5: Visualization of the similarity matrix and adjacency matrices. Using the face representations from deep neural networks, similarity matrix (using cosine similarity) is highly consistent with the ground-truth adjacency matrix. However, using the ground-truth number of clusters ($C = 5, 749$) both k -means and spectral clustering fail to retain the similarities and break the big clusters into small groups. Tuning the number C to smaller values give better performance, but makes the clusters less pure. Thus, to make full use of the similarity matrix, we attempt to find a partition by maximizing the consistency between the adjacency matrix and the given pairwise similarities.

adjacency matrix Y by maximizing the posterior probability $p(Y|X)$. To model this conditional distribution, we need to consider the dependencies between different variables Y_{ij} , for which we propose a triplet interaction constraint to constrain the adjacency matrix Y to be *valid*. By *valid*, we mean that the corresponding graph of that adjacency matrix is transitive and represents a valid partition. This leads to a structured prediction problem, and we use a Conditional Random Field (CRF) model to formulate it and maximize the posterior probability:

$$p(Y|X) = \frac{1}{Z} \prod_{i < j} \psi_u(Y_{ij}) \prod_{i < j < k} \psi_t(Y_{ij}, Y_{ik}, Y_{jk}), \quad (2)$$

where Z is the normalizing factor, the unary association potential $\psi_u(Y_{ij}) = p(Y_{ij}|X_i, X_j)$ is the pairwise conditional distribution over observations and $\psi_t(Y_{ij}, Y_{ik}, Y_{jk})$ is the triplet interaction potential to constrain Y to be valid. Because the adjacency matrix is symmetric, we only need to take Y_{ij} with $i < j$ as variables, so there are in all $\frac{1}{2}N(N-1)$ output nodes. The unary potential is the likelihood of a pair of data points belonging to the same class, which is exactly what the pairwise similarity stands for, so we apply a transformation to the cosine similarity between the deep representations of two faces to attain the genuine unary potential $\psi_u(Y_{ij} = 1)$. In practice, we find that this works well, even without attempting to explicitly model the probability distribution for unary potential.

For a partitioning clustering, if a point i is connected to any point j in a cluster, it should also connect to all the other points in that cluster but not to any point outside the cluster. However, not every adjacency matrix satisfies this requirement. In order to check the validity of an adjacency matrix, we use a measure based on triplet consistency. Consider a triplet of any three points, as shown in Figure 6. Then there are four possible cases regarding the states of three pairs in one triplet.

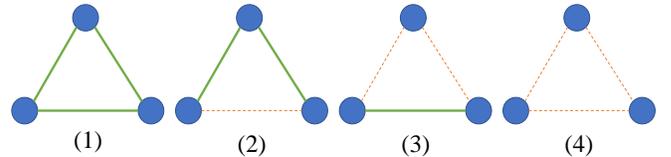


Fig. 6: Four cases of pairwise adjacency in a triplet. A green line means the two points are connected, i.e. assigned to the same cluster. And a red dash line means the two points are not connected. A valid partition is obtained if and only if none of the triplets are in case (2).

An adjacency matrix is valid if and only if none of the triplets is in case (2). Hence we can model the dependencies between different Y_{ij} with triplet cliques. In our undirected graph, every triplet (Y_{ij}, Y_{ik}, Y_{jk}) is fully connected, and forms a clique. The interaction potential for a triplet clique is defined as:

$$\psi_t(Y_{ij}, Y_{ik}, Y_{jk}) = \exp(-\alpha V(Y_{ij}, Y_{ik}, Y_{jk})), \quad (3)$$

where the energy function V is an indicator function which is 1 iff the triplet is inconsistent and 0, otherwise:

$$V(Y_{ij}, Y_{ik}, Y_{jk}) = (1 - Y_{ij})Y_{ik}Y_{jk} + Y_{ij}(1 - Y_{ik})Y_{jk} + Y_{ij}Y_{ik}(1 - Y_{jk}) \quad (4)$$

To seek a valid partition, we consider α in Equation (3) to be sufficiently large such that it dominates the formula. However, it is worth noting here that we do not need to explicitly define α in our algorithm, as shown in the next subsection.

Due to numerical issues, usually we take the negative logarithm on both sides of Equation (2) and minimize its corresponding energy function:

$$E(Y, X) = \sum_{i < j} D(Y_{ij}) + \sum_{i < j < k} \alpha V(Y_{ij}, Y_{ik}, Y_{jk}), \quad (5)$$

where $D(Y_{ij}) = -\log \psi_u(Y_{ij})$ is the unary potential energy.

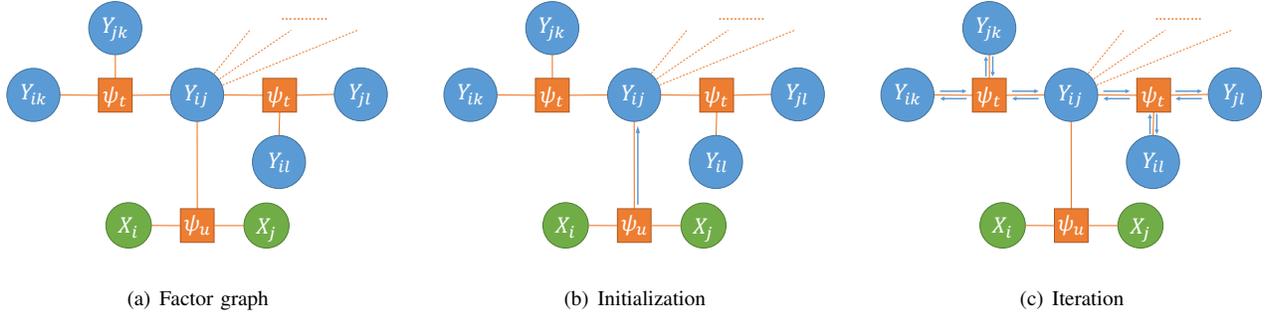


Fig. 7: A graphical illustration of the proposed Conditional Random Field using the neighborhood of output node (pair) Y_{ij} as an example. The figure shows how the nodes are connected, how the factors are related to potentials and how the messages are passed in the graph. Each green node is an input node corresponding to one data point, each blue node is an output node corresponding to an element in the adjacency matrix, and each rectangle is a factor node representing a potential function, which encodes the constraint between variables. The dash lines represent the omitted links in this figure. There are two kinds of constraints: (i) unary potential which pushes the output to conform with the pairwise similarities and (ii) interaction potential which forces output nodes to be consistent so that Y is valid. During the optimization, messages are propagated among output nodes to directly approach a valid adjacency matrix Y which is mostly consistent with the similarity information.

The graph structure of the model is illustrated in Figure 7. Each output node Y_{ij} is in $N - 1$ cliques: one association clique with input pair X_i and X_j , and $N - 2$ interaction cliques consisting of Y_{ij} , Y_{ik} and Y_{jk} .

D. Inference By Belief Propagation

With the factor graph and potentials defined, we can derive a message formula based on the min-sum algorithm, which is the equivalent for max-product algorithm when working with energy functions [60]. We define a message $a_{ij}(Y_{ij})$ as a function of variable Y_{ij} , representing the accumulated energy so far for each state of the variable Y_{ij} . The main procedure of our algorithm is as follows:

1. Initialize all messages as:

$$a_{ij}^0(Y_{ij}) = D(Y_{ij}), \quad (6)$$

which is the message sent from the unary potential factor.

2. At iteration $t = 1, 2, \dots, T$, update the messages as:

$$a_{ij}^t(Y_{ij}) = \sum_{k \in N^{t-1}(i,j)} \min_{Y_{ik}, Y_{jk}} (a_{ik}^{t-1}(Y_{ik}) + a_{jk}^{t-1}(Y_{jk}) + \alpha V(Y_{ij}, Y_{ik}, Y_{jk})), \quad (7)$$

where we are summing up the messages from different factors given a state of Y_{ij} . Within the sum, we are minimizing over different states of Y_{ik} and Y_{jk} .

3. The final state of the variable is determined by:

$$\hat{Y}_{ij} = \arg \min_{Y_{ij}} a_{ij}^T(Y_{ij}), \quad (8)$$

Here, $N^{t-1}(i, j)$ means the set of the points that are adjacent to either i or j at $(t - 1)$ iteration, where by adjacent we mean that it has a lower positive energy than the negative one in that iteration, i.e. $a_{ij}^{t-1}(Y_{ij} = 1) > a_{ij}^{t-1}(Y_{ij} = 0)$. As mentioned in Section II-D, belief propagation is an approximation method for maximizing the posterior probability. It is not guaranteed to find a global optimum. Thus, if there are still

inconsistent triplets after the third step, a transitive merge⁶ is applied to ensure the clustering result Y is valid. There are several issues worth discussing on this procedure:

First, this is not a standard Loopy Belief Propagation algorithm for CRF in two ways: (1) the unary messages are sent only once and (2) the messages are isotropic, i.e. a message sent from a node Y_{ij} is the sum of all messages it receives. We found that these modifications make the algorithm easier to implement, use less memory, converge faster while have little impact on the quality of the results.

Second, the messages could be normalized by subtracting the same value from both states. Theoretically, it makes no difference to the result, but could avoid numerical underflow and provide stability.

Third, the received messages in Equation (7) only include those neighbors k that are adjacent to at least one of i and j in the last iteration, because the messages from other neighbors would have the same value in both states. Thus it makes no difference if we ignore those $k \notin N^{t-1}(i, j)$. For the same reason, we only need to update Y_{ij} whose $N^{t-1}(i, j)$ is not empty.

Fourth, as we assume that α is a very large number, all of the cases where $V(Y_{ij}, Y_{ik}, Y_{jk}) = 1$ could be ignored when taking the minimum in equation (7). For example, for $Y_{ij} = 1$, we do not need to consider the cases where $Y_{ik} = 0, Y_{jk} = 1$ or $Y_{ik} = 1$ and $Y_{jk} = 0$. In all the other cases, because $V(Y_{ij}, Y_{ik}, Y_{jk}) = 0$, α disappears from the formula.

Given the above optimization, along with our use of adjacency lists and an update list, the complexity of the algorithm is $O(TNM^2)$, where N is the number of data points, T is the number of iterations, and M is the maximum degree⁷ of any data point in any iteration. But it should be noted that because M is not a fixed number here, in the worst case the complexity could still become $O(TN^3)$. Besides, we only choose to update the pairs that are in at least one inconsistent

⁶Linking all the positive pairs to build clusters. This can be done with linear complexity by building a disjoint-set data structure.

⁷Number of adjacent points.

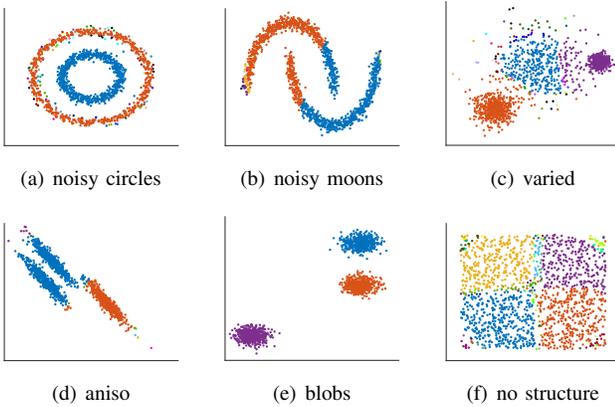


Fig. 8: The result of proposed clustering on toy examples from scikit-learn using a Radial Basis Function (RBF) kernel. We tune the parameter γ according to the datasets. The colors indicate the assigned label for each node.

triplet, so the update list is typically much shorter after several iterations. On the LFW dataset, only 0.59% of the total pairs⁸ are updated, and no more than 1,000 pairs are in the update list after the fourth iteration.

We test the proposed clustering algorithm on a set of toy examples from the scikit-learn library⁹. The results are shown in Figure 8. Because we use the Radial Basis Function (RBF) kernel for the similarity metric on the toy examples, the data points are mainly grouped according to the Euclidean distances between them. Although the number of clusters is not specified in our algorithm, the main groups can be recovered in the belief propagation procedure. The outliers are often assigned as small, separate clusters.

Figure 9 shows how the number of inconsistent triplets decreases with iterations during clustering of the 13,233 images in the LFW dataset. The model converges rather quickly to a stage where only a small number of inconsistent triplets remain. Because the number of remaining inconsistent triplets is usually very small, we do not explicitly force convergence to 0 but apply a transitive merge on the current adjacency matrix to attain the final valid clustering. On LFW, only 6 pairs change their states after we apply transitive merge.

E. Semi-supervised clustering

In semi-supervised or constrained clustering, we utilize the given side information, usually in the form of “must-link” pairs and “cannot-link” pairs. These pairs can either be specified by users or automatically generated with another algorithm to improve clustering performance. The must-links specify pairs of face images that belong to the same identity, while the cannot-links specify pairs that belong to different identities. One way to make use of these pairs is to propagate the constraints. Because our framework is optimized by propagating messages, it becomes quite straight forward to incorporate these constraints: we change the unary potentials of the constrained pairs based on the side information provided. If

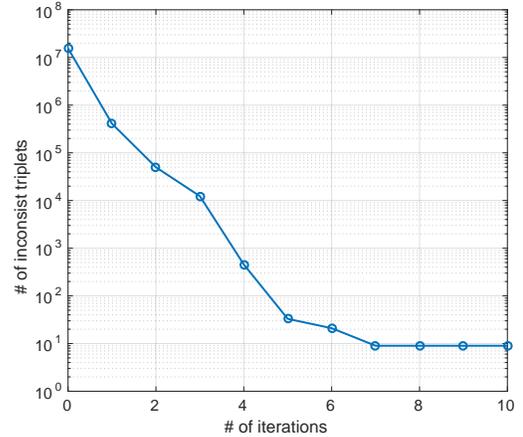


Fig. 9: Number of inconsistent triplets on LFW at each iteration. It decreases rapidly until convergence.

we are very confident with the given constraints (as is the case in our experiments which use ground-truth labels for side information), we can set the positive unary potentials as 1 for must-link and 0 for cannot-link constraints, resulting in very large unary energies. Equation (7) states that very high energy would be avoided when passing messages so the model can still be optimized under these constraints.

F. Efficient Variant of the clustering algorithm using k -NN Graph

The complexity of the proposed clustering algorithm depends on the degrees of data points during the optimization. However, since this number is not fixed, in the worst case its complexity could still be close to $O(TN^3)$, where T and N are the number of iterations and data points, respectively. Therefore, we propose a variant of the algorithm which has a fixed linear complexity. The idea is similar to the one in [2], which takes advantage of approximate k -Nearest Neighbors (k -NN) methods. Instead of estimating the adjacency of every pair within the dataset, we optimize the joint posterior probability of all Y_{ij} where (i, j) is an edge in the k -NN graph. These Y_{ij} compose a subset of elements in the full adjacency matrix Y . The same procedure outlined in section IV-D can still be used for optimization with a few modifications: (1) The neighbor list $N(i)$ now is a fixed list given by the approximate k -NN method, (2) we only update Y_{ij} where $i \in N(j)$ or $j \in N(i)$, and (3) we only need to compute the unary potentials in the initialization step for pairs which will be used in the next iteration, i.e. they are neighbors or they have at least one shared neighbor. While time complexity of this variant, given a pre-computed k -NN graph, is also $O(TNM^2)$ as in section IV-D, M is now a fixed number. In particular, we use the same approximate k -NN method, k -d tree, with the same configuration as in [2] to build the k -NN graph. The complexity of building k -NN graph is $O(N \log N)$ with a fixed search size. If we increase the search size linearly with the size of the dataset, as in [2], it leads to $O(N^2)$ complexity. We use a fixed search size of 2,000 in our experiments.

⁸There are in all 87,549,528 pairs in the LFW dataset.

⁹http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

TABLE I: BLUFR verification performance on LFW. ResNet was trained on the combined VGG+CASIA-Webface dataset. Verification Rates (VR) at a False Alarm Rate (FAR) are reported as (mean - standard deviation) across 10 folds.

Network	VR@FAR=0.1%
50-Layer Pre-activated ResNet	91.04%
50-Layer Pre-activated ResNet, 10-crop	92.22%
101-Layer Pre-activated ResNet	91.18%
101-Layer Pre-activated ResNet, 10-crop	92.10%

V. EXPERIMENTAL RESULTS

A. Face Representation Performance

We evaluate the performance of our representation using ResNet on the BLUFR protocol [11]. Because the performance of state-of-the-art face representations on standar LFW verification protocol has saturate, Liao et al. [11] made use of the entire LFW dataset to design the BLUFR protocol. In this protocol, a 10-fold cross-validation test is defined for both *face verification* and *open-set face identification*. For *face verification*, a Verification Rate (VR) is reported for each split with a false alarm rate: FAR= 0.1%. For *open-set identification*, Detection and Identification Rate (DIR) at Rank-1 corresponding to FAR= 1% is computed.

Table I gives a summary of our face representation performance on the BLUFR protocol’s verification experiment [11]. We trained 50 and 101-layer fully pre-activated ResNets on the combined VGG and CASIA-Webface datasets (using our cleaned version of VGG). A subset o 1,000 images are randomly selected from CASIA-Webface are kept as the validation set. The 50-layer network achieves a 91.04% verification rate at 0.1% FAR after training for 37 epochs, at which point the classification accuracy on the validation set stabilizes. Increasing the network depth to 101 layers does not result in a performance improvement. Using the 10-crop strategy leads to a minor performance improvement (approximately 1% VR at 0.1% FAR), at the cost of substantially increased feature extraction time.

Our best results on BLUFR in Table I are: 92.22% VR at 0.1% FAR for verification, and 62.05% DIR at 1% FAR for open-set identification. This is comparable to some newly reported results on the protocol. For example, Cheng et al. [61] used a GoogLeNet-style Inception architecture combined with traditional Joint-Bayes and attained a 92.19% VR at 0.1% FAR. They further improved this result to 93.05% using their method for estimating the Joint-Bayes parameters. Lv et al. [62] proposed a data augmentation method (perturbing detected facial landmark locations, prior to image alignment), again using an Inception architecture, and attained a 63.73% DIR at 1% FAR in open-set identification, using the fusion of 3 models (the best single-model performance is 57.90%). These results indicate that our results could potentially be further improved, through the incorporation of metric learning methods, or fusing multiple models.

B. Face Clustering

We use the 50-layer ResNet architecture, with 10-crop (Table I) strategy as the representation for our clustering experiments. We evaluate our clustering algorithm on two

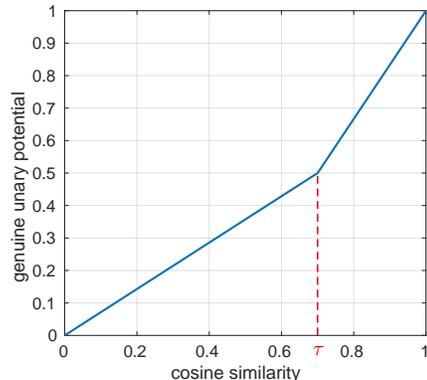


Fig. 10: Transformation function used to map the cosine similarity to the genuine unary probability $\psi_u(Y_{ij} = 1)$. A threshold τ is used to split the function into two pieces. The cosine similarity is non-negative because the the features are extracted from the ReLU layer of the last hidden layer in the ResNet.

unconstrained face datasets (LFW and IJB-B). Before applying the message passing procedure, we obtain unary potentials described in IV-C using a transformation function shown in Figure 10. Threshold τ is the only parameter throughout our experiments. Different τ values control the balance between the recall rate and the precision rate of the resulting partition, which are defined in V-B1. But it is worth emphasizing that the transformation function itself (Figure 10) is not a necessary part of the clustering algorithm and one may choose other ways to initialize the unary potential. We use this transformation function because it is easy to compute and work well empirically. The same threshold $\tau = 0.7$ is used as default unless otherwise stated.

We call our algorithm Conditional Pairwise Clustering (ConPaC), which is implemented in C++ and evaluated on an Intel Xeon CPU clocked at 2.90GHz using 24 cores. The ConPac algorithm is compared to the following benchmarks: (1) k -means, (2) Spectral Clustering [59], (3) Sparse Subspace Clustering (SSC) [30] (4) Affinity Propagation [63], (5) Agglomerative Clustering, (6) Rank-order clustering [31], and (7) Approx. Rank-order clustering [2]. We use the MATLAB R2016a implementation of the k -means algorithm, and a third-party MATLAB implementation of Spectral Clustering¹⁰. We use the author’s implementation of SSC¹¹. For Affinity Propagation and Agglomerative Clustering, we use the scikit-learn implementation [64]. Since all the implementations tested use built-in functions for the core steps, such as matrix decomposition, the difference in the programming language shouldn’t affect the run-time excessively, and we consider it fair to compare the execution times of these implementations.

We use the same representation for all the algorithms except Approx. Rank-order, for which we follow [2] because it generates better clustering results. We use Euclidean distance for k -means, RBF kernel for Affnity Propagation, and cosine similarity for Spectral Clustering.

¹⁰<http://www.mathworks.com/matlabcentral/fileexchange/34412-fast-and-efficient-spectral-clustering/content/files/SpectralClustering.m>

¹¹<http://www.vision.jhu.edu/code/>



Fig. 11: Example clusters by the proposed clustering algorithm on the LFW dataset. The first two rows show example impure clusters while the other two rows show pure clusters. For impure clusters, different colors of bounding boxes indicate the images are from different identities, and images without bounding boxes are from the same identity.

1) *Evaluation Measures*: Two measures are used to evaluate the clustering results, *Pairwise F-measure* and *BCubed F-measure*. Both compute a *F-score*, which is the harmonic mean of *Precision* and *Recall*. The difference between them lies in the metrics used for precision and recall.

In *Pairwise F-measure*, *Precision* is defined as the fraction of pairs that are correctly clustered together over the total number of pairs that belong to the same class. *Recall* is defined as the fraction of pairs that are correctly clustered together over the total number of pairs that are in the same cluster. In other words, we are using the labels for all the $\frac{1}{2}N(N-1)$ pairs in the dataset. Thus, we can define the True Positive Pairs (*TP*), False Positive Pairs (*FP*) and False Negative Pairs (*FN*). Then *Precision* and *Recall* can be calculated as:

$$\text{Pairwise Precision} = \frac{TP}{TP + FP} \quad (9)$$

$$\text{Pairwise Recall} = \frac{TP}{TP + FN} \quad (10)$$

BCubed F-measure [65] defines *Precision* as point precision, namely how many points in the same cluster belong to its class. Similarly, point recall represents how many points from its class appear in its cluster. Formally, we use $L(i)$ and $C(i)$ to, respectively, denote the class and cluster of a point i , and define the *Correctness* between two points i and j as:

$$\text{Correctness}(i, j) = \begin{cases} 1, & \text{if } L(i) = L(j) \text{ and } C(i) = C(j) \\ 0, & \text{if otherwise} \end{cases} \quad (11)$$

The *Precision* and *Recall* are defined as:

$$\text{BCubed Precision} = \frac{1}{N} \sum_{i=1}^N \sum_{j \in C(i)} \frac{\text{Correctness}(i, j)}{|C(i)|} \quad (12)$$

$$\text{BCubed Recall} = \frac{1}{N} \sum_{i=1}^N \sum_{j \in L(i)} \frac{\text{Correctness}(i, j)}{|L(i)|} \quad (13)$$

where $|C(i)|$ and $|L(i)|$ denote the sizes of the sets $C(i)$ and $L(i)$, respectively.

TABLE II: Comparison of the F-measures of the proposed algorithm and other clustering algorithms on LFW dataset. The number of identities (true number of clusters) in LFW is 5,749. Run-time is reported in the format of hh:mm:ss.

Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time
TPE [33]*	0.955	—	5,351	—
<i>k</i> -means	0.098	0.680	5,749	00 : 04 : 26
<i>k</i> -means	0.346	0.468	500	00 : 00 : 14
Spectral	0.033	0.559	5,749	06 : 52 : 22
Spectral	0.257	0.249	75	01 : 00 : 56
SSC	0.186	0.430	500	00 : 31 : 52
Affinity Propagation	0.320	0.577	1,203	00 : 06 : 56
Agglomerative	0.962	0.892	4,500	00 : 03 : 04
Rank-Order	0.813	0.891	5,699	00 : 00 : 33
Approx. Rank-Order	0.861	0.875	6,801	00 : 00 : 12
ConPaC (proposed)	0.965	0.922	6,352	00 : 00 : 39

* This result is reported in [33].

The F-measure, or F-score for both criteria is given by:

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)$$

Pairwise F-measure is a more commonly used measure, and *BCubed F-measure* is the formal evaluation measure for the IJB-B dataset. The difference between the two is that *Pairwise F-measure* puts relatively more emphasis on large clusters because the number of pairs grows quadratically with cluster size, while under *BCubed F-measure* clusters are weighted linearly based on their size.

2) *Evaluation on the LFW Dataset*: LFW is quite an imbalanced dataset, with only 1,680 classes (individuals) containing more than one face. Since we cannot assume that our datasets will be well balanced, we do the experiments on the whole LFW dataset.

The number of clusters C is dynamically selected during the update of the ConPaC, but it is required as an input parameter for *k*-means and spectral clustering. So we first evaluate their performance with the ground-truth or the true number of clusters C , $C = 5,749$. Then we repeat the clustering with several different values and report the one that gives the best performance.

Table II shows that the performance of *k*-means and spectral clustering is poor with the ground-truth C . This is because



Fig. 12: Example clusters by the proposed clustering algorithm on two of the experiments in IJB-B clustering protocol, IJB-B-32 and IJB-B-1024. The first and the second row show an impure and pure cluster on IJB-B-32 dataset, respectively. The third row and the fourth row show an impure and pure cluster on IJB-B-1024 dataset, respectively. For impure clusters, red bounding boxes indicate the images are from a different identity.

these two algorithms do not perform well with unbalanced data. After tuning the parameter C , the proposed algorithm still performs better than competing algorithms. Notice that agglomerative clustering using average linkage performs much better than k-means and spectral clustering. This is consistent with our observation in Section IV-B since agglomerative clustering also doesn't assume a balanced dataset and focuses on the pairwise similarities. We also compare the results with that reported in Triplet Probabilistic Embedding (TPE) [33], which uses agglomerative clustering on a different representation. For the run-time, since the sizes of clusters in the LFW dataset are mostly very small, the time complexity of ConPaC is low and it takes less than one minute to finish.

Some example clusters by ConPaC for LFW are shown in Figure 11, where the first two rows show two impure clusters while the other two show two pure clusters. Face images in these clusters have different illumination conditions, backgrounds, and poses. Given these challenges, the algorithm still groups most images successfully according to true identities. For the two impure clusters, the mis-grouped images are very similar to other images in the same cluster.

3) *Evaluation on the IJB-B Dataset:* The results of the 7 experiments in the IJB-B clustering protocol are shown in Table III. Since B-Cubed F-measure is the adopted evaluation measure in the IJB-B protocol, we only report the result with the best B-Cubed F-measure for each algorithm. Because of the memory limit, some clustering algorithms cannot be tested on larger datasets. In such cases, we report the result as “-”. We set τ as 0.55, 0.6 and 0.65, respectively, for IJB-B-32, IJB-B-64 and IJB-B-128 and 0.7 for the other datasets.

As the number of identities increases, the F-scores of both competing and the proposed algorithm decrease. While the proposed algorithm shows a significant advantage in terms of F-score on the first few experiments, the gain diminishes as the number of clusters increases. As explained in Section V-B6, this decrease in performance with an increase in the number of clusters is mainly due to the brittleness of the representation.

Another thing worth noticing is that the number of clusters found by the proposed algorithm is much larger than the true number of clusters. This is because a large number of points are regarded as outliers by our algorithm and so they form

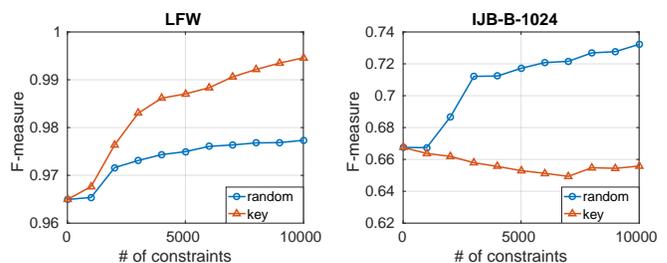


Fig. 13: Performance of the proposed clustering algorithm on LFW and IJB-B-1024 datasets after incorporating pairwise constraints. Both random and key constraints improve clustering performance in terms of pairwise F-score on the LFW dataset. However, for the IJB-B-1024 dataset, only random constraints are able to boost the performance.

singleton clusters. For this reason, we also report the number of “non-singleton” clusters in parentheses, which contain at least two points. The number of non-singleton clusters is closer to the true number of clusters.

Some example clustering results on the IJB-B-32 and IJB-B-1024 are shown in Figure 12. The first two rows show an impure and a pure cluster on IJB-B-32 dataset, respectively. The other two rows show an impure and a pure cluster on IJB-B-1024 dataset. Many face images in the impure clusters have very large pose variations and are thus badly aligned, diminishing the saliency of the representation.

4) *Semi-supervised Clustering:* As we mentioned in Section IV-E, pairwise constraints could be naturally incorporated into the framework of ConPaC without any modification of the algorithm. Therefore in this section, we assume that we have already been given a set of pairwise constraints and evaluate whether the side-information could improve the clustering performance. We consider two types of constraints:

- *Random Constraints:* must-links and cannot-links are picked randomly from ground-truth positive and negative pairs.
- *Key Constraints:* The similarities between every pair of faces are sorted. Must-links are picked by choosing the positive pairs (pairs from the same identity) with the lowest similarities and cannot-links are picked by choosing the negative pairs (faces from different identities) with the highest similarities.

TABLE III: Comparison of the F-measures of the proposed algorithms and other clustering algorithm on IJB-B datasets. The number of images in each dataset is given with the dataset name. Numbers of non-singleton clusters in the proposed algorithms are shown in parentheses. “-” means that algorithm cannot be tested due to memory limit.

(a) IJB-B-32 (1,026)					(b) IJB-B-64 (2,080)				
Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time	Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time
<i>k</i> -means	0.825	0.766	20	00 : 00 : 01	<i>k</i> -means	0.704	0.710	75	00 : 00 : 01
Spectral	0.630	0.720	30	00 : 00 : 01	Spectral	0.564	0.654	50	00 : 00 : 02
SSC	0.532	0.665	30	00 : 00 : 12	SSC	0.467	0.614	50	00 : 00 : 44
Affinity Propagation	0.376	0.534	86	00 : 00 : 01	Affinity Propagation	0.399	0.506	167	00 : 00 : 02
Agglomerative	0.787	0.795	30	00 : 00 : 01	Agglomerative	0.635	0.738	75	00 : 00 : 05
Rank-Order	0.697	0.769	81	00 : 00 : 03	Rank-Order	0.409	0.693	173	00 : 00 : 13
Approx. Rank-Order	0.223	0.526	85	00 : 00 : 02	Approx. Rank-Order	0.163	0.543	236	00 : 00 : 03
ConPaC (proposed)	0.868	0.828	96 (42)	00 : 00 : 03	ConPaC (proposed)	0.776	0.772	280 (93)	00 : 00 : 04

(c) IJB-B-128 (5,224)					(d) IJB-B-256 (9,867)				
Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time	Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time
<i>k</i> -means	0.520	0.655	75	00 : 00 : 04	<i>k</i> -means	0.489	0.624	150	00 : 00 : 12
Spectral	0.377	0.620	100	00 : 00 : 12	Spectral	0.333	0.562	150	00 : 01 : 00
SSC	0.275	0.547	100	00 : 05 : 19	SSC	0.278	0.498	150	00 : 26 : 30
Affinity Propagation	0.230	0.441	351	00 : 00 : 18	Affinity Propagation	0.241	0.434	658	00 : 01 : 66
Agglomerative	0.787	0.750	150	00 : 00 : 34	Agglomerative	0.671	0.725	350	00 : 01 : 57
Rank-Order	0.385	0.661	471	00 : 02 : 01	Rank-Order	0.351	0.653	1,033	00 : 06 : 37
Approx. Rank-Order	0.302	0.653	631	00 : 00 : 05	Approx. Rank-Order	0.214	0.645	13,55	00 : 00 : 07
ConPaC (proposed)	0.895	0.769	738 (243)	00 : 00 : 20	ConPaC (proposed)	0.888	0.721	1,862 (574)	00 : 00 : 48

(e) IJB-B-512 (18,251)					(f) IJB-B-1024 (36,575)				
Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time	Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time
<i>k</i> -means	0.429	0.587	500	00 : 00 : 46	<i>k</i> -means	0.423	0.572	1,000	00 : 03 : 01
Spectral	0.335	0.531	350	00 : 06 : 28	Spectral	0.265	0.495	750	00 : 45 : 43
SSC	0.237	0.450	500	02 : 35 : 21	SSC	-	-	-	-
Affinity Propagation	0.251	0.432	1,276	00 : 17 : 16	Affinity Propagation	0.241	0.423	2,500	01 : 37 : 38
Agglomerative	0.567	0.687	750	00 : 04 : 45	Agglomerative	0.544	0.696	1500	00 : 28 : 21
Rank-Order	0.188	0.638	1,958	00 : 23 : 49	Rank-Order	0.020	0.544	2,831	01 : 40 : 30
Approx. Rank-Order	0.214	0.569	3,758	00 : 00 : 12	Approx. Rank-Order	0.201	0.512	9,553	00 : 00 : 23
ConPaC (proposed)	0.756	0.656	3,981 (1,175)	00 : 02 : 10	ConPaC (proposed)	0.667	0.641	11,258 (2,303)	00 : 08 : 39

(g) IJB-B-1845 (68,195)				
Algorithm	Pairwise F-measure	BCubed F-measure	# of Clusters	Run-time
<i>k</i> -means	0.354	0.551	1500	00 : 11 : 49
Spectral	-	-	-	-
SSC	-	-	-	-
Affinity Propagation	-	-	-	-
Agglomerative	-	-	-	-
Rank-Order	0.005	0.267	4,084	01 : 12 : 25
Approx. Rank-Order	0.299	0.450	20,782	00 : 00 : 43
ConPaC (proposed)	0.611	0.634	15,227 (4,200)	00 : 51 : 33

Key constraints are difficult to acquire in realistic cases, as such they are merely used to test the upper bound of the improvement given constraints on pairs that could be misleading. In both cases, we use knowledge of the ground truth identity labels to sample an equal number of must-link and cannot-link constraints. We then test the performance of the algorithm with an increasing number of constraints. For random constraints, we run 10 trials and report the average performance. The results are reported in terms of pairwise F-score.

We test our algorithm in a semi-supervised scenario on LFW and IJB-B-1024 datasets. The results of semi-supervised clustering are shown in Figure 13. On LFW, for both random

constraints and key constraints, the constraints always boost the performance and the larger the number of constraints, the larger the improvement in F-score. This is because our algorithm tries to find clustering results that are most consistent with the unary potentials, and when more constraints are provided, the unary potentials can be trusted more. Additionally, the number of specified constraints in the experiment are actually very small. For example, 10,000 constraints account for only 0.011% of the total number of all the possible pairs in LFW. But due to message propagation, each pairwise constraint impacts all related pairs. Thus, even a small number of randomly picked constraints could boost the performance significantly. For the 10,000 random constraints, 98.33%

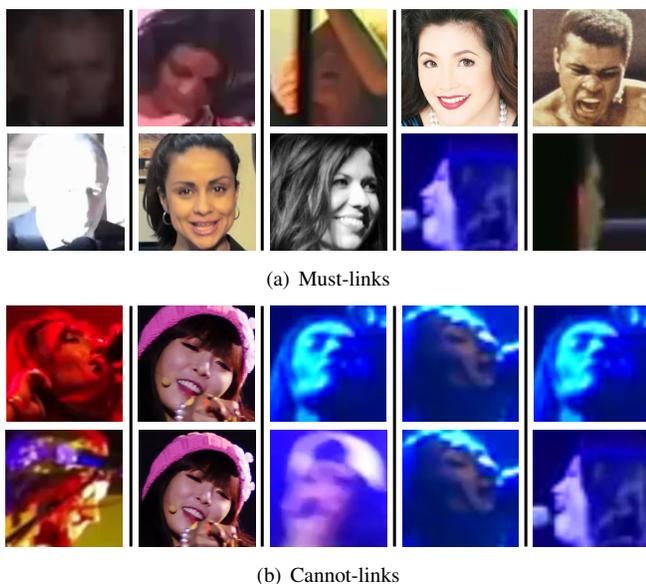


Fig. 14: Example pairs in the key constraints for IJB-B-1024 dataset. Many must-links have extremely different facial appearances, even though they may belong to the same identity. Cannot-links, on contrary, involve some wrong labels and many extremely low-quality images.

must-links and 99.99% cannot-links are satisfied at the end. For key constraints, 98.42% must-links and 99.94% cannot-links are satisfied.

On IJB-B-1024 dataset, as it is a larger dataset, 10,000 constraints account for only 0.0015% of the total number of pairs. The random constraints boost the performance significantly, but the key constraints do not offer any benefits. Furthermore, 99.93% must-links and 75.17% cannot-links are satisfied for random constraints, but only 3.02% must-links and 21.60% cannot-links are satisfied for 10,000 key constraints. Inspecting into the problem, we found that the selected must-links for key constraints on IJB-B-1024 usually have extremely different facial appearances, while images in cannot-links are mostly of very bad quality, as shown in Figure 14. Thus, the algorithm may not satisfy these constraints as they are quite incompatible with the rest of the dataset.

5) *k-NN variant for large datasets*: In this section, we test the run-time and performance of the *k-NN* variant of the proposed clustering. We use the same *k-d* tree library [66] as used in [2] to generate the approximate *k-NN* graph. We also use the same configuration for the *k-d* tree with $k = 200$ and we build four trees with a search size of 2,000. We first compare the performance of the algorithm using approximate *k-NN* graph and full graph (original algorithm) on LFW and IJB-B-1024. The clustering results are shown in Table IV. The proposed *k-NN* variant performs well on both datasets. For LFW, the pairwise F-measure is almost as good as the original algorithm. For IJB-B-1024, we observe a large shift of the distributions of similarities in the *k-NN* graph compared with original distributions, especially for impostor pairs, as shown in Figure 15. This is because the *k-d* tree is expected to select only similar pairs for building the *k-NN* graph, and many impostor pairs in IJB-B-1024 dataset turn out to be similar.

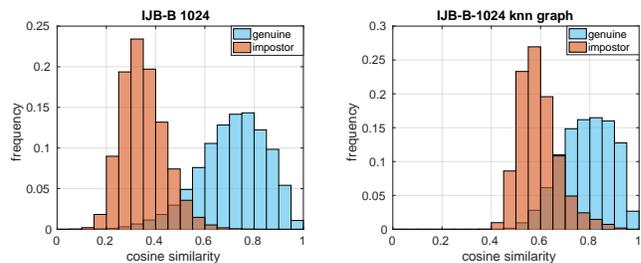


Fig. 15: The distributions of all genuine and impostor pairs in IJB-B-1024 and pairs in *k-NN* graph. Both the impostor and genuine distributions are shifted significantly in the *k-NN* graph.

TABLE IV: Performance of *k-NN* variant and original algorithm on small and large datasets.

Dataset	Version	Pairwise F-measure	# of clusters	Run-time
LFW	full graph	0.965	6,352	00 : 00 : 39
LFW	<i>k-d</i> tree	0.960	6,444	00 : 01 : 04
IJB-B-1024	full graph	0.668	8,094	00 : 08 : 39
IJB-B-1024	<i>k-d</i> tree	0.536	11,258	00 : 01 : 43
LFW + 1M	<i>k-d</i> tree	0.940	536,809	00 : 34 : 03
IJB-B-1024 + 1M	<i>k-d</i> tree	0.541	616,964	00 : 36 : 05

Using old threshold $\tau = 0.7$ would make the majority of pairs positive, thus we tune the threshold to $\tau = 0.75$ for *k-NN* variant when working on IJB-B-1024. Although a similar shift is observed on LFW dataset, but it is smaller and we found that there is no need for tuning the threshold. After tuning the threshold, the F-measure of *k-NN* variant is also close to the original one on IJB-B-1024. Using the same threshold τ , we then test the performance of the *k-NN* variant on 1 million unlabeled face images along with LFW or IJB-B. The 1 million face images is a subset of the same private dataset used in [2]. Since we do not have the labels for the 1 million dataset, we apply pairwise F-measure to only the subset for which we have labels (from LFW or IJB-B) but omit those for which we do not have labels, namely the 1 million distractor images. With such a big number of distractors, the resulting F-measure is almost unchanged for both datasets. Notice that for IJB-B-1024, the performance is, surprisingly, even better when given the distractors. A plausible explanation for this is that there are some high quality images in the 1 million distractors that are from the same identities as in IJB-B, which help to reveal the connections between the face images in IJB-B-1024, while the other distractor images make less impact on the clustering performance.

6) *Influence of the initial similarity matrix*: The motivation and distinguishing feature of our algorithm is that it only depends on the given pairwise similarities. In this subsection we want to investigate how the clustering performance is affected by the similarities and also how it is influenced by the choice of parameter τ .

We first define *similarity reliability* as the pairwise F-measure on adjacency matrix Z , where $Z_{ij} \in \{0,1\}$ is determined by thresholding the cosine similarity between X_i and X_j by τ . Different from Y , the graph represented by Z may not be transitive, so it does not necessarily correspond to a clustering result.

We then determine how the F-measures of Y and Z vary

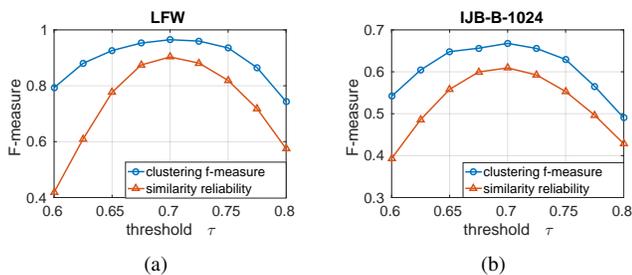


Fig. 16: Performance of the proposed algorithm with different threshold values on LFW and IJB-B-1024 datasets.

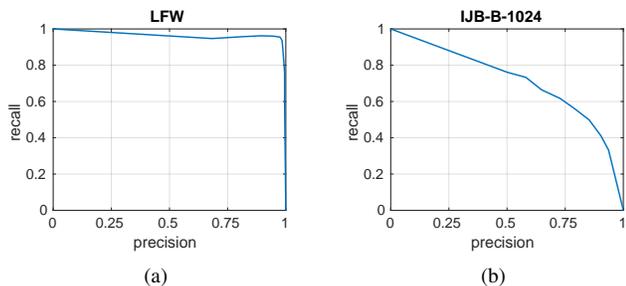


Fig. 17: Precision Recall curves for the proposed algorithm on LFW and IJB-B-1024 datasets.

with different values of threshold τ . The results are shown in Figure 16, and corresponding precision-recall curves are shown in Figure 17. We can see that the clustering performance changes smoothly with different parameter values. Furthermore, the F-measure of clustering result Y is highly correlated with that of Z . In other words, when the similarity matrix is reliable, the clustering does a better job, and vice versa.

To further see the relationship between the clustering performance and the pairwise similarities, we compare the F-measures of Y and that of Z on all the 7 experiments in IJB-B dataset. We find that the two F-measures are almost linearly correlated across all these experiments, with a correlation coefficient of 0.9998. Therefore, we can state that we have achieved our motivation to make full use of the input pairwise similarities, and that the decrease in clustering performance on IJB-B compared to LFW is due to the decrease in reliability of the input pairwise similarities, which in turn depends on the saliency of the face representation (feature vector).

VI. CONCLUSIONS

In this paper, we first trained a ResNet deep network architecture on CASIA-Webface and VGG-Face datasets. The representation from the proposed network shows a good performance on the BLUFR face verification benchmark. Using this representation, we proposed a new clustering algorithm, called Conditional Pairwise Clustering (ConPaC), which learns an adjacency matrix directly from the given similarity matrix. The clustering problem is modeled as a structured prediction problem using a Conditional Random Field (CRF) and is inferred by Loopy Belief Propagation. The proposed algorithm outperforms several well known clustering algorithms on LFW and IJB-B unconstrained datasets and it can also naturally

incorporate pairwise constraints to further improve clustering results. We also propose a k -NN variant of ConPaC which is capable of clustering millions of face images. Our future work would include finding better unary potentials for more robust face clustering and also incorporating pairwise constraints into the k -NN variant.

REFERENCES

- [1] J. C. Klontz and A. K. Jain, "A case study of automated face recognition: The Boston Marathon bombings suspects," *IEEE Computer*, vol. 46, no. 11, 2013.
- [2] C. Otto, D. Wang, and A. K. Jain, "Clustering millions of faces by identity," *IEEE Trans. on PAMI*, 2017.
- [3] A. Nech and I. Kemelmacher-Shlizerman, "Level playing field for million scale face recognition," *arXiv:1705.00393*, 2017.
- [4] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.
- [5] C. Whitelam, E. Taborsky, A. Blanton, B. Maze, J. Adams, T. Miller, N. Kalka, A. K. Jain, J. A. Duncan, K. Allen, J. Cheney, and P. Grother, "Iarpa janus benchmark-b face dataset," in *CVPR Workshop on Biometrics*, 2017.
- [6] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, M. Burge, and A. K. Jain, "Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus benchmark A," in *CVPR*, 2015.
- [7] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, 2004.
- [8] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv:1411.7923*, 2014.
- [9] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015.
- [10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [11] S. Liao, Z. Lei, D. Yi, and S. Z. Li, "A benchmark study of large-scale unconstrained face recognition," in *IJCB*, 2014.
- [12] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *CVPR*, 1991.
- [13] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. on PAMI*, vol. 23, no. 6, 2001.
- [14] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. on PAMI*, vol. 28, no. 12, 2006.
- [15] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on PAMI*, vol. 31, no. 2, 2009.
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
- [17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [18] C.-C. Hsu and C.-W. Lin, "Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data," *arXiv:1705.07091*, 2017.
- [19] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *CVPR*, 2014.
- [20] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *CVPR*, 2014.
- [21] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *NIPS*, 2014.
- [22] Y. Sun, X. Wang, and X. Tang, "Deeply learned face representations are sparse, selective, and robust," in *CVPR*, 2015.
- [23] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv:1502.00873*, 2015.
- [24] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation," in *ECCV*, 2012.
- [25] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.
- [26] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, 2010.
- [27] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *CVPR*, 2003.

- [28] J. Cui, F. Wen, R. Xiao, Y. Tian, and X. Tang, "Easalbum: an interactive photo annotation system based on face clustering and re-ranking," in *ACM SIGCHI Conference on Human factors in Computing Systems*, 2007.
- [29] Y. Tian, W. Liu, R. Xiao, F. Wen, and X. Tang, "A face annotation framework with partial clustering and interactive labeling," in *CVPR*, 2007.
- [30] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *CVPR*, 2009.
- [31] C. Zhu, F. Wen, and J. Sun, "A rank-order distance based clustering algorithm for face tagging," in *CVPR*, 2011.
- [32] R. Vidal and P. Favaro, "Low rank subspace clustering (lrscl)," *Pattern Recognition Letters*, vol. 43, 2014.
- [33] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa, "Triplet probabilistic embedding for face verification and clustering," in *BTAS*, 2016.
- [34] C. Otto, B. Klare, and A. Jain, "An efficient approach for clustering face images," in *ICB*, 2015.
- [35] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl *et al.*, "Constrained k-means clustering with background knowledge," in *ICML*, 2001.
- [36] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *NIPS*, 2002.
- [37] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *KDD*, 2004.
- [38] I. Davidson and S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in *European Conference on Principles of Data Mining and Knowledge Discovery*, 2005.
- [39] Z. Lu and M. A. Carreira-Perpinan, "Constrained spectral clustering through affinity propagation," in *CVPR*, 2008.
- [40] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in *KDD*, 2010.
- [41] S. Basu, I. Davidson, and K. Wagstaff, *Constrained clustering: Advances in Algorithms, Theory, and Applications*. CRC Press, 2008.
- [42] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," *Introduction to Statistical Relational Learning*, 2006.
- [43] J. Lafferty, A. McCallum, F. Pereira *et al.*, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001.
- [44] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," in *CVPR*, 2004.
- [45] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *International Journal of Computer Vision*, vol. 80, no. 1, 2008.
- [46] V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," *NIPS*, 2011.
- [47] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, 1989.
- [48] I. K. K. M. Liang-Chieh Chen, George Papandreou and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *ICLR*, 2015.
- [49] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [50] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *International Journal of Computer Vision*, vol. 40, no. 1, 2000.
- [51] B. J. Frey and D. J. MacKay, "A revolution: Belief propagation in graphs with cycles," *NIPS*, 1998.
- [52] C. Yanover and Y. Weiss, *Approximate Inference and Protein-folding*. Hebrew University of Jerusalem, 2002.
- [53] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, 2005.
- [54] C. Sutton, A. McCallum *et al.*, "An introduction to conditional random fields," *Foundations and Trends in Machine Learning*, vol. 4, no. 4, 2012.
- [55] D. Wang, C. Otto, and A. K. Jain, "Face search at scale," *IEEE Trans. on PAMI*, 2016.
- [56] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *NIPS Workshop on Big Learning*, 2011.
- [57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.
- [58] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on PAMI*, vol. 22, no. 8, 2000.
- [59] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," in *NIPS*, 2001.
- [60] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, 1996.
- [61] C. Cheng, J. Xing, Y. Feng, D. Li, and X.-D. Zhou, "Bootstrapping joint bayesian model for robust face verification," in *ICB*, 2016.
- [62] J.-J. Lv, C. Cheng, G.-D. Tian, X.-D. Zhou, and X. Zhou, "Landmark perturbation-based data augmentation for unconstrained face recognition," *Signal Processing: Image Communication*, 2016.
- [63] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, 2007.
- [64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.
- [65] E. Amigó, J. Gonzalo, J. Artilles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information Retrieval*, vol. 12, no. 4, 2009.
- [66] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. on PAMI*, vol. 36, no. 11, 2014.



Yichun Shi received his B.S degree in the Department of Computer Science and Engineering at Shanghai Jiao Tong University in 2016. He is now working towards the Ph.D. degree in the Department of Computer Science and Engineering at Michigan State University. His research interests include pattern recognition and computer vision.



Charles Otto received his B.S. and Ph.D. degrees in the Department of Computer Science and Engineering at Michigan State University in 2008 and 2016, respectively. He was a research engineer at IBM during 2006-2011. He is currently employed at Noblis, Reston, VA. His research interests include pattern recognition, and computer vision.



Anil K. Jain is a University distinguished professor in the Department of Computer Science and Engineering at Michigan State University. His research interests include pattern recognition and biometric authentication. He served as the editor-in-chief of the IEEE Transactions on Pattern Analysis and Machine Intelligence (1991-1994), a member of the United States Defense Science Board, and is a member of the National Academy of Engineering.