Self-Supervised Learning of Face Representations for Video Face Clustering

Vivek Sharma[1], Makarand Tapaswi[2], M. Saquib Sarfraz[1] and Rainer Stiefelhagen[1]
[1] CV:HCI, Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] University of Toronto, Toronto, Canada

***Abstract*— Analyzing the story behind TV series and movies often requires understanding who the characters are and what they are doing. With improving deep face models, this may seem like a solved problem. However, as face detectors get better, clustering/identification needs to be revisited to address increasing diversity in facial appearance. In this paper, we address video face clustering using unsupervised methods. Our emphasis is on distilling the essential information, *identity*, from the representations obtained using deep pre-trained face networks. We propose a self-supervised Siamese network that can be trained without the need for video/track based supervision, and thus can also be applied to image collections. We evaluate our proposed method on three video face clustering datasets. The experiments show that our methods outperform current state-of-the-art methods on all datasets. Video face clustering is lacking a common benchmark as current works are often evaluated with different metrics and/or different sets of face tracks. The datasets and code are available at `https://github.com/vivoutlaw/SSIAM`.**

I. Introduction

Large videos such as TV series episodes or movies undergo several preprocessing steps to make the video more accessible, *e.g.* shot detection. Character identification/clustering has become one such important step with several emerging research areas [25], [35], [37] requiring it. For example, in video question-answering [35], most questions center around the characters asking who they are, what they do, and even why they act in certain ways. The related task of video captioning [25] often uses a character agnostic way (replacing names by *someone*) making the captions very artificial and uninformative (*e.g. someone* opens the door). However, recent work [24] suggests that more meaningful captions can be achieved from an improved understanding of characters. In general, the ability to predict which characters appear when and where facilitates a deeper video understanding that is grounded in the storyline.

Motivated by this goal, person clustering [4], [10], [15], [48] and identification [7], [31], [2], [23], [19] in videos has seen over a decade of research. In particular, fully automatic person identification is achieved either by aligning subtitles and transcripts [7], [31], [2], or using web images for actors and characters [1], [19] as a form of weak supervision. On the other hand, clustering [48], [4], [41], [42], [47], [5] has mainly relied on *must-link* and *cannot-link* information obtained by tracking faces in a shot and analyzing their co-occurrence.

978-1-7281-0089-0/19/$31.00 ©2019 IEEE

As face detectors improve (*e.g.* [13]), clustering and identification need to be revisited as more faces that exhibit extreme viewpoints, illumination, resolution, become available and need to be grouped or identified. Along with improvements to face detection, deep Convolutional Neural Networks (CNNs) have also yielded large performance gains for face representations [32], [28], [21], [3]. These networks are typically trained using hundreds-of-thousands to millions of face images gathered from the web, and show super-human performance on face verification tasks on images (LFW [14]) and videos (YouTubeFaces [40]). Nevertheless, it is important to note that faces in videos such as TV series/movies exhibit more variety in comparison to *e.g.* LFW, where the images are obtained from Yahoo News by cropping mostly frontal faces. While these deep models generalize well, they are difficult to train from scratch (require lots of training data), and are typically transferred to other datasets via *net surgery*: fine-tuning [29], [47], [48], or use of additional embeddings on the features from the last layer [27], [6], or both.

Recent work shows that CNN representations can be improved via positive and negative pairs that are discovered through an MRF [48]; or a revised triplet-loss [47]. In contrast, we propose simple methods that do not require complex optimization functions or supervision to improve the feature representation. We emphasize that while video-level constraints arent new, they need to be used properly to extract the most out of them. This is especially in light of CNN face representations that are very similar even across different identities. Fig. 5 proves this point as we see a large overlap between the cosine similarity distributions of positive (same id) and negative (across id) track pairs on the base features.

In this paper, we focus on the video face clustering problem. Given a set of face tracks from several characters, our goal is to group them so that tracks in a cluster belong to the same character. We summarize the main contributions of our paper, and also highlight key differences: (1) We propose two variants of discriminative methods (Sec. III) that build upon deep network representations, to learn discriminative facial representations. They are Track-supervised Siamese network (TSiam) and Self-supervised Siamese network (SSiam). In contrast to previous methods [5], in TSiam, we incorporate negative pairs for the singleton (no co-occurring) tracks. In SSiam, we obtain positives and negatives by sorting distances (*i.e.* ranking) on a subset of frames. Note that methods proposed in this paper are either fully unsupervised, or use supervision that is obtained automatically, hence can be thought as unsupervised. Additionally, our fully unsupervised method can mine positive and negative pairs without the need for tracking. This enables application of our method to image collections. (2) We perform extensive empirical studies and demonstrate the effectiveness and generalisation of the methods. Our methods are powerful and obtain performance comparable or higher than state-of-the-art when evaluated on three challenging video face clustering datasets.

II. Related Work

Over the last decade, video face clustering is typically modeled using discriminative methods to improve face representations. In the following, we review some related work in this area.

Video face clustering. Clustering faces in videos commonly uses pairwise constraints obtained by analyzing tracks and some form of representation/metric learning. Face image pairs belonging to the same track are labeled positive (same character), while face images from co-occurring tracks help create negatives (different characters). This strategy has been exploited by learning a metric to obtain cast-specific distances [4] (ULDML); iteratively clustering and associating short sequences based on hidden Markov Random Field (HMRF) [41], [42]; or performing clustering in a sub-space obtained by a weighted block-sparse low-rank representation (WBSLRR) [43]. In addition to pairwise constraints, video editing cues are used in an unsupervised way to merge tracks [34]. Here, track and cluster representations are learned on-the-fly with dense-SIFT Fisher vectors [20]. Recently, Jin *et al*. [15] consider detection and clustering jointly, and propose a link-based clustering (Erdös-Rényi) based on rank-1 counts verification. The linking is done by comparing a given frame with a reference frame and a threshold is learned to merge/not-merge frames.

Face track clustering/identification methods have also used additional cues such as clothing appearance [33], speech [22], voice models [19], context [46], gender [49], name mentions (first, second, and third person references) in subtitles [18], weak labels using transcripts/subtitles [7], [2], and joint action and actor labeling [16] using transcripts.

With the popularity of CNNs, there is a growing focus on improving face representations using video-level constraints. An improved form of triplet loss is used to fine-tune the network and push the positive and negative samples apart in addition to requiring anchor and positive to be close, and anchor and negative far [47]. Zhang *et al*. [48] learn better representations by dynamic clustering constraints that are discovered iteratively during clustering that is performed via a Markov Random Field (MRF). In contrast to related work, we propose a simple, yet effective approach (SSiam) to learn good representations by sorting distances on a subset of frames and not requiring video/track level constraints to generate positive/negative training pairs.

Further, Zhang *et al*. [47]-[48] and Datta *et al*. [5] utilize video-level constraints only to generate a set of similar and

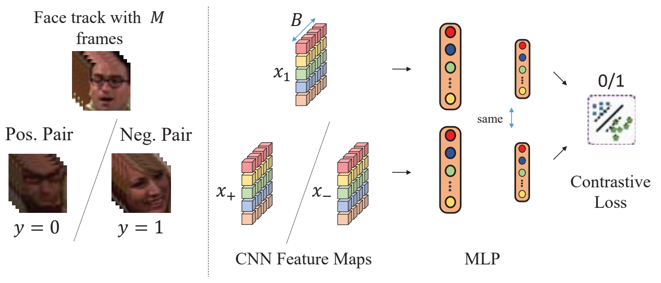

Fig. 1. **Track-supervised Siamese network (TSiam)**. Illustration of the Siamese architecture used in our track-supervised Siamese networks. Note that the MLP is shared across both feature maps. B corresponds to batch size.

dissimilar face pairs. Thus, the model does not see negative pairs for singleton (no co-occurring) tracks. In contrast, our method TSiam incorporates negative pairs for the singleton tracks by exploiting track-level distance.

Finally, it is worth noting the work on "harvesting" training data from unlabeled sources which is in the similar spirit of SSiam and TSiam. Fernando *et al*. [8] and Mishra *et al*. [17] shuffle the video frames and treat them as positive/negative training data for reordering video frames; Wang *et al*. [38] collect positive/negative training data by tracking bounding boxes (*i.e*. motion information) in order to learn effective visual representations.

III. Refining Face Representations for Clustering

Our goal is to improve face representations using simple methods that build upon the success of deep CNNs. More precisely, we propose models to refine the face descriptors automatically, without the need for manual labels. In contrast, fine-tuning the original CNN typically requires supervised class labels. Thus, our approach has three key benefits: (i) it is easily applicable to new videos; (ii) it does not need large amounts of training data (few hundred tracks are enough); and (iii) specialized networks can be trained on each episode, film, or series.

Preliminaries. Consider a video with N face tracks $\{T^1, \ldots, T^N\}$ belonging to C characters. Each track corresponds to one of the characters, and consists of $T^i = \{f_1, \ldots, f_{M^i}\}$ face images. Our goal is to group tracks into $\{G_1, \ldots, G_{|C|}\}$ such that each track is assigned to only one group, and ideally, each group contains all tracks from the same character. We use a deep CNN (VGG2 [3]) and extract a descriptor for each face image $\mathbf{x}_k^i \in \mathbb{R}^D$, $k = 1, \ldots, M^i$ from the penultimate layer (before classification) of the network. We refer to these as base features, and demonstrate that they already achieve a high performance. As a form of data augmentation, we use 10 crops obtained from an expanded bounding box surrounding the face image during training. Evaluation is based on one center crop.

Track-level representations are obtained by aggregating the

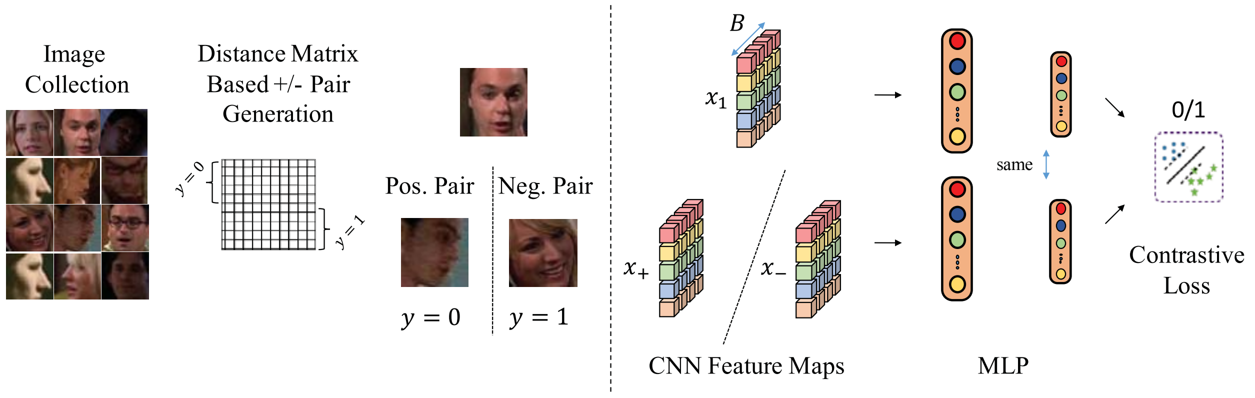

Fig. 2. **Self-supervised Siamese network (SSiam)**. Illustration of the Siamese architecture used in our self-supervised Siamese networks. SSiam selects hard pairs: farthest positives and closest negatives using a ranked list based on Euclidean distance for learning similarity and dissimilarity respectively. Note that the MLP is the same across both feature maps. B corresponds to batch.

face image descriptors

$$\mathbf{t}^i = \frac{1}{M^i} \sum_k \mathbf{x}_k^i \,. \tag{1}$$

We additionally normalize track representations to be unit-norm, $\hat{\mathbf{t}}^i = \mathbf{t}^i / \|\mathbf{t}^i\|_2$ before using them for clustering.

In [34], [47], [48], the authors use Hierarchical Agglomerative Clustering (HAC) as the clustering technique. For a fair comparison to [34], [47], [48], we perform HAC to obtain a fixed number of clusters, equal to the number of characters. We use the minimum variance ward linkage [39] for all methods. See Fig. 3 for illustration.

Discriminative models. Discriminative clustering models typically associate a label y with a pair of features. We choose $y = 0$ when a pair of features $(\mathbf{x}_1, \mathbf{x}_2)$ should belong to the same cluster, and $y = 1$ otherwise [12].

We use a shallow MLP to reduce the dimensionality and improve generalization of the features (see Fig. 1, 2). Here, each face image is encoded as $Q_\phi(\mathbf{x}_k^i)$, where ϕ corresponds to the trainable parameters. We find $Q_\phi(\cdot)$ to perform best when using a linear layer (for details see Sec. IV-B). To perform clustering, we compute track-level aggregated features by average pooling across the embedded frame-level representations [30]

$$\mathbf{t}^i = \frac{1}{M^i} \sum_k Q_\phi(\mathbf{x}_k^i) \,, \tag{2}$$

followed by ℓ_2-normalization.

We train our model parameters at the frame-level by minimizing the contrastive loss [12]:

$$\begin{aligned} &\mathcal{L}\left(W, y, Q_\phi(\mathbf{x}_1), Q_\phi(\mathbf{x}_2)\right) = \\ &\quad \frac{1}{2}\left((1-y)\cdot(d_W)^2 + y\cdot(\max(0, m-d_W))^2\right) \,, \end{aligned} \tag{3}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are a pair of face representations with $y = 0$ when coming from the same character, and $y = 1$ otherwise. $W : \mathbb{R}^{D\times d}$ is a linear layer that embeds $Q_\phi(\mathbf{x})$ such that $d \ll D$ (in our case, $d = 2$). d_W is the Euclidean distance $d_W = \|W \cdot Q_\phi(\mathbf{x}_1) - W \cdot Q_\phi(\mathbf{x}_2)\|^2$, and m is the margin, empirically chosen to be 1.

In the following, we present two strategies to automatically obtain supervision for pairs of frames: Fig. 1 illustrates the Track-level supervision, and Fig. 2 shows the Self-supervision for Siamese network training.

Track-supervised Siamese network (TSiam). Video face clustering often employs face tracking to group face detections made in each frame. The tracking acts as a form of high precision clustering (grouping detections within a shot) and is popularly used to automatically generate positive and negative pairs of face images [4], [5], [34], [42]. In each frame, we assume that characters appear on screen only once. Thus, all face images within a track can be used as positive pairs, while face images from co-occurring tracks are used as negative pairs. For each frame in the track, we sample two frames within the same track to form positive pairs, and sample four frames from a co-occurring track (if it exists) to form negative pairs.

Depending on the filming style of the series/movie, characters may appear alone or together on screen. As we perform experiments on diverse datasets, for some videos 35% tracks have co-occurring tracks, while this can be as large as 70% for other videos. For isolated tracks, we sort all other tracks based on track-level distances (computed on base features) and randomly sample frames from the farthest $F = 25$ tracks. Note that all previous works ignore negative pairs for singleton (not co-occurring) tracks. We will highlight their impact in our experiments.

Self-supervised Siamese network (SSiam). Clustering is an unsupervised task and supervision from tracks may not always be available. An example is face clustering within image collections (*e.g.* on social media platforms). To enable the use of metric learning without any supervision we propose an effective approach that can generate the required pairs automatically during training. SSiam is inspired by pseudo-relevance feedback (pseudo-RF) [45], [44] that is commonly used in information retrieval.

We hypothesize that the first and last samples of a ranked list based on Euclidean distance are strong candidates for learning similarity and dissimilarity respectively. We exploit this in a meaningful way and generate promising similar and dissimilar pairs from a representative subset of the data.

Formally, consider a subset $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_B\}$ of frames from the dataset. We treat each frame $\mathbf{x}_b, b = 1, \dots, B$ as a query and compute Euclidean distance against every other frame in the set. We sort rows of the resulting matrix in an ascending order (smallest to largest distance) to obtain an ordered index matrix $\mathcal{O}(\mathcal{S}) = [s^o_1; \dots; s^o_B]$. Each row s^o_b contains an ordered index of the closest to farthest faces corresponding to $\mathbf{x}_b$. Note that the first column of such a matrix is the index itself at distance 0. The second column corresponds to nearest neighbors for each frame and can be used to form the set of positive pairs $\mathcal{S}_+$. Similarly, the last column corresponds to farthest neighbors and forms the set of negative pairs $\mathcal{S}_-$. Each element of the above sets stores: query index b, nearest/farthest neighbor r, and the Euc. distance d. However, we still need to consider the choice of: (i) the set of frames, and (ii) training pairs from the candidate sets of positive and negative pairs.

We address this during training and form the pairs dynamically by picking a random subset of B frames at each iteration. We compute the distances, sort them, and obtain positive and negative pairs sets $\mathcal{S}_+, \mathcal{S}_-$, each with B elements. Among them, we choose K pairs from the positive set that have the largest distances and K pairs from the negative set with the smallest distances. This allows us to select pairs semi-hard positives and semi-hard negatives from the representative set of B frames. Finally, these $2K$ pairs form the training batch for the network.

To encourage variety in the sample set $\mathcal{S}$ and reduce the chance of false positives/negatives in the chosen $2K$ pairs, B is chosen to be much larger than K ($B = 1000, K = 64$). Experiments on difficult datasets and generalization studies show the benefit and effectiveness of this approach in collecting positive and negative pairs to train the network.

Note that, SSiam can be thought of as an improved version of pseudo-RF with batch processing. Rather than selecting farthest negatives and closest positives for each independent query, we emphasize that SSiam selects $2K$ hard pairs: farthest positives and closest negatives by looking at the batch of queries B jointly. This selection of sorted pairs from the positive $\mathcal{S}_+$ and negative $\mathcal{S}_-$ sets is quite important. See experiments for a more detailed comparison of SSiam with pseudo-RF.

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, ending with a comparison to state-of-the-art.

A. Experimental Setup

Datasets. We conduct experiments on three challenging video face identification/clustering datasets: (i) *Buffy the Vampire Slayer* (BF) [2], [48] (season 5, episodes 1 to 6): a drama series with several shots in the dark at night; (ii) *Big Bang Theory* (BBT) [33], [2], [41], [47] (season 1, episodes 1 to 6): a sitcom with small cast list shot mainly indoors, and (iii) *ACCIO* [9]: *Accio-1* first installment of "*Harry Potter*" movie series with a large number of dark scenes and several tracks with non-frontal faces.

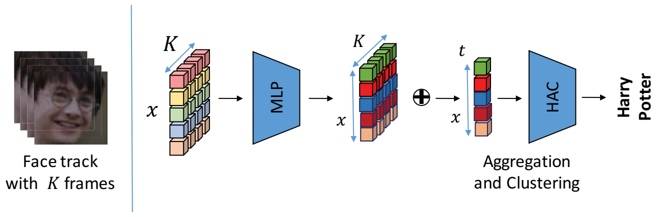

Fig. 3. **Illustration of the testing evaluation**. Given our pre-trained MLP models: TSiam and SSiam. We extract the frame-level features for the track, followed by mean pool for a track-level representation, which is then fed to HAC clustering with fixed number of clusters. Finally, we evaluate the quality of clustering via Accuracy or BCubed F-measure.

TABLE I
Dataset statistics for BBT [41], [42], [47], BF [48] and ACCIO [48].

Datasets	#Cast	This work #TR (#FR)	LC/SC (%)	Previous work #TR (#FR)
BBT0101	5	644 (41220)	37.2 / 4.1	182 (11525)
BF0502	6	568 (39263)	36.2 / 5.0	229 (17337)
ACCIO	36	3243 (166885)	30.93/0.05	3243 (166885)

Most current works [4], [42], [47], [48] that focus on improving feature representations for video-face clustering assume the number of main characters/clusters is known. In this work, for a fair comparison we follow the same protocols that are widely employed in the previous works [30], [47], [48] and train on BF episode 2, BBT episode 1, and ACCIO. We also use the same number of characters as previous methods [48], [47], however, it is important to note that we do not discard tracks/faces that are small or have large pose variation. When not mentioned otherwise, we use an updated version of face tracks released by [2] that incorporate several detectors to encompass all pan angles and in-plane rotations up to 45 degrees. Tracks are created via an online tracking-by-detection scheme with a particle filter.

We present a summary of our dataset in Table I, and also indicate the number of tracks (#TR) and frames (#FR) used in other works, showing that our data is indeed more challenging. Additionally, it is important to note that different characters have wide variations in number of tracks, indicated by the class balance largest class (LC) to smallest class (SC). Fig. 4 shows a few examples of difficult faces our methods have to deal with.

Evaluation metric. We use Clustering Accuracy (ACC) [48] also called Weighted Clustering Purity (WCP) [34] as the metric to evaluate the quality of clustering. As we compare methods providing equal numbers of clusters (number of main cast), ACC is a fair metric for comparison. ACC is

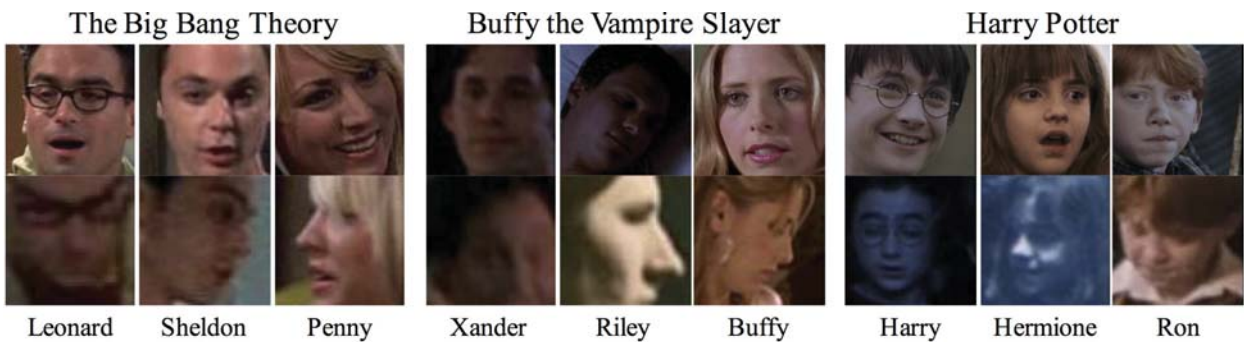

Fig. 4. Example images for a few characters from our dataset. We show one easy view and one difficult view. The extreme variation in illumination, pose, resolution, and attributes (spectacles) make the datasets challenging.

computed as

$$\mathrm{ACC} = \frac{1}{N}\sum_{c=1}^{|C|} n_c \cdot p_c\,, \tag{4}$$

where N is the total number of tracks in the video, 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 cast members, and in our case also the number of clusters. For ACCIO, we report BCubed Precision (P), Recall (R) and F-measure (F) for a fair comparison with the state-of-the-art. Figure 3 illustrates the steps for the testing evaluation.

B. Implementation Details

CNN. We adopt the VGG-2 face CNN [3], a ResNet50 model, pre-trained on MS-Celeb-1M [11] and fine-tuned on 3.31M face images of 9131 subjects (VGG2 data). Input RGB face images are resized to 224×224, and pushed through the CNN. We extract `pool5_7x7_s1` features, resulting in $\mathbf{x}_k^i \in \mathbb{R}^{2048}$.

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. 3), and we use the feature representations at $\mathbb{R}^{256}$ for clustering.

To train our Siamese network with track-level supervision (TSiam), we obtain about 102k positive and 204k negative frame pairs (for BBT-0101) by mining 2 positive and 4 negative pairs for each frame. For Self-supervised Siamese, we generate pairs as described in Sec. III, using $B = 1000, K = 64$. Higher values values of $B = 2000, 3000$, did not provide significant improvements.

The MLP is trained using the contrastive loss, and parameters are updated using Stochastic Gradient Descent (SGD) with a fixed learning rate of 10^{-3}. Since the labels are obtained automatically, overfitting is not a concern. We train our model until convergence (loss does not reduce significantly).

C. Clustering Performance and Generalization

Base features. We compare track- and frame-level performance of two popular deep face representations: VGG1 [21] and VGG2 [3]. Track-level results use mean-pool of frames. Number of clusters matches number of characters. Results are reported in Table II. Note that the differences are typically within 1% of each other indicating that the results in the following tables are not just due to having better CNNs trained with more data. We refer to VGG2 features as Base for the remainder of this paper.

TABLE II
CLUSTERING ACCURACY ON THE BASE FACE REPRESENTATIONS.

Dataset	Track Level		Frame Level	
	VGG1	VGG2	VGG1	VGG2
BBT-0101	0.916	**0.932**	0.938	**0.940**
BF-0502	0.831	**0.836**	0.901	**0.912**

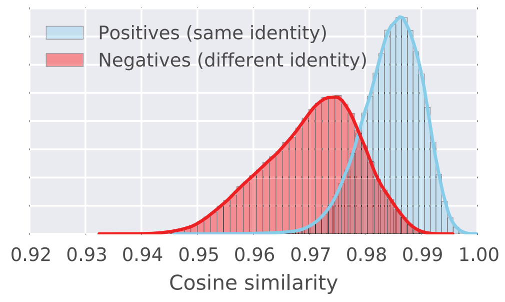

Fig. 5. Histograms of pairwise cosine similarity between tracks of same identity (pos) and different identity (neg) for BBT-0101.

Role of effective mining of +/- pairs. We emphasize that especially in light of CNN face representations, the features are very similar even across different identities, and thus positive and negative pairs need to be created properly to make the most out of them. Fig. 5 proves this point as we see a large overlap between the cosine similarity distributions of positive (same id) and negative (across id) track pairs on the base features.

TSiam, impact of singleton tracks. Previous work with video-level constraints [47], [48] and [5], ignore singleton (not co-occurring) tracks. In TSiam, we include negative pairs for singletons based on track distances. Table III shows that 30-70% tracks are singleton and ignoring them lowers accuracy by 4%. This confirms our hypothesis that incorporating negative pairs of singletons helps improve performance.

TABLE III
IGNORING SINGLETON TRACKS (AND POSSIBLY CHARACTERS) LEADS TO SIGNIFICANT PERFORMANCE DROP. ACCURACY ON TRACK-LEVEL CLUSTERING.

	TSiam		# Tracks		
Dataset	w/o Single [5]	Ours	Total	Single	Co-oc
BBT-0101	0.936	**0.964**	644	219	425
BF-0502	0.849	**0.893**	568	345	223

SSiam, comparison to pseudo-RF [45], [44]. In Pseudo-RF, all samples are treated independent of each other, there is no batch of data from which $2K$ pairs are chosen. A pair of samples closest in distance are chosen as positive, and farthest as negative. However, this usually corresponds to samples that already satisfy the loss margin, thus leading to small (possibly even 0) gradient updates. Table IV shows that SSiam that involves sorting a batch of queries is much more efficient over pseudo-RF as it has the potential to select harder positives and negatives. We see a consistent gain in performance, 3% for BBT-0101 and over 9% for BF-0502.

TABLE IV
COMPARISON OF *SSiam* WITH *pseudo-RF*.

Method	BBT-0101	BF-0502
Pseudo-RF	0.930	0.814
SSiam	**0.962**	**0.909**

TABLE V
CLUSTERING ACCURACY COMPUTED AT TRACK-LEVEL ON THE TRAINING EPISODES, WITH A COMPARISON TO ALL EVALUATED MODELS.

Train/Test	Base	TSiam	SSiam
BBT-0101	0.932	**0.964**	0.962
BF-0502	0.836	0.893	**0.909**

Performance on training videos. We report the clustering performance on training videos in Table V. Note that both TSiam and SSiam are trained in an unsupervised manner, or with automatically generated labels. Both of the proposed models SSiam and TSiam provide a large performance boost over the base VGG2 features on BBT and BF.

Generalization within series. In this experiment, we evaluate the generalization capability of our models. We train on one episode each of BBT-0101 and BF-0502 and evaluate on all other episodes of the same TV series. Table VI reports clustering accuracy. Both SSiam or TSiam perform similar (slightly lower/higher) to the base features, possibly due to overfitting.

TABLE VI
CLUSTERING ACCURACY COMPUTED AT TRACK-LEVEL ACROSS EPISODES WITHIN THE SAME TV SERIES. NUMBERS ARE AVERAGED ACROSS 5 TEST EPISODES.

Train	Test	Base	TSiam	SSiam
BBT-0101	BBT-01[02-06]	0.935	0.930	0.914
BF-0502	BF-05[01,03-06]	0.892	0.889	0.904

TABLE VII
CLUSTERING ACCURACY WHEN EVALUATING ACROSS VIDEO SERIES. EACH ROW INDICATES THAT THE MODEL WAS TRAINED ON ONE EPISODE OF BBT / BF, BUT EVALUATED ON ALL 6 EPISODES OF THE TWO SERIES.

	Trained	BBT-01[01-06]	BF-05[01-06]
TSiam	BBT-0101	0.936	0.875
	BF-0502	0.915	0.890
SSiam	BBT-0101	0.922	0.862
	BF-0502	0.883	0.905

Generalization across series. We further analyze our models by evaluating generalization across series. From Table VII, we wish to point out an interesting observation: TSiam and SSiam retain their discriminative power and can transfer to other series gently. As they learn to score similarity between pairs of faces, the actual identity and characters do not seem to matter much. For example, the drop when training TSiam on BBT-0101 and evaluating on BF is 0.890 (train on BF-0502) to 0.875.

Please note that the generalization experiments are presented here to explore the underlying properties of our models. If achieving high performance is the only goal, we assert that our models can be trained and evaluated on each video rapidly and fully automatically.

TABLE VIII
CLUSTERING ACCURACY WHEN EXTENDING TO ALL NAMED CHARACTERS WITHIN THE EPISODE. BBT-0101 HAS 5 MAIN AND 6 NAMED CHARACTERS. BF-0502 HAS 6 MAIN AND 12 NAMED CHARACTERS.

	BBT-0101		BF-0502	
	TSiam	SSiam	TSiam	SSiam
Main cast	0.964	0.962	0.893	0.909
All named cast	0.958	0.922	0.829	0.870

Generalization to unseen characters. In the ideal setting, we would like to cluster all characters appearing in an episode including (main, other named, unknown, and background). However, this is a very difficult setting, and in fact, disambiguating background characters is even hard for humans and there are no datasets that include such labels. For BBT and BF, we do however have all named characters labeled. Firstly, expanding the clustering experiment to include them drastically changes the class balance. For example, BF-

TABLE IX

COMPARISON TO STATE-OF-THE-ART. METRIC IS CLUSTERING ACCURACY (%) EVALUATED AT FRAME LEVEL. PLEASE NOTE THAT MANY PREVIOUS WORKS USE FEWER TRACKS (# OF FRAMES) (ALSO INDICATED IN TABLE I) MAKING THE TASK RELATIVELY EASIER. WE USE AN UPDATED VERSION OF FACE TRACKS PROVIDED BY [2].

Method	BBT-0101	BF-0502	Data Source BBT	Data Source BF
ULDML (ICCV '11) [4]	57.00	41.62	–	[4]
HMRF (CVPR '13) [42]	59.61	50.30	[26]	[7]
HMRF2 (ICCV '13) [41]	66.77	–	[26]	–
WBSLRR (ECCV '14) [43]	72.00	62.76	–	[7]
VDF (CVPR '17) [30]	89.62	87.46	[2]	[2]
Imp-Triplet (PacRim '16) [47]	96.00	–	[26]	–
JFAC (ECCV '16) [48]	–	92.13	–	[7]
Ours (with HAC)				
TSiam	**98.58**	**92.46**		
SSiam	**99.04**	90.87	[2]*	[2]*

0502 has 6 main and 12 named characters with class balance 36.2/5.0 to 40.8/0.1.

We present clustering accuracy for this setting in Table VIII. Both proposed methods show a drop in performance when extending to unseen characters. Note that the models have been trained on only the main characters data and tested on all (including unseen) characters. However, the drop is small when adding just 1 new character (BBT-0101) vs. introduction of 6 in BF-0502.

SSiam's performance scales well, probably since it is trained with a diverse set of pairs (dynamically generated during training) and can generalize to unseen characters.

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

BBT and BF. We compare our proposed methods (TSiam, and SSiam) with the state-of-the-art approaches in Table IX. We report clustering accuracy (%) on two videos: BBT-0101 and BF-0502. Historically, previous works have reported performance at a frame-level. We follow this for TSiam and SSiam.

Note that our evaluation uses 2-4 times larger number of frames than previous works [48], [47] making direct comparison hard. Specifically in BBT-0101 we have 41,220 frames while [47] uses 11,525 frames. Similarly, we use 39,263 frames for BF-0502 (vs. 17,337 [48]). Even though we cluster more frames and tracks (with more visual diversity), our approaches are comparable to or even better than the current results.

TSiam and SSiam are better than the improved triplet method [47] on BBT-0101. SSiam obtains 99.04% accuracy which is 3.04% higher performance in absolute gains. On BF-0502, TSiam performs the best with 92.46% which is 0.33% better than the JFAC [48].

ACCIO. We evaluate our methods on ACCIO dataset with 36 named characters, 3243 tracks, and 166885 faces. The largest to smallest cluster ratios are very skewed: 30.65% and 0.06%. In fact, half the characters correspond to less than 10% of all tracks. Table X presents the results when performing clustering to yield 36 clusters (equivalent to the number of characters). In addition, as in [48], Table XI (num. clusters = 40) shows that our discriminative methods are not affected much by this skew, and in fact improve performance by a significant margin over the state-of-the-art.

TABLE X

PERFORMANCE COMPARISON OF TSIAM AND SSIAM WITH JFAC [48] ON ACCIO.

Methods	#cluster=36 P	R	F
JFAC (ECCV '16) [48]	0.690	0.350	0.460
Ours (with HAC)			
TSiam	**0.749**	**0.382**	**0.506**
SSiam	**0.766**	**0.386**	**0.514**

TABLE XI

PERFORMANCE COMPARISON OF DIFFERENT METHODS ON THE ACCIO DATASET.

Methods	# clusters=40 P	R	F
DIFFRAC-DeepID2$^+$ (ICCV '11) [48]	0.557	0.213	0.301
WBSLRR-DeepID2$^+$ (ECCV '14) [48]	0.502	0.206	0.292
HMRF-DeepID2$^+$ (CVPR '13) [48]	0.599	0.23.0	0.332
JFAC (ECCV '16) [48]	0.711	0.352	0.471
Ours (with HAC)			
TSiam	**0.763**	**0.362**	**0.491**
SSiam	**0.777**	**0.371**	**0.502**

Computational complexity. Our models essentially consist of a few linear layers and are very fast to compute at inference time. In fact, training the SSiam for about 15 epochs on BBT-0101 requires less than 25 minutes (on a GTX 1080 GPU using the matconvnet framework [36]).

V. CONCLUSION

We proposed simple, unsupervised approaches for face clustering in videos, by distilling the identity factor from deep face representations. We showed that discriminative models can leverage dynamic generation of positive/negative constraints based on ordered face distances and do not have to only rely on track-level information that is typically used. Our proposed models are unsupervised (or use automatically generated labels) and can be trained and evaluated efficiently as they involve only a few matrix multiplications.

We conducted experiments on three challenging video datasets, comparing their differences in usage in past works. Overall, our models are very fast to train and evaluate and outperform the state-of-the-art while operating on datasets that contain more tracks with large changes in appearance.

Acknowledgements: This work is supported by the DFG, a German Research Foundation - funded PLUMCOT project.

References

[1] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Who's that Actor? Automatic Labelling of Actors in TV series starting from IMDB Images. In *ACCV*, 2016.

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

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

[6] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In *CVPR*, 2017.

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

[8] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5729–5738. IEEE, 2017.

[9] 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.

[10] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric Learning Approaches for Face Identification. In *ICCV*, 2009.

[11] 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.

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

[13] P. Hu and D. Ramanan. Finding Tiny Faces. In *CVPR*, 2017.

[14] 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. In *ECCV Workshop on Faces in Real-life Images*, 2008.

[15] 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.

[16] 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.

[17] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[18] Z. A.-H. Monica-Laura Haurilet, Makarand Tapaswi and R. Stiefelhagen. Naming TV Characters by Watching and Analyzing Dialogs. In *WACV*, 2016.

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

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

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

[22] 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.

[23] V. Ramanathan, A. Joulin, P. Liang, and L. Fei-Fei. Linking people in videos with "their" names using coreference resolution. In *ECCV*, 2014.

[24] A. Rohrbach, M. Rohrbach, S. Tang, S. J. Oh, and B. Schiele. Generating Descriptions with Grounded and Co-Referenced People. In *CVPR*, 2017.

[25] A. Rohrbach, A. Torabi, M. Rohrbach, N. Tandon, C. Pal, H. Larochelle, A. Courville, and B. Schiele. Movie Description. *IJCV*, 123(1):94–120, 2017.

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

[27] M. S. Sarfraz, A. Schumann, A. Eberle, and R. Stiefelhagen. A pose-sensitive embedding for person re-identification with expanded cross neighborhood re-ranking. In *CVPR*, 2018.

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

[29] V. Sharma, A. Diba, D. Neven, M. S. Brown, L. Van Gool, and R. Stiefelhagen. Classification driven dynamic image enhancement. In *CVPR*, 2018.

[30] 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*. IEEE, 2017.

[31] J. Sivic, M. Everingham, and A. Zisserman. "Who are you?" – Learning person specific classifiers from video. In *CVPR*, 2009.

[32] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *CVPR*, 2014.

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

[34] 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.

[35] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler. MovieQA: Understanding Stories in Movies through Question-Answering. In *CVPR*, 2016.

[36] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACMMM*, 2015.

[37] P. Vicol, M. Tapaswi, L. Castrejon, and S. Fidler. MovieGraphs: Towards Understanding Human-Centric Situations from Videos. *arXiv:1712.06761*, 2017.

[38] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

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

[40] L. Wolf, T. Hassner, and I. Maoz. Face Recognition in Unconstrained Videos with Matched Background Similarity. In *CVPR*, 2011.

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

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

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

[44] R. Yan, A. Hauptmann, and R. Jin. Multimedia search with pseudo-relevance feedback. In *International Conference on Image and Video Retrieval*, pages 238–247. Springer, 2003.

[45] R. Yan, A. G. Hauptmann, and R. Jin. Negative pseudo-relevance feedback in content-based video retrieval. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 343–346. ACM, 2003.

[46] L. Zhang, D. V. Kalashnikov, and S. Mehrotra. A unified framework for context assisted face clustering. In *ACMMM*. ACM.

[47] 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.

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

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