Clustering based Contrastive Learning for Improving Face Representations

Vivek Sharma[1,2], Makarand Tapaswi[3], M. Saquib Sarfraz[1,4] and Rainer Stiefelhagen[1]
[1]Karlsruhe Institute of Technology, [2]Massachusetts Institute of Technology, [3]Inria Paris, and [4]Daimler TSS

Abstract— **A good clustering algorithm can discover natural groupings in data. These groupings, if used wisely, provide a form of weak supervision for learning representations. In this work, we present Clustering-based Contrastive Learning (*CCL*), a new clustering-based representation learning approach that uses labels obtained from clustering along with video constraints to learn discriminative face features. We demonstrate our method on the challenging task of learning representations for video face clustering. Through several ablation studies, we analyze the impact of creating pair-wise positive and negative labels from different sources. Experiments on three challenging video face clustering datasets: BBT-0101, BF-0502, and ACCIO show that CCL achieves a new state-of-the-art on all datasets.**

I. Introduction

Learning strong and discriminative representations is important for diverse applications such as face analysis, medical imaging, and several other computer vision and natural language processing (NLP) tasks. While considerable progress has been made using deep neural networks, learning a representation often requires a large-scale dataset with manually curated ground-truth labels. To harness the power of deep networks on smaller datasets and tasks, pre-trained models (*e.g.* ResNet-101 [17] trained on ImageNet, VGG-Face [27] trained on a large number of face images) are often used as feature extractors or fine-tuned for the new task.

We introduce a clustering-based learning method *CCL* to obtain a strong face representation on top of features extracted from a deep CNN. An ideal representation has small distances between samples from the same class, and large inter-class distances in feature space. Different from a fully-supervised setting where ground-truth labels are often provided by humans, we view CCL as a *self-supervised* learning paradigm where ground-truth labels are obtained automatically based on natural groupings of the dataset.

Self-supervised learning methods are receiving increasing attention as collecting large-scale datasets is extremely expensive. In NLP, word and sentence representations (*e.g.* word2vec [23], BERT [7]) are often learned by modeling the structure of the sentence. In vision, there are efforts to learn image representations by leveraging spatial context [9], ordering frames in a short video clip [11], [24], or tracking image regions [41]. There are also efforts to learn multimodal video representations by ordering video clips [35].

Among recent works in the *self-supervised* learning paradigm, [4], [13], [19], [46], [49] propose to learn image representations (often with an auto-encoder) using a clustering algorithm as objective, or as a means of providing pseudo-labels for training a deep model. One challenge with the above methods is that the models need to know the number of clusters (usually the number of categories) a priori. Our paper is related to the above as we utilize clustering algorithms to discover the structure of the data, however, we do not assume knowledge of number of clusters. In fact, our method is based on working with a clustering setup that yields a large number of clusters with high purity and few samples per cluster.

Our main contribution, CCL, is a method for refining feature representations obtained using deep CNNs by discovering dataset clusters with high purity and typically few samples per cluster, and using these cluster labels as potentially noisy supervision. In particular, we obtain positive (same class) and negative (different class) pairs of samples using the cluster labels and train a Siamese network. We adopt the recently proposed FINCH clustering algorithm [30] as a backbone since it provides hierarchical partitions with high purity, does not need any hyper-parameters, and has a low computational overhead while being extremely fast.

Towards our task of clustering faces in videos, we demonstrate how cluster labels can be integrated with video constraints to obtain positive (within and across neighboring clusters) and negative pairs (co-occurring faces and samples from farthest clusters). Both the cluster labels and video constraints are used to learn an effective face representation.

Our proposed approach is evaluated on three challenging benchmark video face clustering datasets: Big Bang Theory (BBT), Buffy the Vampire Slayer (BF) and Harry Potter (ACCIO). We empirically observe that deep features can be further refined by using weak cluster labels and video constraints leading to state-of-the-art performance. Importantly, we also discuss why the cluster labels may be more effective than using labels from alternative groupings such as tracking (*e.g.* TSiam [34]).

The remainder of this paper is structured as follows: Section II provides an overview of related work. In Section III, we propose our method for clustering-based representation learning, CCL. Extensive experiments, an ablation study, and comparison to the state-of-the-art are presented in Section IV. Finally, we conclude our paper in Section V.

II. Related Work

This section discusses face clustering and identification in videos. We primarily present methods for face representation learning with/without CNNs, followed by a short overview of FINCH-clustering algorithm and learning from clustering.

Video face clustering methods often follow 2 steps: obtain pairwise constraints typically by analyzing tracks, followed by representation/metric learning approaches and clustering. We present models through a historical perspective, with and without a CNN.

Face representation learning without CNNs. The most common source of constraints is the **temporal information** contained in face tracks. Faces within the same track are assigned a positive label (same character), while faces from temporally overlapping tracks are assigned a negative label (different characters). This approach has been exploited by learning a metric to obtain cast-specific distances [5] (ULDML); iteratively creating a generative clustering model and associating short sequences using a hidden Markov Random Field (HMRF) [43], [44]; or performing clustering with a weighted block-sparse low-rank representation (WB-SLRR) [45]. Additional constraints from **video editing cues** such as shots, threads, and scene boundaries are used in an unsupervised way to merge tracks [39] by learning track and cluster representations with SIFT Fisher vectors [26].

Face representation learning with CNNs. With the popularity of Convolutional Neural Networks (CNNs), there is a growing emphasis on improving face track representations using CNNs. An improved triplet loss that pushes the positive and negative samples apart in addition to the anchor relations is used by [52] to fine-tune a CNN. Zhang *et al.* [53] (JFAC) use a Markov Random Field to discover dynamic constraints iteratively to learn better representations during the clustering process. Inverse reinforcement learning on a ground-truth data is also used to learn a reward function that decides whether to merge a given pair of face features [18]. In a slightly different vein, the problem of face detection and clustering is addressed jointly by [20], where a link-based clustering algorithm based on rank-1 counts verification merges frames based on a learned threshold.

Among recent works, [38] aim to estimate the number of clusters and their assignment and learn an embedding space that creates a fixed-radius balls for each character. Datta *et al.* [6] use video constraints to generate a set of positive and negative face pairs. Perhaps most related to this work, [34] proposes two approaches: *SSiam*, a distance matrix between features on a subset of frames is used to generate hard positive/negative face pairs; and *TSiam* uses temporal constraints and mines negative pairs for singleton tracks by exploiting track-level distances. Features are fine-tuned using a Siamese network with contrastive loss [15].

Different from related work, we propose a simple, yet effective approach where weak labels are generated using a clustering algorithm that is independent of video/track level constraints to learn discriminative face representations. In particular while [18], [38] require ground-truth labels to learn embeddings, CCL does not.

Person identification is a related field of naming characters, and often employs **multimodal** sources of constraints/supervision such as: clothing [37], gender [54], context [51], multispectral information [36]; audio including speech [28] and voice models [25]; and textual cues such as weak labels obtained by aligning transcripts with subtitles [2], [10], joint action/actor labels [22] via transcripts, or name mentions in subtitles [16]. In contrast to the above, CCL is free from additional context and operates directly on detected faces (even without tracking).

FINCH clustering algorithm. Sarfraz *et al.* [30] propose a new clustering algorithm (FINCH) based on first neighbor relations. FINCH does not require training and relies on good representations to discover a hierarchy of partitions. While FINCH demonstrates that pseudo labels can be used to train simple classifiers with cross-entropy loss [30], such a setup is challenging to use in the context of faces. When using a partition at high purity with many more clusters than classes, faces of one character belong to multiple clusters, and the CE loss pushes these clusters and their samples apart – an undesirable effect for our goal of clustering. Thus, in our work, we use FINCH to generate weak positive and negative face pairs that are used to improve video face clustering.

Clustering based learning. Recently clustering is also used to learn an embedding of data [4], [13], [19], [46], [49]. Almost all of these methods cluster data samples at each forward pass (or at pre-determined epochs) into a given number of clusters. Subsequently, generated cluster labels or a loss over the clustering function are used to train a deep model (usually an auto-encoder). These methods require to specify the number of clusters, that is often the target number of classes. This is different from CCL that leverages a large number of small and pure clusters without specifying the particular number of clusters.

In summary, we show how pseudo-labels from clustering can be used effectively: (i) without needing to know the number of clusters, and (ii) by creating positive/negative pairs at a high-purity stage of clustering.

III. Clustering based Representation Learning

We present our method to improve face representations using weak cluster labels instead of manual annotations. We start this section by introducing the notation used throughout the remainder of the paper. We then present a short overview and motivation for FINCH, followed by a description on how we use it to obtain automatic labels. Finally, we discuss how we train our embedding and perform inference at test time. Algorithm 1 provides an overview of our proposed CCL approach, independent to the face clustering task.

A. Preliminaries

Let the dataset contain N face tracks $\mathcal{D} = \{(T_i, y_i)_{i=1}^{N}\}$, where each track T_i has K, a variable number of face images, *i.e.* $T_i = \{(f_i^k)_{k=1}^{K}\}$. $y_i \in \{1, \ldots, C\}$ is the label (person identity) corresponding to the track T_i, where C is the total number of characters in the video. Please note that tracks are not necessary for the proposed learning approach, however, they are used during a part of the evaluation.

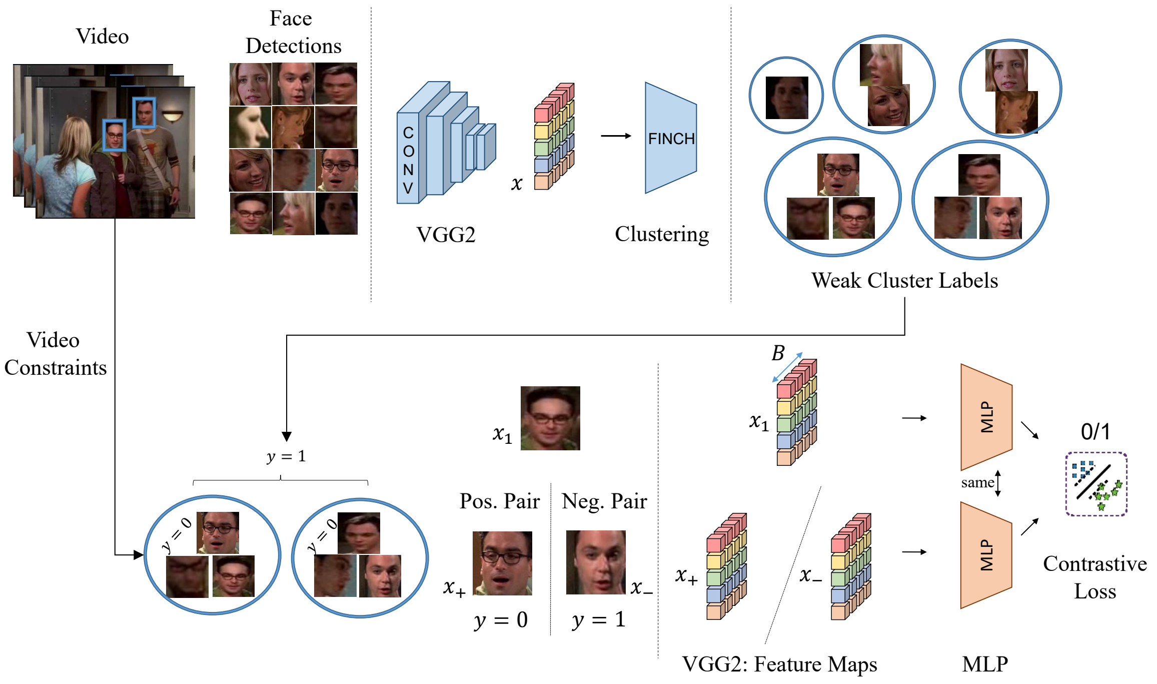

Fig. 1. **CCL training overview:** Given a video with several face detections (top left), we first start by extracting features using a deep CNN and perform clustering using FINCH to obtain a large number of small but highly pure clusters (top). We create several positive and negative face image pairs using these cluster labels and train an MLP to further improve the feature representation using the contrative loss (bottom). At test time, the MLP is used as an embedding, and we cluster our samples using Hierarchical Agglomerative Clustering (HAC).

We use a CNN trained for recognizing faces (VGG2Face [3]) and extract features for each face image in the track from the penultimate layer of the network. The features for a track are vectors $\{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^K\}$ of size $\mathbf{x}_i^k \in \mathbb{R}^{D\times 1}$, where D denotes the feature dimension of the CNN feature maps. We refer to these as *base* features as they are not refined further by a learning approach.

Track-level representations are formed by an aggregation function $\Phi : \{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^K\} \rightarrow \mathbf{v}_i$, that combines K features to produce a single vector $\mathbf{v}_i \in \mathbb{R}^{D\times 1}$. While there are several methods of aggregation such as Fisher Vectors [26], Covariance Learning [40], Quality Aware Networks [21] or Neural Aggregation Networks [50], Temporal 3D Convolution [8], we find that simple average pooling often works quite well. In our case, averaging allows us to learn from features at the frame level (without requiring tracking), but evaluate at the track level.

We additionally ℓ_2 normalize the features to be unit vectors before using them for clustering, *i.e.* $\|\mathbf{v}_i\|_2 = 1$. Several previous works in this area [33], [34], [38], [52], [53] have used Hierarchical Agglomerative Clustering (HAC) as the preferred technique for clustering. For a fair comparison, we too use HAC with the stopping condition set to the number of clusters equalling the number of main cast (C) in the video. We use the minimum variance ward linkage [42] for all methods.

B. Clustering Algorithm

Integral to our core method, we adopt the recently proposed FINCH algorithm [30] to obtain weak labels from clustering. FINCH belongs to the family of hierarchical clustering algorithms and automatically discovers groupings in the data without requiring hyper-parameters or a priori knowledge such as number of clusters. The output of FINCH is a *small* set of partitions that provide a fine to coarse view on the discovered clustering. Note that this is different from classical HAC methods where each iteration merges one sample with existing clusters providing a complete list of partitions ranging from N clusters (all samples in their own cluster) to 1 cluster (all samples in one cluster).

FINCH works by linking first neighbors of each sample. An adjacency matrix is defined between all sample pairs:

$$A(i,j) = \begin{cases} 1 & \text{if } j = \kappa_i^1 \text{ or } \kappa_j^1 = i \text{ or } \kappa_i^1 = \kappa_j^1 \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where κ_i^1 represents the first neighbor of sample i. The adjacency matrix joins each sample i to their first neighbors via $j = \kappa_i^1$, enforces symmetry through $\kappa_j^1 = i$, and links samples that share a common first neighbor $\kappa_i^1 = \kappa_j^1$. Clustering is performed by identifying the connected components in the adjacency matrix.

Why use FINCH? FINCH is a suitable clustering algorithm for our task for several reasons: (i) it does not require specifying the number of clusters; (ii) it provides clusters with very high purity at early partitions; and (iii) it is a fast and scalable algorithm.

As demonstrated in [30], FINCH outperforms Affinity Propagation (AP), Jarvis and Patrick (JP), Robust Continuous Clustering (RCC), and Rank Order Clustering (RO) on several datasets. Additionally, all algorithms AP, JP, RCC, and RO (except FINCH) have memory requirements quadratic in number of samples and are unable to cluster features on our machines with 128 GB RAM.

The first partition of FINCH corresponds to linking samples through the first neighbor relations, while the second partition links clusters created in the first step. We use clusters from the second partition of the FINCH algorithm to mine positive and negative pairs as this partition increases diversity, without compromising on quality of the labels. We find that most samples clustered in the first partition are from within a track (*e.g.* neighboring frames), while those in the second partition are often from across tracks (*c.f.* Fig. 2 (right)). Empirically, we analyze the impact of choosing different partitions through ablation studies.

Using FINCH to obtain weak labels. We treat each face image as a sample and perform clustering with FINCH to obtain the second clustering partition $P_2 = \{G_1, \dots, G_M\}$. G_i corresponds to the i^{th} group of samples. Based on the cluster labels, we obtain positive and negative pairs as:

- **PosC**: All face images in a cluster are considered for positive pairs. For clusters that have less than 10 samples, we also create positive pairs by combining samples from the current cluster G_{p1} with randomly sampled frames from another cluster G_{p2}, where G_{p2} is among the $Z = 25$ nearest clusters to G_{p1}.
- **NegC**: We create negative pairs by combining samples from a cluster G_{n1} with randomly sampled frames from a different cluster G_{n2}, where G_{n2} is among the Z farthest clusters to G_{n1}.

K-means as a baseline. As an alternative to FINCH, we perform experiments with K-means as a baseline to obtain clusters that provide weak labels. Note that K-means has an important disadvantage – we need to know the number of clusters before hand. For a fair empirical comparison, we use the number of clusters estimated by FINCH at each partition. Specifically, we use *MiniBatch K-means* [32] that is more suitable to work with large datasets. Our analysis shows that FINCH provides purer clusters that yield more accurate pairwise labels.

C. Video Level Constraints

Video face clustering often employs face tracking to link face detections made in a sequence of consecutive frames. Note that tracking can be thought of as a form of clustering in a small temporal window (typically within a shot). Using tracks, positive pairs are created by sampling frames within a track, while negative pairs are created by analyzing co-occurring tracks [5], [6], [39], [44].

In this work, we consider video constraints that can be derived at a frame level (*i.e.* without tracking). We include co-occurring face images appearing in the same frame as potential negative pairs (**NVid**). About 50%-70% of the face images appear as singletons in a frame (as shown by TSiam [34]), and do not yield negative pairs through the video level constraints. Thus, being able to sample negative pairs using the cluster labels as discussed above is beneficial.

In addition to sampling negative pairs, we also use co-occurrence to correct clusters provided by FINCH. For example, if a cluster contains samples from a known negative pair, we keep the sample closest to the cluster mean, and create a new cluster using the rejected sample.

Algorithm 1 Overview of CCL (task agnostic).

```
Input: Feature maps $X = \{(\mathbf{x}_n)_{n=1}^N\} \in \mathbb{R}^D$.
$N$ is total number of samples. $D$ is feature dimension.
Output: Final clustering of the $N$ samples.
Procedure:
1. Compute the partitions using FINCH clustering algorithm. $P_{1:L} = \text{FINCH}(X)$.
2. Select second partition $P_2 = \{G_1, \dots, G_M\}$.
3. Train model parameters for 20 epochs:
    – Sample positive and negative pairs for a batch.
    $(\mathbf{x}_{p1}, \mathbf{x}_{p2}) \sim (G_{p1}, G_{p2})$ and $(\mathbf{x}_{n1}, \mathbf{x}_{n2}) \sim (G_{n1}, G_{n2})$.
    – Train MLP $Q_\theta(\cdot)$ using Contrastive loss.
4. Compute embeddings for each sample using $Q_\theta(\mathbf{x}_n)$ and perform HAC to get $C$ clusters.
```

D. Training and Inference

We use the weak cluster labels and video constraints to obtain positive and negative face pairs for training a shallow Siamese network. We adopt the Contrastive loss [15] as it brings together positive samples, in contrast to the widely used Triplet loss [31] that only maintains a margin between positive and negative samples. We sample a pair of face representations $\mathbf{x}_1$ and $\mathbf{x}_2$ with $y = 0$ for a positive pair and $y = 1$ when corresponding to a negative pair and train a multi-layer perceptron Q_θ by minimizing

$$\begin{aligned}\mathcal{L}\left(W, y, Q_\theta(\mathbf{x}_1), Q_\theta(\mathbf{x}_2)\right) = \\ \frac{1}{2}\left((1-y)\cdot(d_W)^2 + y\cdot(\max(0, m-d_W))^2\right),\end{aligned} \tag{2}$$

where $W \in \mathbb{R}^{D\times d}$ is a linear layer that embeds $Q_\theta(\cdot)$ such that $d \ll D$ (in our case, $d = 2$). Here, each face image is encoded as $Q_\theta(\mathbf{x})$, where θ corresponds to the trainable parameters. We find that $Q_\theta(\cdot)$ performs well when using a single linear layer. d_W is the Euclidean distance $d_W = \|W\cdot Q_\theta(\mathbf{x}_1) - W\cdot Q_\theta(\mathbf{x}_2)\|^2$, and m is the margin, empirically chosen to be 1. Fig. 1 illustrates this learning procedure.

During inference, we compute embeddings for face images using the MLP trained above. For frame-level evaluation, we cluster the features for each face image, while for track-level evaluation, we first aggregate image features using mean pool, followed by HAC. We report the clustering performance at the ground truth number of clusters (*i.e.* the number of main characters in the video).

E. Implementation Details

CNN. We employ the VGG-2 Face CNN [3] (ResNet 50) pre-trained on MS-Celeb-1M [14] and fine-tuned on 3.31M face images of 9131 subjects (VGG2 data). We extract `pool5_7x7_s1` features by first resizing RGB face images to 224×224 and pushing them through the CNN, resulting in $\mathbf{x}_k^i \in \mathbb{R}^{2048}$. At test time, we compute face embeddings using the learned MLP Q_θ and ℓ_2 normalize before HAC.

TABLE I
DATASET STATISTICS FOR BBT [43], [44], [52], BF [53] AND ACCIO [53].

Datasets	#Cast	This work #TR (#FR)	LC/SC (%)	Previous work #TR (#FR)
BBT-0101	5	644 (41220)	37.2 / 4.1	182 (11525)
BF-0502	6	568 (39263)	36.2 / 5.0	229 (17337)
ACCIO	36	3243 (166885)	30.93/0.05	3243 (166885)

Siamese network MLP. The network comprises of fully-connected layers: $\mathbb{R}^{2048} \rightarrow \mathbb{R}^{256} \rightarrow \mathbb{R}^{2}$. Note that the second linear layer is part of the contrastive loss (corresponds to W in Eq. 2), and we use the feature representations at $\mathbb{R}^{256}$ for clustering. The model is trained using the Contrastive loss, and parameters are updated using Adam optimizer. We initialize the learning rate with 10^{-5} and decrease it by a factor of 10 at 15 epochs. We train the model for 20 epochs. We use batch normalization.

Sampling positive and negative pairs for training. An epoch corresponds to sampling pairs from all clusters created by the FINCH algorithm. We consider 5 clusters per batch. For each data point in a cluster, we randomly sample from the same (or nearest) cluster and create one positive pair, and sample from the farthest clusters to create two negative pairs. Finally, we randomly subsample the above list to obtain 25 positive and 25 negative pairs for each cluster, resulting in a batch with 250 pairs (125 positive/negative).

IV. EVALUATION

We present our evaluation on three challenging datasets. We first describe the clustering metric, followed by a thorough analysis of the proposed methods, and end this section with a comparison of our approach against state-of-the-art.

Datasets. We conduct experiments on three popular video face clustering datasets: (i) *Buffy the Vampire Slayer* (BF) [2], [53] (season 5, episodes 1): a drama series with several dark scenes and non-frontal faces; (ii) *Big Bang Theory* (BBT) [2], [37], [43], [52] (season 1, episodes 1): a primarily indoors sitcom with a small cast, and (iii) *ACCIO-1* [12]: the first movie in the "*Harry Potter*" series featuring a large number of scenes at night and a higher number of characters.

We follow the protocol used in several recent video face clustering works [5], [34], [44], [52], [53] that focus on improving feature representations for video-face clustering. First, we assume the number of main characters is known. Second, as the labels are obtained automatically, we learn episode specific embeddings. We use the tracks provided by [34] that incorporate several detectors to encompass all pan angles and in-plane rotations up to 45 degrees. The face tracks are obtained by an online tracker that uses a particle filter.

Table I presents key information about the datasets such as the number of tracks (#TR) and face images (#FR). In particular, note that previous works (except [34]) use much smaller datasets. It is also important to note that different characters have wide variations in the number of tracks, indicated by the cluster skew between the largest class (LC) to the smallest class (SC).

TABLE II
CLUSTERING ACCURACY AT TRACK-LEVEL. COMPARISON AGAINST ALL EVALUATED MODELS.

Train/Test	Base	PRF [48]	TSiam [34]	SSiam [34]	CCL
BBT-0101	0.932	0.930	0.964	0.962	**0.982**
BF-0502	0.836	0.814	0.893	0.909	**0.921**

TABLE III
CLUSTERING ACCURACY AT TRACK-LEVEL, EXTENDING TO ALL NAMED CHARACTERS. BBT-0101 HAS 5 MAIN AND 6 NAMED CHARACTERS; BF-0502 HAS 6 MAIN AND 12 NAMED CHARACTERS.

	BBT-0101			BF-0502		
	TSiam	SSiam	CCL	TSiam	SSiam	CCL
Main cast	0.964	0.962	**0.982**	0.893	0.909	**0.921**
All named	0.958	0.922	**0.966**	0.829	0.870	**0.903**

Evaluation metric. We use Clustering Accuracy (ACC) [34] also called Weighted Clustering Purity (WCP) [39] as the metric to evaluate the quality of clustering. Accuracy is computed by assigning the most common ground truth label within a cluster to all elements in that cluster:

$$\text{ACC} = \frac{1}{N} \sum_{c=1}^{|C|} n_c \cdot p_c \,, \tag{3}$$

where N is the total number of samples, n_c is the number of samples in the cluster c, and cluster purity p_c is measured as the fraction of the largest number of samples from the same label to n_c. C corresponds to the number of main casts. For ACCIO, we also report B-Cubed Precision (P), Recall (R) and F-measure (F) [1]. Unless stated otherwise, clustering evaluation is performed at track-level.

A. *Clustering Performance and Generalization*

Comparison against baselines. We present a thorough comparison of CCL against related baselines that employ a similar learning scheme.

In pseudo-relevance feedback [48], [47] (PRF), samples are treated independent of each other, and pairs are created by considering closest positives and farthest negatives. However, these pairs often provide negligible training signal (gradients) as they already conform to the loss requirements.

SSiam [34] is an improved version of PRF with batch processing where farthest positives and closest negatives are formed by looking at a batch of queries rather than whole dataset. This creates harder training samples that show improved performance over PRF.

In TSiam [34], positive pairs are formed by looking at samples within a track (a track is treated as a cluster of face images) and negative pairs by using co-occurrence and distances between track representations.

Finally, our proposed method CCL does not rely on tracking and uses pure clusters created by an automatic

TABLE IV

A study on the impact of clustering algorithm. FINCH partitions are created for each dataset, and shown as separate table rows. In each row, results are presented for FINCH (above) and MiniBatch K-means (below). From left to right, **Part.** indicates the partition level of FINCH. **#C** is the total number of clusters in that partition as estimated by FINCH. Largest and smallest cluster sizes are indicated as **LC/SC**. Clustering purity of FINCH/K-means clusters (before CCL) is presented as **ACC**. **L+/L-** represents the number of samples correctly and wrongly clustered for the given partition. Finally, **CCL-ACC** is the performance of CCL: by training a model using weak labels from FINCH/K-means estimated clusters.

FINCH	BBT-0101					BF-0502					ACCIO				
Partition	#C **41220**	LC/SC	ACC	L+/L-	CCL-ACC @#C=5	#C **39263**	LC/SC	ACC	L+/L-	CCL-ACC @#C=6	#C **166885**	LC/SC	ACC	L+/L-	CCL-ACC @#C=36
1	10156	48/2	0.997	41128/92	0.978	9677	42/2	0.994	39054/209	0.915	21444	3937/2	0.847	141496/25389	0.816
		1030/1	0.988	40744/476	0.971		758/1	0.986	38717/546	0.909		16041/1	0.899	150177/16708	0.841
2	**2236**	777/4	0.990	40809/411	**0.995**	**2167**	1346/4	0.978	38414/849	**0.938**	**3972**	33461/4	0.832	138903/27982	**0.837**
		1236/1	0.987	40687/533	0.991		1566/1	0.971	38161/1102	0.923		18076/1	0.866	144613/22272	**0.857**
3	490	2031/8	0.974	40149/1071	0.954	560	2574/9	0.957	37588/1675	0.922	944	41405/13	0.812	135543/31342	0.801
		1568/1	0.979	40367/853	0.957		2120/1	0.962	37806/1457	0.926		9838/1	0.845	141036/25849	0.813
4	101	7258/32	0.968	39936/1284	0.943	127	9335/27	0.923	36260/3003	0.903	161	70825/39	0.784	130987/35898	0.767
		2019/1	0.981	40475/745	0.945		2360/1	0.956	37555/1708	0.923		16502/1	0.800	133639/33246	0.782
5	13	10.9K/0.K	0.968	39.9K/1.2K	0.943	24	13.7K/0.2K	0.895	35.1K/4.0K	0.869	24	84.8K/0.2K	0.690	115.1K/51.7K	0.652
		7.3K/1	0.966	39.8K/1.3K	0.931		3.7K/0.2K	0.920	36.1K/3.1K	0.875		26.7K/1	0.724	120.9K/45.9K	0.696

TABLE V

Impact of mining positive and negative pairs from different sources. Track-level accuracy of CCL.

		PosC	NegC	NVid	BBT-0101	BF-0502
1	Base	-	-	-	0.932	0.836
2		✓	-	-	0.951	0.859
3		-	✓	-	0.968	0.883
4		✓	-	✓	0.956	0.865
5		✓	✓	-	0.971	0.896
6		-	✓	✓	0.979	0.917
7	CCL	✓	✓	✓	**0.982**	**0.921**

partitioning method (FINCH). We sort clusters by distances between their mean representations and then form positive and negative pairs. Table II shows that CCL outperforms both SSiam and TSiam, and also provides significant gains over the base VGG2 features on BBT and BF.

Generalization to unseen characters. We further study how CCL can cope with adding unseen characters at test time by clustering all named characters in an episode. Results from Table III show that CCL is better poised at clustering new characters and achieves higher performance in all cases. We believe that CCL's performance scales well as it is trained on a diverse set of pairs and can generalize to unseen characters.

B. Sources of Positive and Negative Pairs

In Table V, we present the importance of each source from which we obtain positive and negative pairs. Recall that positive pairs obtained from clusters are denoted as **PosC**; negative pairs from clusters as **NegC**; and negative pairs using video constraints as **NVid**.

Rows 2 to 6 evaluate different combinations of the sources of pairs and show their relative importance. It can be seen that negative pairs alone (row 3) are more important than positive pairs (row 2). Additionally, using pairs from the clusters alone (row 5) provides performance similar to TSiam and SSiam (refer to Table II). In Fig. 2, we show the number of faces in each cluster and the number of tracks that they are derived from. CCL's ability to obtain positive pairs from faces across multiple tracks (different from TSiam) might be an explanation for improved performance. Further, including negatives from videos (row 7) provides best performance. This means that our learning approach sees relatively hard positive pairs, and standard (neither hard nor easy) negative pairs. We believe that together with the contrastive loss that brings together samples from positive pairs, this is a good way to learn representations, and is an important factor for the increased performance.

C. Impact of Clustering Algorithm

We now present a study on the impact of the clustering algorithm used for obtaining weak labels. We first obtain the hierarchy of partitions by processing each dataset with the FINCH algorithm. Then, as a comparison, we use MiniBatch K-means with K set to the number of clusters provided by FINCH. Each row of Table IV shows performance at one partition. Results using FINCH are in the top half and K-means in the bottom half. The number of clusters in higher partitions (6 onwards) are less than the number of characters and are omitted.

We observe that FINCH not only automatically discovers meaningful partitions of the data but also achieves higher-performance (ACC in Table IV) as compared to K-means, especially at higher number of clusters. It is interesting to note the number of samples that are clustered correctly or wrongly (L+/L- in Table) as these play an important role while creating weak labels. Fig. 2 (left) shows that while the clusters are often very small (< 10 samples), they contain have faces from more than one track (right). CCL with FINCH at the second partition obtains high performance after training with weak labels (CCL-ACC) and outperforms labels provided by K-means clustering.

Additionally, as indicated earlier, running FINCH on large datasets such as ACCIO takes $\sim$2 minutes with fast nearest

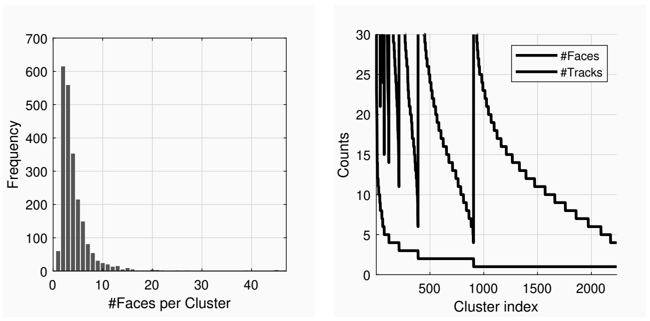

Fig. 2. Key characteristics of FINCH (second partition) clustering for BBT-0101. **Left:** Histogram showing the number of faces in a cluster. Even if most clusters have less than 5 samples, we are able to obtain meaningful positive and negative pairs to train our model. **Right:** We plot the number of faces and number of tracks for each of the cluster indices (sorted by size for convenience). About 900 of the 2200 clusters created by FINCH contain faces from more than one track, leading to increased diversity of pairs.

TABLE VI
COMPARISON TO STATE-OF-THE-ART WITH CLUSTERING ACCURACY (%) AT FRAME LEVEL. NOTE THAT MANY PREVIOUS WORKS USE FEWER TRACKS (# OF FRAMES) (ALSO INDICATED IN TABLE I) MAKING THE TASK RELATIVELY EASIER. WE USE THE TRACKS FROM [30], [34].

Method	BBT-0101	BF-0502	Data Source	
			BBT	BF
FINCH (CVPR '19) [30]	99.16	92.73	[2]*	[2]*
ULDML (ICCV '11) [5]	57.00	41.62	–	[5]
HMRF (CVPR '13) [44]	59.61	50.30	[29]	[10]
HMRF2 (ICCV '13) [43]	66.77	–	[29]	–
WBSLRR (ECCV '14) [45]	72.00	62.76	–	[10]
VDF (CVPR '17) [33]	89.62	87.46	[2]	[2]
Imp-Triplet (PacRim '16) [52]	96.00	–	[29]	–
JFAC (ECCV '16) [53]	–	92.13	–	[10]
TSiam (FG '19) [34]	98.58	92.46	[2]*	[2]*
SSiam (FG '19) [34]	99.04	90.87	[2]*	[2]*
CCL (Ours with HAC)	**99.56**	**93.79**	[34]	[34]

neighbors as compared to MiniBatch K-means that takes $\sim$1 hour even though it is optimized for large datasets.

D. Comparison with the state-of-the-art

While previous analysis has been done at the track-level, we now report frame-level clustering performance to present a fair comparison against previous work.

BBT and BF. We compare the results obtained with CCL to the current state-of-the-art approaches for video face clustering in Table VI. We report clustering accuracy (%) on two videos: BBT-0101 and BF-0502. Following [34], we similarly report the data source to indicate that our dataset is much harder in comparison to previous works such as [53], [52], that evaluate on a subset of tracks. We can observe that CCL outperforms previous approaches, achieving 0.52% and 1.33% absolute improvement in accuracy on BBT-0101 and BF-0502 respectively.

ACCIO. As common in previous works [34], [53], we evaluate our method on the ACCIO dataset with 36 (the main cast) and 40 clusters – see Table VIII. CCL significantly outperforms the state-of-the-art in unsupervised learning techniques and is comparable to [30]. Note that FINCH [30] is not trainable.

TABLE VII
FRAME-LEVEL ACCURACY (%) OF CCL AND COMPARISON TO TSIAM AND SSIAM OVER ALL DATASETS: BBT-0101, BF-0502 AND ACCIO.

	#Cast	Base	TSiam [34]	SSiam [34]	CCL
BBT-0101	5	94.00	98.58	99.04	**99.56**
BF-0502	6	91.20	92.46	90.87	**93.79**
ACCIO	36	79.90	81.30	82.00	**83.40**

TABLE VIII
COMPARISON OF CCL WITH THE STATE-OF-THE-ART ON ACCIO, EVALUATED AT **36** AND **40** CLUSTERS. CLUSTERING ACCURACY AT FRAME-LEVEL.

#Clusters = 36			
Methods	P	R	F
FINCH (CVPR '19) [30]	0.748	0.677	0.711
JFAC (ECCV '16) [53]	0.690	0.350	0.460
TSiam (FG '19) [34]	0.749	0.382	0.506
SSiam (FG '19) [34]	0.766	0.386	0.514
CCL (Ours with HAC)	**0.779**	0.402	**0.530**

#Clusters = 40			
Methods	P	R	F
FINCH (CVPR '19) [30]	0.733	0.711	0.722
DIFFRAC-DeepID2$^+$ (ICCV '11) [53]	0.557	0.213	0.301
WBSLRR-DeepID2$^+$ (ECCV '14) [53]	0.502	0.206	0.292
HMRF-DeepID2$^+$ (CVPR '13) [53]	0.599	0.230	0.332
JFAC (ECCV '16) [53]	0.711	0.352	0.471
TSiam (FG '19) [34]	0.763	0.362	0.491
SSiam (FG '19) [34]	0.777	0.371	0.502
CCL (Ours with HAC)	**0.786**	0.392	**0.523**

CCL vs. TSiam and SSiam. In Table VII, we report the frame-level clustering performance on all three datasets (as compared to track-level performance in Table II). We observe that CCL outperforms TSiam, SSiam and also the base features by significant gains. CCL operates directly on face detections (like SSiam), while TSiam requires tracks.

Computational complexity. The time to compute FINCH partitions for BBT, BF, and ACCIO is approximately 45 seconds, and $\sim$2 minute respectively on CPU. Training CCL for 20 epochs on BBT-0101 requires less than half an hour on a GTX 1080 GPU using the PyTorch framework.

V. CONCLUSION

We proposed a self-supervised, clustering-based contrastive learning approach for improving deep face representations. We showed that we can train discriminative models using positive and negative pairs obtained through clustering and video level constraints that do not rely on face tracking. Through experiments on three challenging datasets, we showed that CCL achieves state-of-the-art performance while being computationally efficient and easily scalable.

Acknowledgments. This work is supported by the DFG (German Research Foundation) funded PLUMCOT project.

References

[1] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 2009.

[2] M. Bäuml, M. Tapaswi, and R. Stiefelhagen. Semi-supervised Learning with Constraints for Person Identification in Multimedia Data. In *CVPR*, 2013.

[3] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. VGGFace2: A Dataset for Recognising Faces across Pose and Age. In *FG*, 2018.

[4] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[5] R. G. Cinbis, J. Verbeek, and C. Schmid. Unsupervised Metric Learning for Face Identification in TV Video. In *ICCV*, 2011.

[6] S. Datta, G. Sharma, and C. Jawahar. Unsupervised learning of face representations. In *FG*, 2018.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] A. Diba, M. Fayyaz, V. Sharma, A. Hossein Karami, M. Mahdi Arzani, R. Yousefzadeh, and L. Van Gool. Temporal 3d convnets using temporal transition layer. In *CVPR Workshops*, 2018.

[9] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[10] M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is ... Buffy" Automatic Naming of Characters in TV Video. In *BMVC*, 2006.

[11] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017.

[12] E. Ghaleb, M. Tapaswi, Z. Al-Halah, H. K. Ekenel, and R. Stiefelhagen. Accio: A Dataset for Face Track Retrieval in Movies Across Age. In *ICMR*, 2015.

[13] X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, 2017.

[14] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. In *ECCV*, 2016.

[15] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*, 2006.

[16] M.-L. Haurilet, M. Tapaswi, Z. Al-Halah, and R. Stiefelhagen. Naming TV Characters by Watching and Analyzing Dialogs. In *WACV*, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] Y. He, K. Cao, C. Li, and C. C. Loy. Merge or not? learning to group faces via imitation learning. In *AAAI*, 2018.

[19] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *IJCAI*, 2017.

[20] S. Jin, H. Su, C. Stauffer, and E. Learned-Miller. End-to-end Face Detection and Cast Grouping in Movies using ErdsRnyi Clustering. In *ICCV*, 2017.

[21] Y. Liu, J. Yan, and W. Ouyang. Quality Aware Network for Set to Set Recognition. In *CVPR*, 2017.

[22] A. Miech, J.-B. Alayrac, P. Bojanowski, I. Laptev, and J. Sivic. Learning from video and text via large-scale discriminative clustering. In *ICCV*, 2017.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[24] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016.

[25] A. Nagrani and A. Zisserman. From Benedict Cumberbatch to Sherlock Holmes: Character Identification in TV series without a Script. In *BMVC*, 2017.

[26] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A Compact and Discriminative Face Track Descriptor. In *CVPR*, 2014.

[27] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep Face Recognition. In *BMVC*, 2015.

[28] G. Paul, K. Elie, M. Sylvain, O. Jean-Marc, and D. Paul. A conditional random field approach for audio-visual people diarization. In *ICASSP*, 2014.

[29] M. Roth, M. Bäuml, R. Nevatia, and R. Stiefelhagen. Robust Multi-pose Face Tracking by Multi-stage Tracklet Association. In *ICPR*, 2012.

[30] M. S. Sarfraz, V. Sharma, and R. Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *CVPR*, 2019.

[31] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *CVPR*, 2015.

[32] D. Sculley. Web-scale K-means Clustering. In *International Conference on World Wide Web*, 2010.

[33] V. Sharma, M. S. Sarfraz, and R. Stiefelhagen. A simple and effective technique for face clustering in tv series. In *CVPR: Brave New Motion Representations Workshop*, 2017.

[34] V. Sharma, M. Tapaswi, M. S. Sarfraz, and R. Stiefelhagen. Self-supervised learning of face representations for video face clustering. In *FG*, 2019.

[35] V. Sharma, M. Tapaswi, and R. Stiefelhagen. Deep multimodal feature encoding for video ordering and retrieval tasks. In *ICCV Workshop on Holistic Video Understanding (HVU)*, 2019.

[36] V. Sharma and L. Van Gool. Image-level classification in hyperspectral images using feature descriptors, with application to face recognition. *arXiv:1605.03428*, 2016.

[37] M. Tapaswi, M. Bäuml, and R. Stiefelhagen. "Knock! Knock! Who is it?" Probabilistic Person Identification in TV-Series. In *CVPR*, 2012.

[38] M. Tapaswi, M. T. Law, and S. Fidler. Video face clustering with unknown number of clusters. In *ICCV*, 2019.

[39] 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 *ICVGIP*, 2014.

[40] R. Wang, H. Guo, L. S. Davis, and Q. Dai. Covariance Discriminative Learning: A Natural and Efficient Approach to Image Set Classification. In *CVPR*, 2012.

[41] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[42] J. H. Ward Jr. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[43] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji. Simultaneous Clustering and Tracklet Linking for Multi-face Tracking in Videos. In *ICCV*, 2013.

[44] B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji. Constrained Clustering and its Application to Face Clustering in Videos. In *CVPR*, 2013.

[45] S. Xiao, M. Tan, and D. Xu. Weighted Block-sparse Low Rank Representation for Face Clustering in Videos. In *ECCV*, 2014.

[46] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

[47] R. Yan, A. Hauptmann, and R. Jin. Multimedia search with pseudo-relevance feedback. In *International Conference on Image and Video Retrieval*, 2003.

[48] R. Yan, A. G. Hauptmann, and R. Jin. Negative pseudo-relevance feedback in content-based video retrieval. In *ACM MM*, 2003.

[49] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

[50] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua. Neural Aggregation Network for Video Face Recognition. In *CVPR*, 2017.

[51] L. Zhang, D. V. Kalashnikov, and S. Mehrotra. A unified framework for context assisted face clustering. In *ICMR*, 2013.

[52] S. Zhang, Y. Gong, and J. Wang. Deep Metric Learning with Improved Triplet Loss for Face Clustering in Videos. In *Pacific Rim Conference on Multimedia*, 2016.

[53] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Joint Face Representation Adaptation and Clustering in Videos. In *ECCV*, 2016.

[54] C. Zhou, C. Zhang, H. Fu, R. Wang, and X. Cao. Multi-cue augmented face clustering. In *ACM MM*, 2015.