To appear in the IEEE Trans PAMI, 2017

Clustering Millions of Faces by Identity

Charles Otto, *Student Member, IEEE*, Dayong Wang, *Member, IEEE,* and Anil K. Jain, *Fellow, IEEE*

Abstract—Given a large collection of unlabeled face images, we address the problem of clustering faces into an unknown number of identities. This problem is of interest in social media, law enforcement, and other applications, where the number of faces can be of the order of hundreds of million, while the number of identities (clusters) can range from a few thousand to millions. To address the challenges of run-time complexity and cluster quality, we present an approximate Rank-Order clustering algorithm that performs better than popular clustering algorithms (k-Means and Spectral). Our experiments include clustering up to 123 million face images into over 10 million clusters. Clustering results are analyzed in terms of external (known face labels) and internal (unknown face labels) quality measures, and run-time. Our algorithm achieves an F-measure of 0.87 on the LFW benchmark (13K faces of $5,749$ individuals), which drops to 0.27 on the largest dataset considered (13K faces in LFW + 123M distractor images). Additionally, we show that frames in the YouTube benchmark can be clustered with an F-measure of 0.71. An internal per-cluster quality measure is developed to rank individual clusters for manual exploration of high quality clusters that are compact and isolated.

Index Terms—face recognition, face clustering, deep learning, scalability, cluster validity

1 Introduction

In this work, we attempt to address the following problem: Given a large number of unlabeled face images, cluster them into the individual identities present in this data. This situation is encountered in a number of different application scenarios ranging from social media to law enforcement, where the number of faces in the collection can be of the order of hundreds of million. Often, the labels attached to the face images are either missing or contain noise. The number of clusters or the unknown number of identities can range from a few thousand to hundreds of millions, leading to difficulties in terms of both run-time and clustering quality.

Considering social media, Facebook reported that 350 million images are uploaded per day on average[1], and of those images, a large number may reasonably be assumed to be images of people. In social media some identity information may be provided via tagging, but in general this is incomplete and may be inaccurate. We consider grouping face images into discrete identities as one possible approach for organizing this large volume of data.

In forensic investigations, triaging large-scale face collections is also an emerging problem. Few examples are more relevant than the Boston Marathon bombing [1], where tens of thousands of images and videos needed to be analyzed during a time sensitive investigation [2]. Other common cases that require the investigation of large media collections include identifying perpetrators and victims in child exploitation cases[2], an understanding of which individuals exist in a collection of social media (such as imagery from gang and terrorist networks), and organizing media collections from hard drives (personal computers or servers).

In both social media, and forensic investigations we expect the unknown number of individual identities present in a dataset to be large, which is challenging from a scalability perspective since runtimes tend to be related to the number of clusters. Additionally, we expect the number of images per individual to be unbalanced (some people may appear very often, others much less frequently), which is challenging for e.g. clustering algorithms like k-means which tend to generate similar sized clusters. It can also be assumed that the quality of images in terms of pose, illumination, occlusion, etc. being considered is relatively low, since social media images, images taken at public events etc. are not generally captured in the most favorable conditions for face recognition. Following recent progress in unconstrained face recognition, we attempt to mitigate the difficulty of the underlying face clustering problem by using a state-of-the-art convolutional neural network based face representation [11].

Even using a strong face representation, accuracy is not perfect on verification tasks (particularly when considering difficult data). Zhu et al. [7] reported success in clustering collections of personal photographs using a Rank-Order clustering method which develops a distance measure based on shared nearest-neighbors of face images being compared (since direct feature vector-to-feature vector distances may be inaccurate given the difficulty of the face recognition task). However, in addition to the problem of poor face quality, large scale face clustering tasks (on the order of 100 million face images) are inherently difficult in terms of scalability (run-time). We develop a version of the rank-order clustering algorithm of Zhu et al. [7] leveraging an approximate nearest neighbor method for improved scalability, and simplifying the actual clustering procedure to achieve improved scalability and clustering accuracy.

We evaluate large-scale clustering performance by combining the well-known Labeled Faces in the Wild (LFW) dataset [12] with up to 123M unlabeled images (downloaded from the web), and clustering the augmented dataset. Additionally, considering that even a reasonably accurate clustering of a truly large dataset may still result in too many clusters to be manually investigated, we investigate per-cluster "internal" quality measures (which do not require external labels on face images) to identify a subset of "good" clusters (relatively compact and isolated), for manual exploration. In addition to large-scale clustering on unconstrained still face images, we perform preliminary investigations of clustering video frames leveraging the YouTube Faces (YTF) database [13], clustering hundreds of thousands of video frames.

The perceived contributions of this paper include: (i) an updated clustering algorithm, improving on the method presented by Zhu et al. [7] using an approximate nearest neighbor method for

1. https://goo.gl/FmzROn
2. http://www.nist.gov/itl/iad/ig/chexia-face.cfm

TABLE 1: A summary of related studies on face clustering.

Publication	Features	Clustering method	# Face images	# Subjects
Ho et al. [3]	Gradient and pixel intensity features	Spectral clustering	1,386	66
Zhao et al. [4]	2DHMM + contextual	Hierarchical clustering	1,500	8
Cui et al. [5]	LBP, clothing color + texture	Spectral	400	5
Tian et al. [6]	Image + contextual	Partial clustering	1,147	34
Zhu et al. [7]	Learning-based descriptor [8]	Rank-order	1,322	53
Vidal and Favaro [9]	Joint subspace learning and clustering	-†	2,432	38
Otto et al. [10]	Component-based features, commercial face matcher	k-Means, spectral, rank-order	1M	195,494
Ours	Deep features [11]	Approximate rank-order	123M	Unknown‡

† In this work a unified algorithm is used for representation and clustering
‡ Due to the nature of the dataset used (face images blindly harvested from the Internet), we do not know the true number of identities, as is the case in practical scenarios.

improved scalability, which also attains better clustering accuracy, (ii) large-scale face clustering experiments using a state-of-the-art face representation learned for large scale supervised face recognition based on deep networks [11], (iii) a preliminary investigation of the applicability of the presented face clustering method to video, and (iv) definition of a per-cluster quality measure suitable for prioritizing a subset of clusters out of millions of detected clusters.

2 Background

2.1 Face Clustering

The clustering problem, a tool for exploratory data analysis, has been well studied in pattern recognition, statistics, and machine learning literature (Jain [14] provides a survey). Less studied is the challenging problem of clustering face images, especially when both the number of images and the number of clusters are very large. An important consideration in clustering (and classifying) face images is that since there is no universally agreed upon face representation or distance metric, the clustering results depend not only on the choice of clustering algorithm, but also on the quality of the underlying face representation and metric. Table 1 lists prior work on face clustering, with the face representation and clustering algorithm used, along with the largest dataset size employed in terms of face images, as well as number of subjects.

Ho et al. [3] developed variations on spectral clustering wherein the affinity matrix is computed based on (i) assuming a Lambertian object with fixed camera/object positioning, and then computing the probability that two face images are of the same object (same convex polyhedral cone in the image space), or (ii) the local gradients of the images being compared; evaluation is done on the Yale-B and PIE-66 datasets.

Zhao et al. [4] clustered personal photograph collections. Their approach combines a variety of contextual information including time based clustering, and the probability of faces of certain people to appear together in images, with identity estimates obtained via a 2D-HMM, and hierarchical clustering results based on body detection; a dataset of $1,500$ face images of 8 individuals is used for evaluation.

Cui et al. [5] developed a semi-automatic tool for annotating photographs, which employs clustering as an initial method for organizing photographs. LBP features are extracted from detected faces, and color and texture features are extracted from detected bodies. Spectral clustering is performed, and the clustering results can then be manually adjusted by a human operator. Evaluation is done on a dataset consisting of 400 photographs of 5 subjects. Tian et al. [6] further developed this approach, incorporating a probabilistic clustering model, which incorporates a "junk" class, allowing the algorithm to discard clusters that do not have tightly distributed samples.

Zhu et al. [7] developed a dissimilarity measure based on the rankings of two faces being compared in each face's nearest neighbor lists (formed using a basic distance metric), and perform hierarchical clustering based on the resulting rank-order distance function. The feature representation used is the result of unsupervised learning [8]. The clustering method is evaluated on several small datasets (the largest of which contains only $1,322$ face images). Wang et al. [15] primarily develop an approximate k-NN graph construction method; in one of their experiments they apply this method to construct the nearest neighbor lists required by [7], on a dataset containing LFW and an additional 500K unlabeled face images, and use the rank-order distance measure to produce an improved k-NN graph (but do not perform hard assignment of faces into clusters).

Vidal and Favaro [9] developed a joint subspace learning and clustering approach. It derives several subspaces from the input dataset which best capture clusters in the data. They evaluate the method on the extended Yale-B database.

In related applications, Bhattarai et al. [16] develop a semi-supervised method for organizing datasets for improved retrieval speed via hierarchical clustering. Tapaswi et al. [17] address organization of video frames, performing both within video and cross-video clustering, incorporating constraints from face tracking and common video editing patterns. Schroff et al. [18] give some qualitative results of clustering personal photos using a deep learning based face representation.

There have recently been some additional work on larger-scale face recognition problems [19], considering datasets on the order of a million images. Some experimental work in face clustering has considered hundreds of thousands of images, while some general object clustering tasks have used datasets on the order of billions of images [20]. In cases where the true number of clusters is known a priori, that number is typically orders of magnitude lower than the number of images. In general, the evaluation methods used to determine how well clustering algorithms perform (when true labels are available) are split. In some cases the clustering accuracy is used [3], in others precision/recall [4], and in still others normalized mutual information is employed [7].

2.2 General Image Clustering

For clustering images in general, rather than faces in particular, Liu et al. [20] (i) extracted Haar wavelet features from images, (ii) applied a distributed algorithm consisting of an approximate nearest neighbor step, (iii) generated an initial set of clusters by applying a distance threshold to the nearest neighbor lists, and

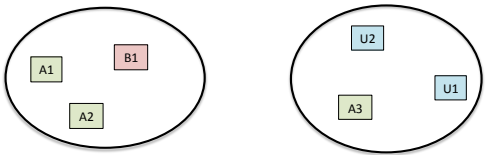

Fig. 1: Diagram of a possible clustering configuration, used to illustrate evaluation metrics. Six samples are partitioned into 2 clusters; A1, A2, and A3 are labeled with the same identity, sample B1 is labeled with a different identity, and samples U1 and U2 are unlabeled.

(iv) applied a union-find algorithm to get a final set of clusters. Clustering was performed on approximately 1.5 billion unlabeled images, along with an evaluation on $3,385$ labeled images. The main goal of the procedure was to group images into sets of near duplicates, but the total number of such sets in the 1.5 billion image dataset was unknown.

Gong et al. [21] develop a version of k-means clustering which is suitable for handling large datasets by encoding their feature vectors to binary vectors, and then using an indexing scheme to support constant time lookup of cluster centers for the assignment step of k-means. They apply their binary k-means algorithm to a subset of the ImageNet dataset, containing 1.2 million general object images in $1,000$ classes.

Foo et al. [22] consider a related problem, the detection of near-duplicate images in large datasets. In this case, rather than grouping images of people by identity, the goal is to identify near-duplicate images, which may be the result of various image processing operations, such as cropping, rotation, colorspace conversion, etc. Their image representation consists of applying a visual words approach to local PCA-SIFT descriptors, indexed with a Locality Sensitive Hashing (LSH) scheme. The clustering method used is a union-find algorithm. Evaluation was performed by generating a synthetic set of near duplicate images, and performing clustering in the presence of a separate noise set; the largest dataset used contained $300,000$ images.

2.3 Approximate Nearest Neighbor Methods

A common problem in some of the well-known clustering methods is finding nearest neighbor sets for all n samples in a dataset. Naively, the runtime is $O(n^2)$, which is a problem for large n. This can be considered an instance of the k-NN graph construction problem, or alternatively it can be considered a set of n approximate nearest neighbor searches. For both of these cases, approximation methods are available in the literature.

2.3.1 k-nn Graph Construction

One approximation method for computing the full k-NN graph is given by Chen et al. [23]. The algorithm is a procedure based on recursive subdivision of the feature space via Lanczos bisection. We use a parallelized version of this algorithm, presented in [10], which branches at each recursive subdivision, handling both halves in separate threads.

This algorithm achieves improved runtime over the brute-force method by skipping some sets of comparisons (the portion of comparisons at each split between samples in opposite partitions, not included in the overlap set), and as such the runtime is a function in the degree of overlap chosen.

2.3.2 Randomized k-d Tree

In addition to k-NN graph construction, we may consider building nearest neighbor lists for the entire dataset as n discrete nearest neighbor search problems, and improve the total runtime by employing an approximate nearest neighbor search method. Among various approximate nearest neighbor algorithms, one classic family is partitioning tree-based approaches. They build on the classic k-d tree algorithm which develops an index that subdivides the feature space by selecting a subset of features to split the data on, typically by conducting non-exhaustive searches of such a tree. The randomized k-d tree algorithm [24] improves efficiency by building multiple randomized k-d tree indices, then searches those indices in parallel, stopping the search after a specified number of tree nodes have been visited.

2.4 Clustering Evaluation

In evaluating clustering performance, since we use a pre-defined definition of "correct" clustering (clustering by identity), we can evaluate accuracy in terms of clusters corresponding to known identity labels. *External* measures for evaluating clustering quality rely on identity labels; we will use pairwise precision/recall since it can be computed efficiently. Run time is also an important evaluation metric.

Pairwise precision is defined as the fraction of pairs of samples within a cluster (considering all possible pairs) which are of the same class (have the same identity), over the total number of same-cluster pairs within the dataset. In Figure 1, (A1, A2) is a matching pair, and (A1, B1) and (A2, B1) are mismatched pairs.

Pairwise recall is defined as the fraction of pairs of samples within a class (considering all possible pairs) which are placed in the same cluster, over the total number of same-class pairs in the dataset. In Figure 1 (A1, A2) is a same class pair in the same cluster, while (A1, A3) and (A2, A3) are same-class pairs in different clusters.

These measures capture two types of error, a clustering which places all samples as individual clusters will have high precision, but low recall, while a clustering which places all samples in the same cluster will have high recall, but low precision. The two numbers can be summarized using F-measure, defined as $F = 2 \times (Precision \times Recall)/(Precision + Recall)$.

We extend these measure to handle partially labeled data, as encountered in large-scale clustering problems, by simply omitting the unlabeled data from evaluation, to the extent possible. In our experiments, partially labeled data occurs when we mix LFW face images (with known labels) against a large collection of faces downloaded from the web with unknown labels.

We define modified pairwise recall by simply not counting whether or not unlabeled identities are grouped together. For precision, we consider labeled-unlabeled pairs (e.g. (A3, U1) and (A3, U2) in Figure 1) mismatches, and omit unlabeled-unlabeled pairs (i.e. (U1, U2)) from the calculation. So, rather than considering all possible pairs in a given cluster, we omit any unlabeled-unlabeled pairs from the total. In the right cluster in Figure 1, we would only use pairs (A3, U1) and (A3, U2) to calculate the modified precision. The modified precision in Figure 1 is then $1/5$ (only the A1-A2 pair is correct, the U1-U2 pair is not counted), the modified recall is $1/3$, only class A has more than one sample (and is labeled), and of the class A pairs, only A1 and A2 are in the same cluster.

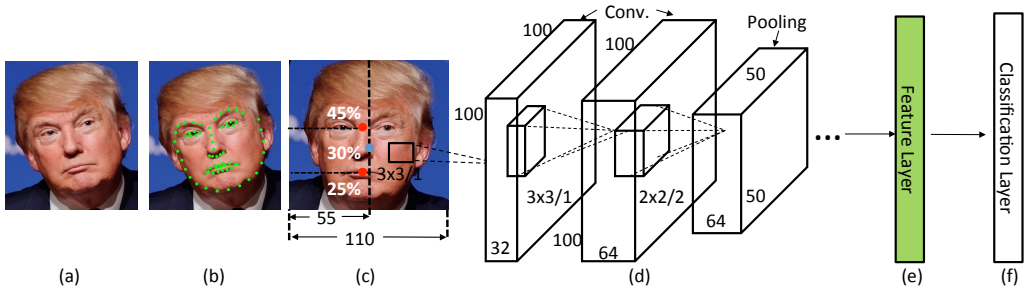

Fig. 2: Face representation. An RGB image is input (a), keypoints are detected (b), the image is normalized following the procedure described in [11] (c), the normalized image is input to a convolutional neural network (d), and the 320-dimensional output of the final average-pooling layer is used as the face representation (e). An N-way softmax classification layer (f) is used during training only.

3 Proposed Face Clustering Approach

3.1 Face Representation

Since we are clustering faces captured under unconstrained conditions, we leverage a deep convolutional neural network for our face representation following the success of such methods by various researchers on the LFW benchmark[3]. Many deep learning approaches have been successfully applied to the LFW benchmark; however, most leverage private training sets. In our case, we use the architecture described in [11] and train the network directly on aligned face images from the publicly available CASIA-webface dataset [25]. Results on both the LFW and IJB-A [26] benchmarks, and under larger-scale face retrieval scenarios, using this trained network, were shown to be reasonably competitive in [11] (reaching 96.96% overall accuracy on the standard LFW protocol), compared to the best approaches on LFW, particularly considering the different scales of training data involved.

The feature extraction process is outlined in Figure 2. Given an input image, 68 facial landmarks are detected using the DLIB implementation of Kazemi and Sullivan's [27] ensemble of regression trees method. Image normalization is performed based on the detected keypoints, in particular in-plane rotation is corrected based on the angle between the eyes, the eye line is placed at 45% of image height from the top of the image, the mouth line is placed at 25% of image height from the bottom of the image, the midpoint of all detected points is centered in the x dimension, the aligned image is scaled to 110×110, and the center 100×100 region is the final normalized image.

The normalized image is passed as input to a convolutional neural network following a very deep architecture [28], with a total of 10 convolution layers, and small (3×3) filters. The architecture consists of pairs of convolutional layers followed by max-pooling layers, repeated 4 times, then a final 2 convolutional layers followed by an average pooling layer, with ReLU neurons following all convolutional layers, except for the last one. The 320-dimensional output of the final average pooling layer is used as our feature vector, and during training is fed into a fully connected layer (regularized via dropout), followed by a softmax loss. Only the 320-dimensional output of the average-pooling layer is used in our clustering experiments.

The network is trained using $404,992$ face images of $10,533$ subjects from the CASIA-webface dataset (the images for which face alignment was performed successfully), in minibatch stochastic gradient descent. The loss layer used for training is a single softmax loss function. The weight decay of all layers is set to 5×10^{-4}, and the learning rate for stochastic gradient descent (SGD) is initialized to 10^{-2}, and gradually reduced to 10^{-5}. The network is implemented using the cuda-convnet2 library[4].

3.2 Clustering Method

A large number of clustering methods have been proposed in the literature based on squared-error, mixture models, nearest neighbor and graph-theoretic approaches [14]. Based on evaluation of different approaches for face clustering in [10], we leverage an approximate version of the rank-order clustering algorithm proposed by Zhu et al. [7]. We present the original algorithm in detail, then our modified version.

3.2.1 *Rank-Order Clustering*

The rank-order clustering algorithm proposed by Zhu et al. [7], similar to the method of Gowda and Krishna [29], is roughly a form of agglomerative hierarchical clustering, using a nearest neighbor based distance measure. The overall flow of the algorithm is to initialize all samples to be separate clusters, compute distances between pairs of clusters, merge those for which the computed distances are below a threshold, then iteratively recompute a new set of cluster-to-cluster distances, and perform merges based on the new distances. This requires defining a cluster-to-cluster distance metric. In the algorithm, the distance between two clusters is considered to be the minimum distance between any two samples in the clusters.

The first distance measure used in Rank-Order clustering is given by:

$$d(a,b) = \sum_{i=1}^{O_a(b)} O_b(f_a(i)), \tag{1}$$

where $f_a(i)$ is the i-th face in the neighbor list of a, and $O_b(f_a(i))$ gives the rank of face $f_a(i)$ in face b's neighbor list, where the nearest neighbor lists are generated according to some underlying distance measure (we use Euclidean distance). This asymmetric distance function is then used to define a symmetric distance between two faces, a and b, as:

$$D(a,b) = \frac{d(a,b) + d(b,a)}{\min(O_a(b), O_b(a))}. \tag{2}$$

An additional cluster-level normalized distance measure is used to effectively constrain merges to local neighborhoods. In

3. http://vis-www.cs.umass.edu/lfw/results.html

4. https://code.google.com/p/cuda-convnet2/

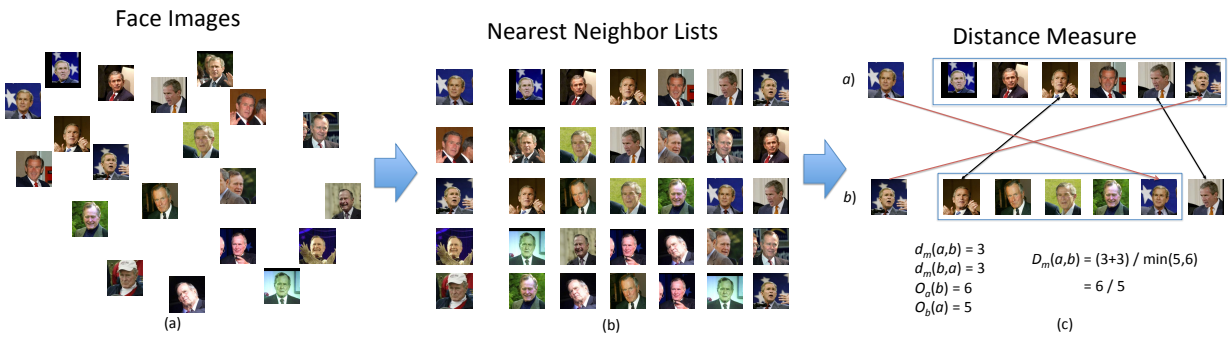

Fig. 3: Approximate Rank-Order clustering. Given a set of unlabeled face images (a), nearest neighbor lists are computed for each image (b); nearest neighbor lists are then used to compute distances between faces (c). (b) shows the nearest neighbor lists of only five faces in (a). $d_m(a,b)$ (Eq. 5) is the asymmetric distance between faces a and b whereas $D_m(a,b)$ (Eq. 6) is the symmetric distance between faces a and b.

particular, the minimum distance between any two points in a pair of clusters is computed, and divided by the average distance, as:

$$D^N(C_i, C_j) = \frac{1}{\phi(C_i, C_j)} \times d(C_i, C_j) \tag{3}$$

where

$$\phi(C_i, C_j) = \frac{1}{|C_i| + |C_j|} \times \sum_{a \in C_i \cup C_j} \frac{1}{K} \sum_{k=1}^{K} d(a, f_a(k)) \tag{4}$$

that is, $D^N(C_i, C_j)$, is the minimum distance between any two points in clusters i and j divided by the average of each point in C_i or C_j to their K nearest neighbors. A threshold of 1 is always used for this function, with the local neighborhood size K left as a free parameter, so effectively the test is whether or not the minimum distance between any two points in the clusters is below the average distance of all points in both clusters to their K nearest neighbors.

The symmetric rank order distance function gives low values if the two faces are close to each-other (face a ranks high in face b's neighbor list, and face b ranks high in face a's neighbor list), and have neighbors in common (high ranking neighbors of face b also rank highly in face a's neighbor list). After distances are computed, clustering is performed by initializing every face image to its own cluster, then computing the symmetric distances between each cluster, along with the cluster-level normalized distances, and merging any clusters with distances below both specified thresholds. Then, nearest neighbor lists for any newly formed clusters are merged, and distances between the remaining clusters are computed again iteratively, until no further clusters can be merged. In this case, rather than specifying the desired number of clusters C, a distance threshold is specified for the rank-order distance measure, along with a neighborhood size for the cluster-level normalized distance; it is these parameters which determines the specific number of clusters for a particular dataset being clustered, and their effective values are empirically determined. We use our own implementation of this algorithm.

In terms of run-time, computing the full nearest neighbor lists for each sample incurs an $O(n^2)$ cost. Additionally, the actual clustering step used here is iterative, with cost per iteration proportional to the current number of clusters squared (with number of clusters starting at n and decreasing across iterations), so both the nearest neighbor computation, and the clustering step itself are costly with increasing dataset size.

3.2.2 *Proposed Approximate Rank-Order Clustering*

The Rank-Order clustering method has an obvious scalability problem in that it requires computing nearest neighbor lists for every sample in the dataset, which has an $O(n^2)$ cost if computed directly. Although various approximation methods exist for computing nearest neighbors, they are typically only able to compute a short list of the top k nearest neighbors efficiently, rather than exhaustively ranking the dataset. We use the FLANN library implementation of the randomized k-d tree algorithm [30] to compute a short list of nearest neighbors.

Applying approximation methods for faster nearest neighbor computation then requires some modification of the original Rank-Order clustering algorithm. In particular, rather than considering all neighbors in the summation equation (1), we sum up to at most the top k neighbors (under the assumption that cluster formation relies on local neighborhoods). Additionally, rather than using the cluster-level normalized distance from the original algorithm of Zhu et al. [7], we enforce locality by only computing approximate rank-order distances between pairs of samples for which both are within the other sample's top-200 nearest neighbors.

Further, we note that if only a short list of the top-k neighbors is considered, the presence or absence of a particular example on the short list may be more significant than the sample's numerical rank. As such, we consider a distance measure based on directly summing the presence/absence of shared nearest neighbors, rather than the ranks, resulting in the following distance function:

$$d_m(a,b) = \sum_{i=1}^{\min(O_a(b),k)} I_b(O_b(f_a(i)), k), \tag{5}$$

where $I_b(x,k)$ is an indicator function with a value of 0 if face x is in face b's top k nearest neighbors, and 1 otherwise. In practice, we find that this modification leads to better clustering accuracy compared to summing the ranks directly, as in the original formulation. Effectively, this distance function implies that the presence or absence of shared neighbors towards the top of the nearest neighbor list (say within the top-200 ranks) is important, while the numerical values of the ranks themselves are not.

The normalization procedure employed in the original algorithm (only summing up to the rank of the other sample being compared, and dividing by $\min(O_a(b), O_b(a))$) is still effective, and contributes to more accurate clustering results even with this

modification to the original algorithm. The combined modified distance measure is defined as:

$$D_m(a,b) = \frac{d_m(a,b) + d_m(b,a)}{\min(O_a(b), O_b(a))}. \quad (6)$$

Additionally, to improve the runtime of the clustering step itself, as mentioned earlier we 1) only compute distances between samples which appear in each other's nearest neighbor lists, and 2) only perform one round of merges of individual faces into clusters, rather than performing multiple merge iterations as in the original algorithm. This means that compared to the original algorithm which has a runtime of C^2 per clustering iteration, we only perform one iteration of clustering, and additionally only check for merges on a subset of all possible pairs (since we consider the 200 nearest neighbors for each sample). This results in a final runtime of the clustering step (assuming pre-computed nearest neighbors) of $O(n)$.

The final clustering procedure we employ is then:

1) Extract deep features for every face in the dataset
2) Compute a set of the top-k nearest neighbors for each face in the dataset
3) Compute pairwise distances between each face, and those faces in its top-k nearest neighbor list for which the face is also on the neighbor's nearest neighbor list, following equation 6
4) Transitively merge all pairs of faces with distances below a threshold

Selecting a threshold to determine the number of clusters, C in a given dataset is one of the perennial difficult issues in data clustering. In practical applications we cannot assume that the true number of clusters will be known a priori, therefore in the absence of a robust procedure for determining the true number of clusters, we simply evaluate our algorithm at several effective values of C and report the best results attained in our experiments.

3.3 Per-Cluster Quality Evaluation

Our overall goal is to facilitate the investigation of very large collections of unlabeled face images. We have proposed clustering face images by identity as a first approach, but for very large datasets even clustering by identity may leave too many clusters for manual exploration. We attempt to address this issue by using internal cluster validity measures to identify a subset of "good" individual clusters, suitable for manual investigation.

In practical applications where the dataset is completely unlabeled, evaluating clustering according to external labels is not possible. But, there is a body of work in the literature on different internal cluster quality measures [31] which attempt to characterize cluster quality without the use of labels. These measures can typically be understood as measures of either *compactness* (how well the cluster members are grouped together in terms of pairwise similarity), or *isolation* (how well different clusters are separated from each other in terms of inter-cluster similarity). Additionally, we can make a distinction between evaluating the overall quality of a given clustering of a dataset, and evaluating the quality of individual clusters in a particular clustering; we will use per-cluster quality measures as a means of ranking individual clusters.

When dealing with very large datasets, one fundamental concern is run-time. It is generally infeasible to compute distances between all samples in the dataset, and additionally infeasible to compute distances between all clusters in cases where both the dataset and number of clusters present in the dataset are large. In this case, we are pre-computing a k-nearest neighbor graph, so it is natural to consider graph-based quality measures, and alleviate computational concerns by using the pre-computed graph.

Coverage [32] is defined as the fraction of intra-cluster edges present out of the complete set of edges in the graph. We modify this for use as a per-cluster quality measure by just considering nodes in the current cluster, i.e. define per-cluster coverage as the fraction of edges out-bound from nodes in the current cluster which link to other nodes in the cluster. Modularization quality (MQ) [33] is defined as the difference between an inter-cluster connectivity measure (the fraction of edges present between nodes in a cluster out of possible edges in a complete graph of those nodes), and an intra-cluster connectivity measure (average fraction of cross-cluster edges present out of possible edges between each pair of clusters in a complete graph).

These graph-based measures are formulated solely in terms of the presence or absence of edges between certain vertices. But we also find motivation to look at some simple distance based measures by observing that in some cases, clusters pairs of images which have a low rank-order distance, have a relatively high Euclidean distance. We will therefore also consider simple compactness and isolation measures based on the distances between edges present in the k-NN graph, primarily the average distance of samples to other samples in the same cluster in their nearest neighbor lists (average intra-cluster distance), and the average distance of samples in a cluster to samples outside that cluster in their nearest neighbor lists (average inter-cluster distance).

4 Datasets

Our clustering experiments use several unconstrained face datasets, the CASIA-webface face dataset [25] for training the deep network feature representation, the Labeled Faces in the Wild (LFW) [12] and YouTubeFaces (YTF) [13] datasets for clustering evaluation, and a collection of 123M unlabeled web face images used to augment the labeled datasets for larger-scale clustering evaluation. Example face images from each dataset are shown in Figure 4.

- **LFW** [12]: LFW 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 images for which an automatically detectable face was present.
- **YTF** [13]: Similar in spirit to LFW, the YouTube Faces (YTF) dataset consists of videos of celebrities and public figures harvested from the Internet. The dataset contains $1,595$ subjects (which are a subset of the subjects in LFW), in $3,425$ videos, consisting of a total of $621,126$ individual frames. Labels are provided for the subject of interest for every frame of video where a face could be detected. In our experiments we use the pre-cropped frames, to avoid confusion between the primary subject in each video, and any unlabeled individuals that may be in a given frame.
- **Webfaces**: To evaluate our clustering method on larger scale datasets, a cooperating research group used a crawler to automatically download a total of $123,654,141$ web

Fig. 4: Example face images from the a) LFW, b) Youtube Faces, c) Webfaces, and d) CASIA-webface datasets.

images. Similar to LFW, these images were filtered to contain faces detectable by an automatic face detector, in particular the DLIB face detector.[5]

- **CASIA-webface** [25]: The CASIA-webface dataset contains $494,414$ images of $10,575$ subjects (mostly celebrities); however, we are unable to localize faces in some of the images, and so use a subset of $404,992$ face images of $10,533$ subjects to train our network [11].

5 Experiments

In this section, we will present our overall evaluation of large-scale face clustering, in several steps. First, we will evaluate various clustering algorithms on a small dataset (the entire LFW dataset), evaluate the nearest neighbor approximation used, then carry out large-scale face clustering experiments (involving up to a 123 million face dataset), and finally present some preliminary work on video clustering.

5.1 Clustering Algorithm Evaluation

Before investigating performance on large-scale datasets, we will attempt to cluster the entire LFW dataset by identity. One issue is that the distribution of images per subject is quite imbalanced in LFW (indeed, the majority of subjects have only a single image, accounting for approximately a third of all images in the dataset). Although we could conceivably construct a subset of LFW with more "balanced" clusters, in practice for the application domains of large-scale clustering (analyzing social media imagery, and forensic applications), there is no basis to assume that the number of images per subject is well balanced. In the absence of prior knowledge about the expected distribution of images per subject in practical applications, we cluster the entire LFW dataset.

As a baseline, we will consider k-means clustering, since (i) it is perhaps the most well-known clustering algorithm, (ii) has only a few parameters for tuning, (iii) is one of the most efficient, and (iv) large-scale clustering methods are often approximations of k-means clustering with improved scalability. We use the MATLAB r2015a implementation of the k-means algorithm, with the euclidean distance metric. We additionally use spectral clustering [34], which approaches the problem from a graph theory perspective, as a baseline. We employ spectral clustering on two different graph structures: we induce a graph structure in the adjacency matrix by keeping the top 200 neighbors non-zero using Euclidean distance, and also use the effective graph structure used by our proposed algorithm (using binarized rank-order distance, only considering images which are both in each other's top-200 nearest neighbors as computed by the randomized k-d tree algorithm). We use a MATLAB implementation of spectral clustering[6]. We also applied simple single-link hierarchical clustering based on Euclidean distance (using the MATLAB implementation of hierarchical clustering). Additionally, we use our implementation of the original rank-order clustering algorithm as a baseline.

TABLE 2: Clustering results on the complete LFW dataset. Times are given as HH:MM:SS, measured using 20 cores of an Intel Xeon CPU clocked at 2.5 GHz. The proposed algorithm (last row) has the highest clustering accuracy (F-measure) and the shortest run-time.

Clustering Algorithm	# Clusters	F-measure	Run-Time
k-Means	100	0.36	00:00:16
k-Means	6,508	0.07	04:58:49
Spectral (Euclidean)	200	0.20	00:11:18
Spectral (Approx. ROD)	200	0.43	00:14:04
Hierarchical	1,000	0.005	00:00:25
Rank-Order	7,059	0.65	00:00:33
Approx. Neighborhood Rank-order	6,440	0.83	00:00:09
Approx. Rank-Order (proposed)	6,508	0.87	00:00:08

For k-means, spectral clustering, and hierarchical clustering, the algorithm is parameterized on a fixed number of clusters, while for rank-order clustering the number of clusters found depends on the distance threshold parameter. In practical applications we cannot assume that the true number of clusters will be known a priori, therefore in the absence of a robust procedure for determining the true number of clusters we simply evaluate all algorithms at several effective values of C, and report the best results attained in Table 2.

For k-means and spectral clustering, clustering performance (in terms of F-measure) with C close to the true number of identities is quite poor. This is expected, since these algorithms are not able to handle highly unbalanced data well. For example, if there is one large ground truth cluster, centered in a dense region of the feature space, and outlying small, less dense, clusters the large cluster will be partially split with the outlying clusters

5. In some cases, the detected faces are in fact false positive detections (e.g. non-human faces (such as cartoons), or non-face objects). We estimate approximately 2% of the total detections may be false positives, based on a manual examination of a random sample of $10,000$ detections from the full dataset. We did not delete the identified non-human faces from the dataset.

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

Fig. 5: Examples of "pure" (single individual) clusters (a, b), and "impure" (multiple individuals) clusters (c,d) generated by the proposed Approximate Rank-Order clustering on the entire LFW dataset. Faces not belonging to the majority identity in each cluster are outlined in red.

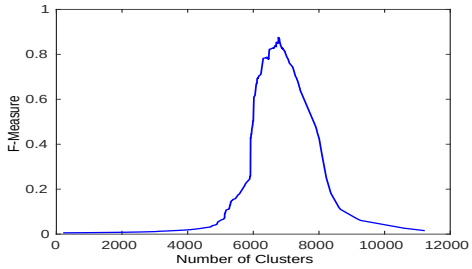

Fig. 6: F-measures attained by approximate rank-order clustering at different effective numbers of clusters (generated by varying the distance threshold).

(even if initial cluster centers are placed well). For this reason, the optimal value of C in terms of F-measure is relatively low for k-means and spectral. For rank-order clustering, the distance threshold which leads to the best overall F-measure results in a number of clusters close to the true number of identities, and the overall F-measure is significantly higher than the spectral and k-means results. The local neighborhood used in approximate rank-order clustering (only considering shared neighbors within the top-200 to be connected) can handle the case where there is significant density variation. For example, points within a large dense ground-truth cluster may be well connected to each-other, and not connected to outlying clusters at all. Figure 6 shows the F-measures attained at different effective numbers of clusters by the proposed approximate rank-order algorithm.

We attained the best overall F-measure on this experiment with the two rank-order clustering variants, with the proposed method outperforming the original algorithm. Use of the randomized k-d tree nearest neighbor method (and a neighborhood based on shared nearest neighbors) is responsible for the majority of our improvement over the baseline rank-order method, with the binarized distance function contributing an additional 4% improvement to F-measure. Spectral clustering leveraging our graph structure performed well relative to the Euclidean distance based graph, attaining somewhat higher performance than k-means. In our experiments, directly applied hierarchical clustering formed large impure clusters early on in the process, leading to poor results overall.

TABLE 3: Clustering Results on the LFW dataset, with approximate rank-order clustering, and LFW with additional 1 million web-downloaded face images. Times measured using 20 cores of an Intel Xeon CPU clocked at 2.5 GHz.

Nearest Neighbor Algorithm	Dataset	F-measure	Run-Time
Brute-Force	LFW	0.72	00:00:12
Chen et al. [23]	LFW	0.69	00:26:36
Randomized k-d Tree [24]	LFW	0.87	00:00:08
Brute-Force	LFW + 1M	0.49	14:18:24
Chen et al.	LFW + 1M	0.41	01:06:58
Randomized k-d Tree	LFW + 1M	0.79	00:07:20

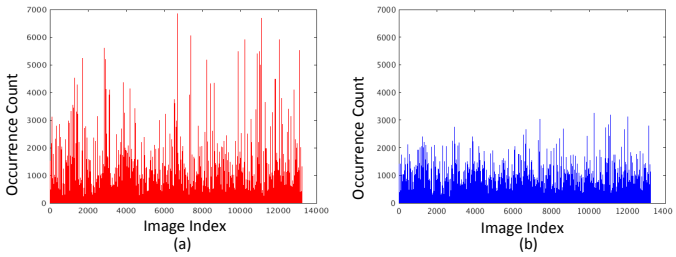

Fig. 7: Numbers of times each face in the LFW database appeared in any other face's top-200 nearest neighbor list for a) the exact nearest neighbors, and b) the nearest neighbors computed via randomized k-d tree approximation.

In terms of runtime, per Table 2, even for just $13,233$ face images in LFW spectral clustering takes noticeably large compute time, while the proposed rank-order clustering is substantially faster. Some example clusters are shown in Figure 5; 5(a) and (b) show pure clusters, while 5(c) and (d) show example impure clusters in terms of subject identity. In cluster 5(c), 3 images of different individuals, all with similar demographics, were grouped in with the majority identity (Walter Mondale); while in cluster 5(d), 2 images of 1 additional individual with similar demographics, and face pose were grouped with the majority identity (Michael Douglas).

5.2 Approximation Performance

We evaluate the performance of our k-NN approximation method in terms of clustering accuracy, and run-time. We consider two approximation methods for computing the full k-NN graph, and compare their performance to the brute force approach of performing all pairwise comparisons. Results are shown in Table 3 for these three nearest neighbor calculation methods on the full LFW dataset, and the LFW dataset augmented with an additional 1 million unlabeled images from the Webfaces dataset. In practice, the Randomized k-d Tree method [24] achieves the best run-time of the three methods and the best clustering accuracy as well.

This is a surprising result, since an approximation method would generally be expected to give less accurate results than the process it is approximating; however, since our objective is to perform clustering based on the nearest neighbor lists, rather than simply find the exact k nearest neighbors for each item, this counter-intuitive result can be explained as follows. Figure 7 plots the number of times each face in the LFW dataset occurs in the top 200 nearest neighbor list of every face in the dataset. For the exact nearest neighbors, there are a number of face images which occur very frequently in the nearest neighbor lists (up to

TABLE 4: Clustering results using Approximate Rank-Order clustering on the LFW dataset with increasing amounts of augmented data, and different search size strategies for the approximate nearest neighbor calculations. Times measured using 5 cores for LFW+5M dataset experiments, and a single core used for the smaller experiments, on an Intel Xeon CPU clocked at 2.5 GHz.

Dataset	Search Size	F-measure	Run-Time
Just LFW	2,000	0.87	00:00:19
LFW + 1M	2,000	0.79	01:03:25
LFW + 5M	10,000 (linear increase)	0.67	06:28:42
LFW + 5M	4,000 (logarithmic increase)	0.33	02:51:13
LFW + 5M	2,000 (fixed)	0.13	01:52:32

TABLE 5: Large-scale clustering results using the proposed Approximate Rank-Order clustering, with randomized k-d tree nearest neighbor approximation. Times measured using the specified number of cores of Intel Xeon CPUs clocked at 2.5 GHz. # Clusters is the resulting number of clusters, excluding single-image clusters.

Dataset	F-measure	# Clusters	# Cores	Run-Time
LFW	0.87	1,463	1	00:00:19
LFW + 1M	0.79	94,740	1	01:03:25
LFW + 5M	0.67	445,880	5	06:28:42
LFW + 10M	0.56	933,278	10	12:11:33
LFW + 30M	0.42	2,800,202	30	30:44:58
LFW + 123M	0.27	10,619,853	123	289:04:53
LFW + 121M	0.32	10,281,612	123	245:12:54

over half of all nearest neighbor lists), while for the approximate nearest neighbors these faces occur less frequently. The action of the k-d Tree algorithm is to pick a highest variance dimension at each node, and splits the feature vectors at that dimension. After the tree is built, splitting on the selected dimensions at each node partitions the space into hyperrectangles, and an approximate search is accomplished by restricting the total number of these nodes that we visit in a given search. As a result of this, there are cases where closer neighbors exist in the feature space, but due to the truncated search we never visit the node containing them, and this sampling effect has reduced the total number of times the most frequently occurring nearest neighbors (under Euclidean distance) are found. From the perspective of clustering based on the nearest neighbor lists, the lists computed from the randomized k-d tree approximation actually form more discriminative features, since certain faces are not present in very large fractions of the nearest neighbor lists, as is the case with the exact nearest neighbors.

Generally, the randomized k-d tree algorithm has $O(n\ log\ n)$ expected run-time for tree construction, and performing n searches. In practice, the FLANN implementation of the algorithm is parametrized with the number of randomized trees constructed, as well as the total number of nodes available to visit per search. If fixed parameters are used, the total runtime is indeed $O(n\ log\ n)$; however, if either the number of indices built or search size is increased with larger dataset size, the effective runtime of the algorithm will increase. In practice, we construct 4 trees per index (and have found little impact from using slightly higher or lower values), but the number of nodes visited per search must be selected with care. One primary question is to determine if a fixed number of node visits per search is feasible for larger datasets, or if the number of nodes visited per search should increase with dataset size. Table 4 presents results for clustering based on the LFW dataset, the LFW + 1M dataset, and the LFW + 5M dataset, using different strategies for selecting the number of nodes visited per search on the LFW + 5M dataset. In practice, using the same number of nodes visited per search on the LFW + 5M dataset as was used on the LFW + 1M dataset leads to a drastic reduction in clustering accuracy on the larger dataset. In fact, even a logarithmic increase in search size leads to significant accuracy loss, relative to a linear increase in search size. In the following large-scale experiments, we therefore increase the search size linearly with dataset size. In practice, this means the run-time of the approximation algorithm cannot be considered to be $O(n\ log\ n)$, since we increase the cost of each search linearly with the dataset size n, giving a full $O(n^2)$ cost for performing n nearest neighbor searches.

By using the randomized k-d tree algorithm for approximate nearest neighbor computation, with our updated clustering algorithm we get improved runtime in the clustering step, and also better clustering accuracy (compared to the baseline algorithm). Although we still have an $O(n^2)$ run-time for the nearest neighbor computation step, there is still a significant reduction in run-time, an improvement by a factor of 120 for the LFW+1M image dataset over brute-force computation.

5.3 Large-Scale Face Clustering

In this section, we will consider clustering truly large-scale face datasets, up to 123 million face images. As discussed, we will use the randomized k-d tree nearest neighbor approximation method to reduce the total cost of computing nearest neighbors for these datasets; however, when considering very large scale datasets, additional problems arise. Considering the total size of the dataset, 123 million 320-dimensional feature vectors, with each dimension represented by one float takes up approximately 157 gigabytes of space, without considering any supporting data structures. This amount of data is difficult to fit on a single machine, considering that a tree structure must also be loaded in memory, and additionally since the approximation method we are using incurs a full $O(n^2)$ cost in time, computing nearest neighbors for this dataset becomes infeasible on a single machine.

Fortunately, a distributed memory variation of the randomized k-d tree algorithm is available as part of the FLANN library [30]. The strategy employed is to split the dataset into disjoint subsets, assign one subset to every discrete machine used, and construct separate k-d tree indices for each disjoint chunk. During nearest neighbor computation, we find a separate set of nearest neighbor candidates from each chunk, and merge the results to get a final set of nearest neighbors for the search. In practice, this simple strategy works well. In the following experiments the initial dataset is partitioned into 1 million image chunks, and each chunk is distributed to a separate machine for index construction. Since our datasets are a small labeled subset (LFW), in a larger unlabeled background set, we randomize the order of the LFW images, and assign a portion of the labeled images to each of the discrete chunks of data, to avoid any bias due to constructing one of the sub-indices with e.g. the entire LFW dataset as part of it.

Results for progressively larger datasets (constructed by adding larger and larger sets of unlabeled background data to LFW) are presented in Table 5. Due to our strategy of allocating 1 million images per core, we linearly increase the number of cores with dataset size, resulting in an overall $O(n)$ increase

Fig. 8: Example image pairs flagged at 5 std. dev. below the mean genuine score on LFW. (a) a genuine LFW match, (b), an impostor LFW match, (c) a near duplicate in the background dataset, and (d) poor quality images in the background dataset.

in runtime when moving to larger datasets. Computing nearest neighbors for the largest dataset considered (123 million images) took approximately 2 weeks of real-time using 123 nodes in the MSU High-Performance Computing Center. While the observed run time increases is approximately linear, the clustering accuracy progressively decays when considering larger and larger datasets. This is as expected, considering a larger dataset means a larger chance of finding impostors for each individual image as nearest neighbors. Even so, on the 123 million image dataset, we still attain 0.27 F-measure on the labeled subset, which is considerably better than a random result (which is close to zero, since $13,233$ images can easily be grouped into 123M images without keeping any of the same identity face images together).

5.4 Deduplication

The background set used in these experiments consists of 123 million unlabeled images harvested from the Internet, and some fraction of these images are near-duplicates of each-other. For example, there may be multiple copies of the same image, which were uploaded at slightly different resolutions, or with slightly different encodings, or digitally modified. We attempted to identify these images by examining outlier match distances in the set of nearest neighbors we computed for the full 123 million image dataset. To define outliers, we considered the distribution of genuine match distances produced by our face representation on the LFW dataset, computed the mean and standard deviation of that distribution, and flagged comparisons more than 5 standard deviations below the genuine mean as outliers.

On the LFW dataset, this threshold flags 3 genuine matches (all very similar looking images of George W. Bush, one of which is shown in Figure 8-a), and a number of impostor matches. Examining these high scoring impostor comparisons typically indicated cases like Figure 8-b where one of the involved images was handled poorly by our algorithm (e.g. extreme profile images are not typically present in our training data, and are handled poorly by our representation). Applying this threshold to the full 123 million image background set flagged a total of $33,359,232$ of our computed nearest neighbor pairs, but these matches involved a total of just $3,405,294$ unique image files. Manually examining these matches revealed some near-duplicate images (as in Figure 8-c), but also revealed a number of cases similar to Figure 8-d, where one or both of the pairs of images are of low visual quality, but the pair are not near-duplicates.

For simplicity, we used a transitive merge on the flagged match pairs resulting in $1,188,326$ groups, randomly selected one image per group to retain, and deleted the rest (removing a total of $2,216,968$ images). We evaluated LFW images removed in this process as recall errors, that is we considered them to still be part of their relevant classes when computing recall, but did not place them in any cluster (this reduces the F-measure of our method on just the LFW images from 0.87 to 0.86). We then performed clustering again on the remaining 121 million images, and as shown in the last two rows of Table 5, deletion of approximately 1.8% of the background dataset in this manner improved F-measure from 0.27 to 0.32.

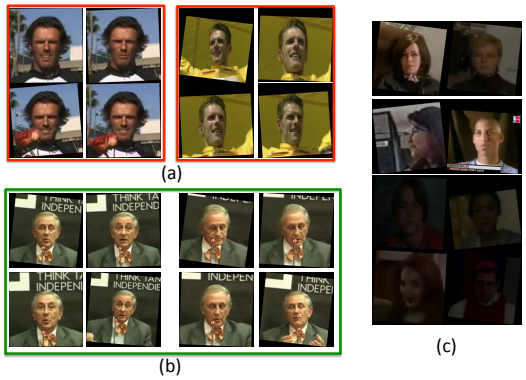

Fig. 9: Example images from clusters generated from the YTF dataset. a) two clusters, each containing frames from one video of the same subject, b) a cluster containing frames from two videos of the same subject, where the background for the video is apparently identical, c) a subset of a cluster containing 28 different identities; many of these images are poorly lit.

5.5 Video Frame Clustering

We also consider the problem of clustering video frames, using the Youtube Faces (YTF) dataset. Similar to our treatment of LFW, we cluster all faces in YTF, and evaluate the results in terms of their consistency with arranging the individual frames by identity. The results are summarized in Table 6. The overall F-measure appears reasonably consistent with our LFW results, at 0.74 for $621,126$ total frames of video (compared to 0.79 F-measure for clustering the LFW + 1M dataset); a lower accuracy on the YTF dataset is expected because of its generally lower image quality. However, closer analysis of the results reveals some confounding factors.

Unlike LFW clustering results, where precision and recall are relatively close for the optimal F-measure values, our clustering results on YTF have very high precision, and relatively lower recall. Effectively, we are getting more clusters than the number of identities, but the clusters are relatively pure. Further analyzing the recall, we find that although the overall value is 0.589, the fraction of same-video pairs grouped together is much higher than the fraction of cross-video pairs grouped together. This indicates that we are successfully grouping frames into videos, but having relatively little success grouping identities across videos. Some example clusters are shown in Figure 9. In most cases, clusters roughly correspond to single videos, in a few cases, e.g. 9(b), frames from different videos of the same individual are correctly grouped together, and in a small subset of clusters (e.g. 9(c)), multiple identities are grouped in the same cluster.

To investigate the impact of the high number of similar frames in each video on our algorithm, we also performed clustering after

TABLE 6: YTF clustering results using the proposed Approximate Rank-Order clustering for clustering all frames in the dataset, and for randomly sampling 3 frames per video. Time is given as HH:MM:SS, measured using 20 cores of an Intel Xeon CPU clocked at 2.5 GHz.

Performance Measure	All Frames	Sampled 3 Frames
F-Measure	0.71	0.53
Precision	0.79	0.82
Recall	0.67	0.39
Within-Video Recall	0.91	0.89
Cross-Video Recall	0.56	0.20
Nearest Neighbor Computation Time	00:04:10	00:00:07

sampling a random set of 3 frames per video, and again performed clustering. These results are also shown in Table 6, and exhibit similar overall trends to the full frame clustering experiment. In particular, precision and recall are again unbalanced, but overall results are worse since within-video pairs form a much smaller part of the overall evaluation, and cross-video recall also decreases. This indicates that the presence of many near-duplicate frames is not primarily responsible for reducing the performance of our algorithm on the task, rather it seems that the main difficulty is in reliably matching frames across videos. The difficulty in matching across video is also reflected in the performance of our face representation on the standard YTF evaluation protocol, where our True Accept Rate at 1% False Accept Rate is 61%, compared to 92% on the standard LFW protocol.

5.6 Per-Cluster Quality: Internal Measures

We are interested in identifying a good subset of clusters from a large group of clusters, as a means of aiding manual exploration of large datasets. To evaluate the effectiveness of different measures experimentally, we need methods for evaluating the effectiveness of the internal measures. As a first approach, we consider the correlation between the internal measures and an external measure, the pairwise precision computed individually for each cluster. Correlation between the various internal measures considered, and precision are show in Table 7. In practice, the graphical measures (coverage, modularization quality) do not perform particularly well, while the simpler measures based on edge weights perform better. In particular, the best correlation is observed for the "average inter-cluster edge weight", and this can be further improved by subtracting the average inter and intra cluster edge weights. This is reasonable, since we are effectively combining a compactness measure (average intra cluster edge weight), and a separability measure (average inter-cluster edge weight). Even so, the best correlation achieved is only 0.42. This correlation can be improved by excluding size 2 clusters from consideration (so only examining clusters with 3 or more members), which improves correlation to 0.46.

While a correlation of 0.46 is not very high, for our application there is no particular need for the relationship between the internal and external measures to be linear. Figure 10 plots the external measure (Precision) vs. the best performing internal measure $\{$(avg. intra-cluster edge weight) - (avg. inter-cluster edge weight)$\}$. One notable feature on the left side of the plot is a set of clusters with exactly zero precision, that still score relatively highly on the internal measure. Closer examination reveals that all of these high scoring zero-precision clusters are of size two

TABLE 7: Correlation of internal cluster quality measures with precision, for the LFW dataset. Average Prec.@100 is the unweighted average of per-cluster pairwise precision for the top-100 scoring clusters for each metric, Fraction Pure @ 100 is the fraction of the top-100 clusters which contain only a single identity.

Internal Measure	Correlation	Avg. Prec.@100	Pure@100
Inter-MQ	0.128	0.748	0.70
Intra-MQ	0.117	0.837	0.72
MQ (Combined)	0.120	0.829	0.80
Coverage	0.117	0.748	0.70
Max Intra-Cluster Edge Distance	0.022	0.860	0.86
Total Intra-Cluster Edge Distance	0.030	0.850	0.85
Avg. Intra-Cluster Edge Distance (1)	0.080	0.853	0.85
Min Inter-Cluster Edge Distance	0.205	0.893	0.87
Total Inter-Cluster Edge Distance	0.125	0.852	0.56
Avg. Inter-Cluster Edge Distance (2)	0.325	0.880	0.85
(1) - (2)	0.427	0.908	0.89
(1) - (2), cluster size ≥ 3	0.460	0.979	0.95

(so they consist of relatively isolated faces of different people that happen to score highly), which explains the improvement in correlation (from 0.42 to 0.46) when restricting consideration to size 3 or larger clusters. Another interesting feature of Fig. 10 is that the points on the plot almost form a triangle (with a variety of precision values for low-scoring clusters, but mostly just high precision values for high-scoring clusters), so although the relationship between the external and internal measures is not linear, it is still possible to select a subset of high precision clusters by taking a high threshold on the internal measure.

5.6.1 *Ranking Evaluation*

As an alternative to considering correlation, we can use the internal measure to rank all the clusters, and compute the unweighted average of per-cluster precision values for the top C clusters (ranked by the internal measure), inspired by analysis typically done in retrieval problems. Table 7 shows the average precision of the various internal measures considered for the top-100 clusters in the full LFW dataset, and additionally shows the fraction of clusters in the top-100 for each measure that contain a single identity (are "pure"). The blue line in Figure 11 plots the average precision of the top C clusters, with C cut off at each possible rank in the sorted list of clusters, for the best performing internal measure, while the red line plots the same information when the clusters are ordered by their actual precision (and thus reflects the best possible performance on this task). The internal measure is effective in selecting high precision clusters (relative to the average precision of all the clusters) on the LFW dataset. In fact, the first several clusters ranked by the internal measure have a pairwise precision of 1. Figure 12 extends this concept to the augmented datasets (in this case, only clusters containing some labeled data from LFW are ranked, and precision is computed omitting unlabeled clusters as in our previous evaluation).

Initially, the internal measure is still effective; however, for very large datasets (LFW+ 30M and above), ranking clusters according to the internal measure, as expected, becomes less effective. Some example top ranking clusters are shown in

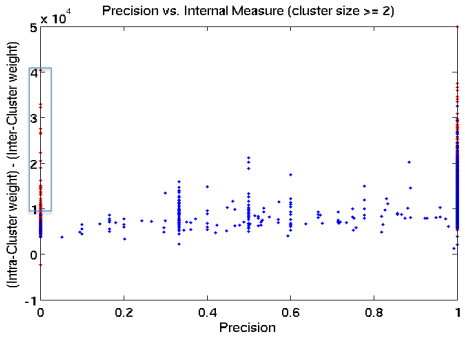

Fig. 10: Pairwise precision vs. the proposed internal quality measure, for all clusters generated by the proposed Approximate Rank-Order clustering algorithm on the full LFW dataset. Points in blue are clusters of size 3 or larger, points in red are of size 2. The highlighted set of points on the left edge of the figure are all of size 2, with zero pairwise precision. Since we can't reliably distinguish between good and bad 2-item clusters, we discard them from consideration.

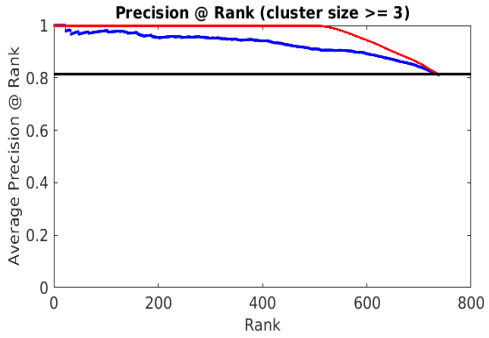

Fig. 11: The blue line displays the average pairwise precision for lists of clusters ordered by the proposed internal cluster quality measure, terminated at each possible rank, while the red line displays the average pairwise precision for a list ordered by the true precision of each cluster (representing the best possible performance on this task). The horizontal black line indicates the unweighted average precision of all clusters considered. Clusters are generated by the proposed Approximate Rank-Order clustering algorithm from the full LFW dataset.

Figure 13. The top-5 clusters for the LFW dataset are all single identity, relatively small clusters, indicating that the quality measure works as expected. For larger datasets (LFW+10M, LFW+123M), we show both the top-5 clusters ranked purely in terms of the quality measure, (b) and (d), as well as the top-5 results containing any labeled data, (c) and (e) (since we use the labeled subset in our numerical evaluations). The top clusters in absolute ranking typically involve near-duplicate images (e.g. similar images uploaded in different locations, with minor differences due to cropping, resolution, or color correction differences), and often cartoon faces (which were detected by face detectors) in addition to actual photographs.

The top clusters involving LFW images show that there are in fact a number of images of the LFW subjects in the unlabeled dataset–this indicates that our performance evaluation is overly conservative, since we consider grouping LFW and unlabeled data together to be incorrect (due to lack of label information). Although the results for the LFW+10M dataset appear reasonable (in the sense that multiple images of the same identity are being grouped together, that are not just slight alterations of the same original image), for the LFW+123M dataset we begin to see a large number of near duplicate images, e.g. the clusters ranked 1 through 4 on the list of clusters with LFW images have a single LFW image, and multiple near duplicate images that happened to be in the background set.

Considering the results for the de-duplicated 121M image dataset, there are still a number of near-duplicate images that were not caught by the deduplication process. Manual examination of higher ranked clusters (up to rank 150), revealed that after the first 20 or so clusters, actual near duplicate images became rare. But even with that all clusters involving LFW data until rank 116 were actually single subject clusters, consisting of some LFW images, and images from the background set of the same subject. Deduplication of unconstrained face images is indeed a challenging problem.

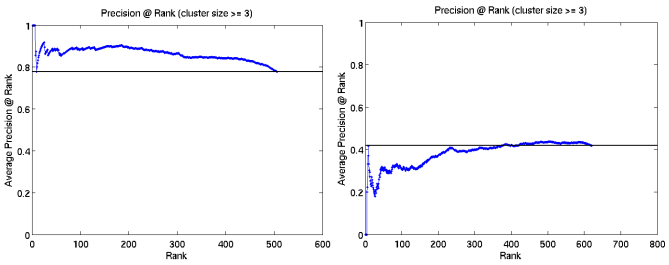

Fig. 12: Average pairwise precision at rank, ordered by the proposed internal cluster quality measure for augmented datasets. Clusters are generated by the proposed Approximate Rank-Order clustering algorithm.

6 Conclusions

We have shown the feasibility of clustering a large collection of unlabeled face images (up to 123M) into an unspecified number of identities (on the order of millions). This problem is of practical interest as a first step in organizing a large collection of unlabeled face images prior to human examination due to the high volume of face images uploaded to social media, and potentially encountered in forensic investigations. There are computational challenges in processing datasets with tens of millions of faces (which we address via approximation methods, and parallelization). Even if the computational challenges are met, producing meaningful clusters on data of this scale is very difficult. In terms of clustering accuracy, we achieved 0.27 pairwise F-measure on the largest dataset considered (123M unlabeled faces + 13,233 labeled images from LFW), which indicates that at least some of the clusters produced by our algorithm correspond well to true identities in LFW. To identify these high quality clusters, we developed an internal per-cluster quality measure, that does not involve external identity labels, to rank the clusters by quality for manual examination. Experimental results showed that this measure was extremely effective for smaller datasets, but for the larger datasets considered (LFW + 123M unlabeled faces), performance apparently falls (although this point is complicated by the degree of overlap between the full background dataset and LFW). Still, some good quality (compact and isolated) face clusters can be identified.

In terms of future work, while the underlying face representation we employ works reasonably well for unconstrained face images, it could still be improved in a number of ways (e.g. using larger training sets, or improving the deep model architecture). While we were able to apply our clustering algorithm to datasets up to 123 million face images, we need to improve the clustering method (e.g. by incorporating more accurate nearest neighbor methods) to obtain better clustering accuracy. Other areas for improvement include the automatic selection of the number of clusters in a fully unlabeled dataset, as well as improving our per-cluster quality evaluation methods, and utilizing pair-wise constraints (must-link and cannot-link) to improve clustering accuracy.

ACKNOWLEDGEMENTS

We would like to thank the Noblis corporation for their assistance in acquiring the unlabeled background images used in this work.

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, pp. 91–94, 2013. 1

[2] B. S. Swann, "FBI video analytics priority initiative," in *17th Annual Conference & Exhibition on the Practical Application of Biometrics*, 2014. 1

[3] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *Proc. CVPR*. IEEE, 2003. 2

[4] M. Zha, Y. Teo, S. Liu, T. Chua, and R. Jain, "Automatic person annotation of family photo album," in *Image and Video Retrieval*. Springer, 2006, pp. 163–172. 2

[5] J. Cui, F. Wen, R. Xiao, Y. Tian, and X. Tang, "Easyalbum: an interactive photo annotation system based on face clustering and re-ranking," in *Proc. of the SIGCHI conference on Human factors in computing systems*. ACM, 2007, pp. 367–376. 2

[6] Y. Tian, W. Liu, R. Xiao, F. Wen, and X. Tang, "A face annotation framework with partial clustering and interactive labeling," in *Proc. CVPR*. IEEE, 2007. 2

[7] C. Zhu, F. Wen, and J. Sun, "A rank-order distance based clustering algorithm for face tagging," in *Proc. CVPR*. IEEE, 2011, pp. 481–488. 1, 2, 4, 5

[8] Z. Cao, Q. Yin, X. Tang, and J. Sun, "Face recognition with learning-based descriptor," in *Proc. CVPR*. IEEE, 2010, pp. 2707–2714. 2

[9] R. Vidal and P. Favaro, "Low rank subspace clustering (lrsc)," *Pattern Recognition Letters*, vol. 43, pp. 47–61, 2014. 2

[10] C. Otto, B. Klare, and A. Jain, "An efficient approach for clustering face images," in *Proc. ICB*. IEEE, 2015. 2, 3, 4

[11] D. Wang, C. Otto, and A. K. Jain, "Face search at scale: 80 million gallery," *arXiv preprint arXiv:1507.07242*, 2015. 1, 2, 4, 7

[12] 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. 1, 6

[13] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. CVPR*. IEEE, 2011, pp. 529–534. 1, 6

[14] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010. 2, 4

[15] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-NN graph construction for visual descriptors," in *Proc. CVPR*. IEEE, 2012, pp. 1106–1113. 2

[16] B. Bhattarai, G. Sharma, F. Jurie, and P. Pérez, "Some faces are more equal than others: Hierarchical organization for accurate and efficient large-scale identity-based face retrieval," in *ECCV Workshops*, 2014, pp. 160–172. 2

[17] M. Tapaswi, O. M. Parkhi, E. Rahtu, E. Sommerlade, R. Stiefelhagen, and A. Zisserman, "Total cluster: A person agnostic clustering method for broadcast videos," in *Proc. of the Indian Conference on Computer Vision Graphics and Image Processing*. ACM, 2014, p. 7. 2

[18] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, 2015. 2

[19] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *Proc. CVPR*, June 2016. 2

[20] T. Liu, C. Rosenberg, and H. A. Rowley, "Clustering billions of images with large scale nearest neighbor search," in *Proc. WACV*. IEEE, 2007, pp. 28–28. 2

[21] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Boundev, and R. Fergus, "Web scale photo hash clustering on a single machine," in *Proc. CVPR*. IEEE, 2015, pp. 19–27. 3

[22] J. J. Foo, J. Zobel, and R. Sinha, "Clustering near-duplicate images in large collections," in *Proc. of the International Workshop on Multimedia Information Retrieval*. ACM, 2007, pp. 21–30. 3

[23] J. Chen, H. Fang, and Y. Saad, "Fast approximate k-NN graph construction for high dimensional data via recursive lanczos bisection," *The Journal of Machine Learning Research*, vol. 10, pp. 1989–2012, 2009. 3, 8

[24] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Proc. CVPR*. IEEE, 2008, pp. 1–8. 3, 8

[25] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014. 4, 6, 7

[26] 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 *Proc. CVPR*. IEEE, 2015, pp. 1931–1939. 4

[27] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proc. CVPR*. IEEE, 2014, pp. 1867–1874. 4

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 4

[29] K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighbourhood," *Pattern Recognition*, vol. 10, no. 2, pp. 105–112, 1978. 4

[30] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. on PAMI*, vol. 36, 2014. 5, 9

[31] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice Hall, 1988. 6

[32] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," *Algorithms-ESA 2003*, pp. 568–579, 2003. 6

[33] S. Mancoridis, B. S. Mitchell, C. Rorres, Y.-F. Chen, and E. R. Gansner, "Using automatic clustering to produce high-level system organizations of source code." in *IWPC*, vol. 98. Citeseer, 1998, pp. 45–52. 6

[34] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007. 7

Charles Otto received his B.S. degree in the Department of Computer Science and Engineering at Michigan State University in 2008. He was a research engineer at IBM during 2006-2011. Since 2012, he has been 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.

Dayong Wang received his bachelor degree from Tsinghua University in 2008 and his Ph.D. degree from Nanyang Technological University, Singapore, 2014. He is currently a Postdoctoral Researcher at Michigan State University, USA. His research interests are statistical machine learning, pattern recognition, and multimedia information retrieval.

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 and is a member of the National Academy of Engineering.

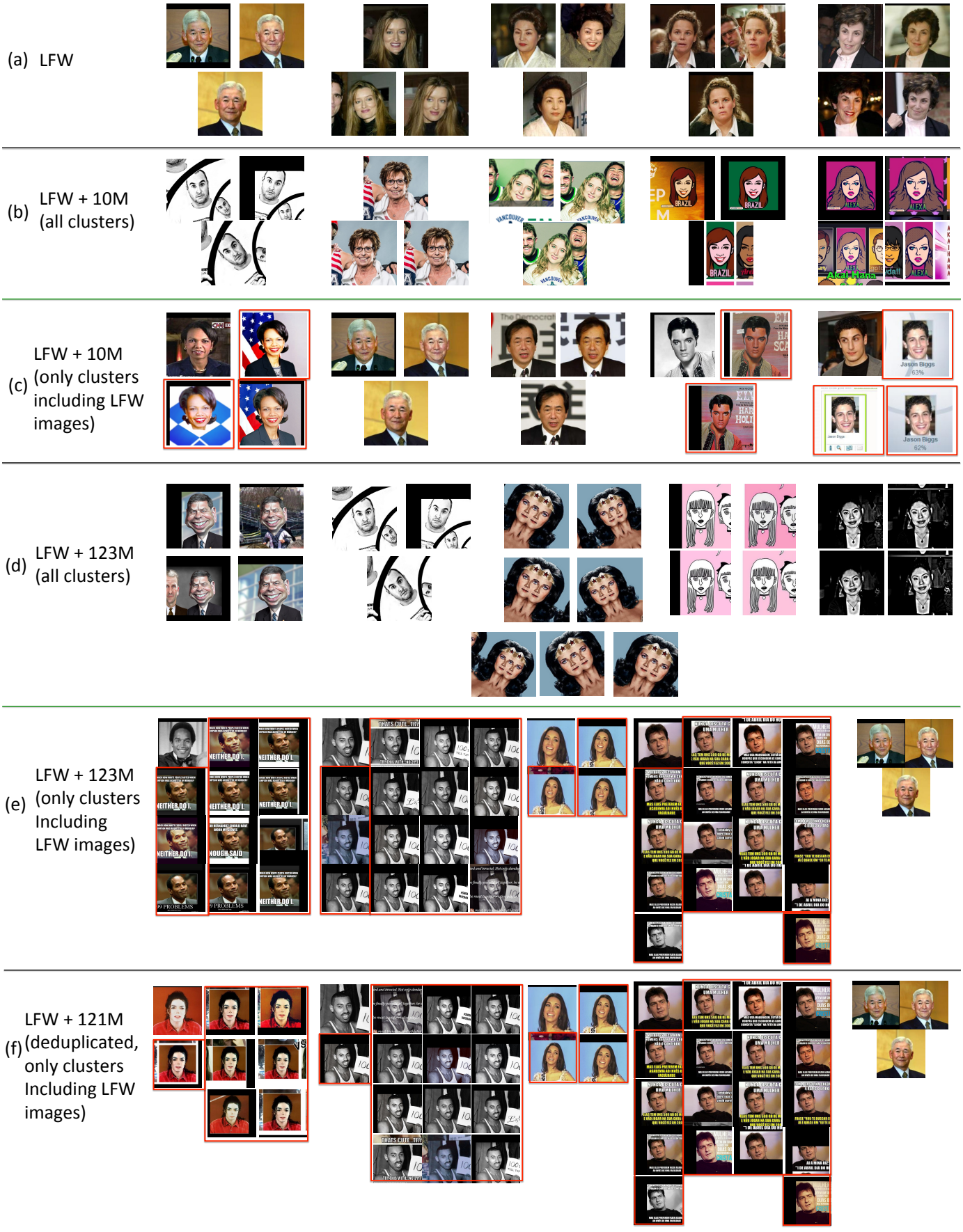

Fig. 13: Top-5 ranked clusters for the LFW, LFW+10M, LFW+123M, and deduplicated datasets. For the LFW+10M, and LFW+123M datasets, both the absolute top-5 ranking clusters in terms of the proposed quality measure, and the top-5 ranking clusters out of those clusters containing at least some LFW images are shown. Unlabeled images grouped in with the LFW images are outlined in red.