Video Face Clustering with Unknown Number of Clusters

Makarand Tapaswi[1,2,3] Marc T. Law[2,3,4] Sanja Fidler[2,3,4]
[1]Inria [2]University of Toronto [3]Vector Institute [4]NVIDIA
makarand.tapaswi@inria.fr, {makarand,law,fidler}@cs.toronto.edu
https://github.com/makarandtapaswi/BallClustering_ICCV2019

Abstract

Understanding videos such as TV series and movies requires analyzing who the characters are and what they are doing. We address the challenging problem of clustering face tracks based on their identity. Different from previous work in this area, we choose to operate in a realistic and difficult setting where: (i) the number of characters is not known a priori; and (ii) face tracks belonging to minor or background characters are not discarded.

To this end, we propose Ball Cluster Learning *(BCL), a supervised approach to carve the embedding space into balls of equal size, one for each cluster. The learned ball radius is easily translated to a stopping criterion for iterative merging algorithms. This gives BCL the ability to estimate the number of clusters as well as their assignment, achieving promising results on commonly used datasets. We also present a thorough discussion of how existing metric learning literature can be adapted for this task.*

1. Introduction

Characters are a central aspect of any story. While video streaming platforms such as Netflix provide the ability to find a movie based on metadata, searching a video collection to find the right clip when "Jack Sparrow first meets Will" requires analyzing the content of the video. Understanding characters also has a direct influence on important research such as video captioning [34, 35], question-answering [22, 44], studying social situations [45] and 4D effects [56].

Characters are often studied by analyzing face tracks (sequences of temporally related detections) in videos. A significant part of this analysis is *identification* - labeling face tracks with their names, and typically employs supervision from web images [1, 29], transcripts [3, 9], or even dialogs [7, 15]. We are interested in an equally popular alternative - *clustering* face tracks based on identity. Note that clustering is complementary to identification, and if achieved successfully can dramatically reduce the amount of required labeling effort. Clustering is also an interesting problem in itself as it can answer questions such as who are the main characters, or what are their social interaction groups.

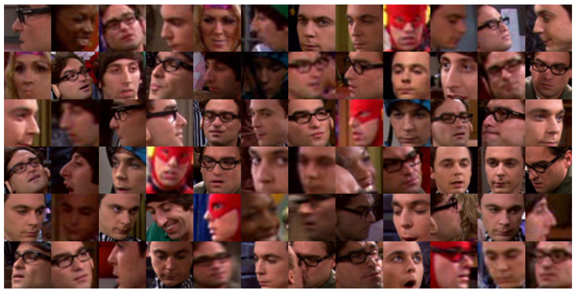


Figure 1. Video face clustering is a challenging problem that is further accentuated by a large portion of characters that play small roles. Can you guess how many characters are in this montage and which faces belong to them? See Fig. 2 for our solution.

While there exists a large body of work in video face clustering (*e.g.* [6, 18, 55]), most of it addresses a simplified setup where background characters[1] are ignored and the total number of characters is known. With recent advances in face representations [4], their application towards clustering [38], and the ability to learn cast-specific metrics by looking at overlapping face tracks [6], we encourage the community to address the challenging problem of estimating the number of characters and not ignoring background cast (see Fig. 1).

In this paper, we propose *Ball Cluster Learning* (BCL) - a supervised approach to carve the embedding space into equal-sized balls such that samples within a ball belong to one cluster. In particular, we formulate learning constraints that create such a space and show how the ball radius (also learned) can be associated with the stopping criterion for agglomerative clustering to estimate both the number of clusters and assignment (Sec. 3). We demonstrate BCL on video face clustering in a setup where we are unaware of the number of characters, and *all* face tracks, main character or otherwise, are included (Sec. 4). Thus, BCL is truly applicable to all videos as it does not place assumptions on availability of cast lists (to determine number of clusters) or track labels (to discard background characters). To evaluate our approach, we augment standard datasets used in video

[1]We consider three types of characters based on their roles: *primary* or *recurring* characters have major roles in several episodes; *secondary* or *minor* characters are named and play an important role in some episodes; and *background* or *unknown* (Unk) characters are unnamed and uncredited.

face clustering by resolving labels between all background characters. Our approach achieves promising results in estimating the number of clusters and the cluster assignment. We also present a thorough analysis of commonly used loss functions in verification (*e.g*. contrastive loss), compare them against BCL, and discuss how and when they may be suitable for clustering. To the best of our knowledge, BCL is the first approach that learns a threshold to estimate the number of clusters at test time. Code and data are available at Github.

2. Related Work

We survey work on identifying and clustering characters in videos. We also review metric learning approaches, some of which are adopted for clustering in this work (Sec. 3.4).

Character identification in videos. Over a decade earlier, the availability of transcripts (speaker names and dialogs) and their alignment with subtitles (dialogs and timestamps) opened exciting avenues for fully automatic identification [3, 9, 33, 40]. Dialog-based supervision proved to be a harder but scalable approach [7, 15]. Face track representations (*e.g*. [23, 30, 31, 48, 53]) further improved performance. Recently, the source of supervision moved towards web images from IMDb [1, 45] or image search [29], and a combination of modalities such as hair [29], speech [28] and clothing [42]. However, these advances are limited to identifying named characters and grouping all remaining characters in a common "*others*" label.

Video face clustering. A common idea adopted by many clustering approaches is to use unsupervised constraints that arise from the video to learn cast-specific metrics [6]. Pairs of face images within tracks are considered similar; and faces that appear simultaneously in the video are assumed dissimilar. These constraints are used with Hidden Markov Random Fields [49, 50], or to learn low-rank block-sparse representations [51]. They also see use in conjunction with the video editing structure (shots, threads, and scenes) [43]. The constraints are also used to fine-tune CNNs and learn clustering jointly [55], or to learn an embedding using an improved triplet loss [54].

Ignoring tracks, metrics are learned by ranking a batch of frames and creating hard positive and negative pairs [39]. However, all of the above methods require knowledge of the number of clusters K that is difficult to estimate beforehand; and only consider primary characters (tracks for background characters are ignored). In online face clustering spatio-temporal constraints along with CNN representations are used to assign a new track to existing or new cluster [19]. However, only primary characters in the video are targeted. Recently, an end-to-end detection and clustering approach considers false positive and missed detections [18].

In this paper, we consider a setup where *all* face tracks are to be clustered into an *unknown* number of characters.

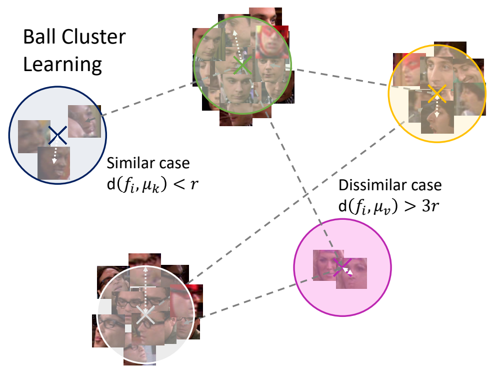


Figure 2. The face tracks in Fig. 1 can be clustered into 5 characters. *Ball Cluster Learning* carves the feature space into balls of equal radius. The number of samples in the cluster does not affect the ball radius or minimum separation to other balls.

Metric learning. Early examples of state-of-the-art approaches in face recognition adopt metric learning [5, 12]. The learning task is often posed as verification - are two face images of the same person. The main difficulty is ensuring that the model generalizes to test images of people that are not seen during training. When working with videos, this is mitigated by obtaining positive and negative pairs through tracking and training on the video itself.

Other loss functions involving *triplets* [37] are also proposed for face verification [36]. While FaceNet [36] claims to be good at clustering, performance is only evaluated qualitatively. Training with the triplet loss is cumbersome as it requires creation of all possible triplets that is computationally expensive. Sampling strategies become crucial to ensure fast convergence while avoiding degenerate solutions.

Centroid-based losses [20, 21, 26, 41] are also proposed for face verification [46]. Here, models are trained to make each sample closer to the representative of its category than to the representative of any other category resulting in samples of the same category being grouped into a single cluster. These methods are ideal when the number of clusters is known at test time. However, there is neither a constraint on the size/radius of the clusters, nor is there a threshold to predict whether two samples are similar, *e.g*. NormFace [46] trains a classifier to determine whether pairs are similar.

Joint Unsupervised LEarning (JULE) [52] learns representations while performing hierarchical clustering. However, as JULE has to learn both the cluster assignments and representations, it is hard to scale [10, 11, 27] and its computational cost and memory complexity are extremely high. Moreover, JULE is only tested in cases where the number of clusters is known at test time.

We propose a model that groups similar samples into non-overlapping balls. The radius of the ball clusters is learned and is directly related to the threshold used as a stopping criterion of our clustering algorithm (Sec. 3.3). In addition, our training algorithm has very low algorithmic complexity: it is linear in the batch size and in the number of clusters.

3. Ball Cluster Learning (BCL)

The main goal of our supervised learning approach is to carve the embedding space into balls with a shared but trainable radius for each cluster, while simultaneously creating a well-defined separation between balls of different cluster labels (Fig. 2). We first define the constraints that achieve the above goal (Sec. 3.1), formulate the learning problem with loss functions (Sec. 3.2), and then explain how to perform clustering at test time (Sec. 3.3). Finally, we review several losses from the metric learning literature that may be suitable for clustering (Sec. 3.4).

Notation. Let $B = \{(x_i, y_i)\}_{i=1}^N, y_i \in \{1, \cdots, K\}$ be a mini-batch containing N samples that we wish to group into K clusters. We learn a mapping $\varphi_\theta : \mathcal{X} \to \mathcal{F}$ (*e.g.* a neural network) parameterized by θ. The embedding space can either be $\mathcal{F} = \mathbb{R}^D$ or $\mathcal{F} = \mathbb{S}^{D-1} = \{\mathbf{f} \in \mathbb{R}^D : \|\mathbf{f}\|_2 = 1\}$ inspired by recent work [46] that shows benefits of ℓ_2 normalization for face recognition. The i-th sample is represented by the output of the mapping $\mathbf{f}_i = \varphi_\theta(x_i)$.

"Ball" terminology: We define samples of our clusters as lying in a ball. However, when $\mathcal{F} = \mathbb{S}^{D-1}$, our clusters technically lie on the hypersurface of *hyperspherical cones*.

3.1. Constraints

We analyze similar and dissimilar samples separately. Let $\mathcal{C}_k$ be the k-th set of similar samples (*i.e.* samples x_i that satisfy $y_i = k$). A pair of samples (x_i, x_j) is similar iff $y_i = y_j$, and it is dissimiliar otherwise.

Similar case. We define $\boldsymbol{\mu}_k \in \mathcal{F}$ as the centroid of all the samples in $\mathcal{C}_k$ w.r.t. the squared Euclidean distance:

$$\boldsymbol{\mu}_k = \frac{1}{\nu_\mathcal{F}} \sum_{x_i \in \mathcal{C}_k} \mathbf{f}_i \in \arg\min_{\boldsymbol{\mu} \in \mathcal{F}} \sum_{x_i \in \mathcal{C}_k} \mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}), \qquad (1)$$

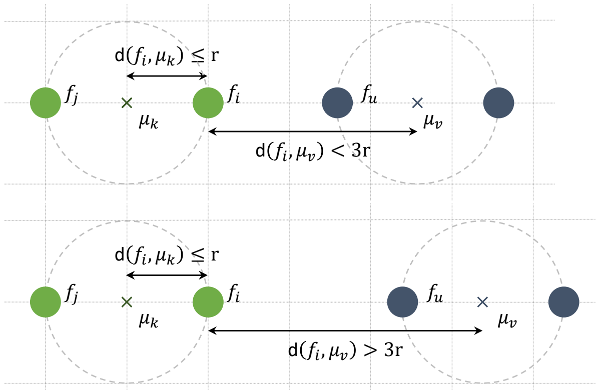
where $\mathsf{d} : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ is the Euclidean distance (*i.e.* $\mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_k) = \|\mathbf{f}_i - \boldsymbol{\mu}_k\|_2^2$). The factor $\nu_\mathcal{F}$ is N if $\mathcal{F} = \mathbb{R}^D$, or $\|\sum_{x_i \in \mathcal{C}_k} \mathbf{f}_i\|_2$ (we assume for simplicity that it is non-zero) if $\mathcal{F} = \mathbb{S}^{D-1}$ since $\boldsymbol{\mu}_k$ is constrained to be in $\mathcal{F}$ (on the unit-norm hypersphere). For any sample x_i that belongs to $\mathcal{C}_k$, we would like to learn a representation $\mathbf{f}_i$ such that its squared distance to $\boldsymbol{\mu}_k$ is smaller than some learned threshold $b > 0$. Our goal is to satisfy the constraints:

$$\forall x_i \in \mathcal{C}_k, \; \mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_k) \leq b. \qquad (2)$$

Note that b is trained as a model parameter. We consider that the *radius* r of the balls is $r = \sqrt{b} \geq \max_{x_i \in \mathcal{C}_k} \mathsf{d}(\mathbf{f}_i, \boldsymbol{\mu}_k)$. By using the triangle inequality, similar samples satisfy the following constraint:

$$\forall x_i \in \mathcal{C}_k, x_j \in \mathcal{C}_k, \; \mathsf{d}(\mathbf{f}_i, \mathbf{f}_j) \leq 2r = 2\sqrt{b}. \qquad (3)$$

We choose $2r$ as the threshold to determine whether two samples are similar or not.



$\mathsf{d}(f_i, \mu_k) \leq$ r
f_j
f_i
f_u
μ_k
μ_v
$\mathsf{d}(f_i, \mu_v) <$ 3r
$\mathsf{d}(f_i, \mu_k) \leq$ r
f_j
f_i
f_u
μ_k
μ_v
$\mathsf{d}(f_i, \mu_v) >$ 3r

Figure 3. Consider a toy scenario with 4 samples (2 green, 2 blue in each cluster) in $\mathcal{F} = \mathbb{R}^2$. We illustrate the constraints derived in Eq. (2) and Eq. (4). Each grid square in this 2D-space corresponds to the ball radius r. **Top**: When $\mathsf{d}(f_i, \mu_v) < 3r$, we see that $\mathbf{f}_i$ and $\mathbf{f}_u$ are the closest samples and will be merged by hierarchical agglomerative clustering (HAC) in the first iteration. **Bottom**: When $\mathsf{d}(f_i, \mu_v) > 3r$, the distance between $\mathbf{f}_i$ and $\mathbf{f}_u$ is larger than either the green or blue pair of samples. Additionally, by adopting the max linkage and choosing the stopping criterion for HAC as $\tau = 2r$ (in Euclidean distance), iterative merging stops after the green and blue samples are grouped. Best seen in color.

Dissimilar case. From the above discussion, two dissimilar samples (x_i, x_u) should satisfy $\mathsf{d}(\mathbf{f}_i, \mathbf{f}_u) > 2r$. Furthermore, as the distance between $x_u \in \mathcal{C}_v$ and its centroid $\boldsymbol{\mu}_v$ is at most r, the Euclidean distance between $\mathbf{f}_i$ and $\boldsymbol{\mu}_v$ should be greater than $3r$ to ensure that all the clusters are separated (see Fig. 3). This implies $\mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_v) > (3r)^2 = 9b$. We denote $\gamma = 9b + \varepsilon$ where $\varepsilon \geq 0$ is a small fixed margin and formulate the constraint:

$$\forall x_i \in \mathcal{C}_k \neq \mathcal{C}_v, \; \mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_v) \geq \gamma. \qquad (4)$$

A major difference to existing metric learning approaches is that we enforce an upper bound on the distance between each example and its desired centroid (Eq. (2)), which in turn enforces samples of each cluster to be within a ball of radius r. We also enforce different clusters to be separated by a margin that is a function of the radius (Eq. (4)).

Computational complexity. Formulating our constraints on the distances between samples and cluster centroids significantly lowers the number of computations in contrast to pairwise distances that yield quadratic constraints.

A fixed radius for all balls allows us to use it as a threshold to delimit clusters. In addition, it has the potential to address the long-tail since each identity gets the same volume of embedding-space, agnostic to the number of tracks.

3.2. Problem Formulation

Based on the desired constraints in Sec. 3.1, we now formulate an optimization problem that tries to satisfy them. Our goal is to learn the squared radius $b > 0$ of the cluster balls and parameters θ of the model φ_θ that minimize the objective problem $\mathcal{L}_{ball}$ defined as the sum of the two losses:

$$\mathcal{L}_{ball} = \alpha \mathcal{L}_{sim} + \mathcal{L}_{dis}, \tag{5}$$

where $\alpha \geq 0$ is a hyperparameter to balance the losses. We present details of the loss terms in the following.

The goal of the loss $\mathcal{L}_{sim}$ is to satisfy the similar pairs constraint in Eq. (2), and is formulated as:

$$\mathcal{L}_{sim} = \frac{1}{N} \sum_{x_i \in \mathcal{C}_k} \left[\mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_k) - b\right]_+, \tag{6}$$

where $[x]_+ = \max(0, x)$. In the context of metric learning, this often corresponds to the *positive* loss as it brings together samples of the same cluster.

The goal of the dissimilar loss $\mathcal{L}_{dis}$ is to satisfy dissimilar pairs constraints in Eq. (4) and is formulated as:

$$\mathcal{L}_{dis} = \frac{1}{N} \sum_{x_i \in \mathcal{C}_k} \max_{v \neq k} \left[\gamma - \mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_v)\right]_+. \tag{7}$$

This loss aims to push away from the *most* offending cluster centroid by employing $\max_{v \neq k}$, and is equivalent to hard negative mining in metric learning [36].

3.3. Clustering Algorithm

We now describe how to perform clustering and predict the number of clusters on some given (test) dataset. Recall that we are interested in solving problems where the number of clusters is unknown at test time.

As explained in Sec. 3.1, our constraints are formulated so that similar samples should satisfy $\mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_j) \leq 4b$ and dissimilar samples should have larger distances. We apply a clustering algorithm which groups pairs of examples that satisfy those constraints into a single cluster.

Even when the number of clusters is known, finding the partitions that minimize some clustering energy function is an NP-hard problem [2]. Thus, methods that find a good local minimum solution with reasonable complexity are often used (*e.g.* K-means [24]). For this reason, we adopt the Hierarchical Agglomerative Clustering (HAC) method [8]: each sample starts in its own cluster, and pairs of clusters are iteratively merged until some specific stopping criterion. In the context of *complete linkage*, two clusters U and V are merged into a single cluster if they minimize:

$$\ell_{complete}(U, V) = \max_{x_u \in U, x_v \in V} \mathsf{d}^2(\mathbf{f}_u, \mathbf{f}_v). \tag{8}$$

Let us denote $\tau > 0$ the threshold chosen such that the HAC algorithm stops when there are *no* two clusters U and V that satisfy $\ell_{complete}(U, V) \leq \tau$. Once the HAC algorithm stops, all the examples assigned to a same cluster U satisfy $\forall x_a \in U, x_b \in U,\ \mathsf{d}^2(\mathbf{f}_a, \mathbf{f}_b) \leq \tau$ by definition of the complete linkage. Thus, we choose $\tau = 4b$. With this value of τ, when the (ideal) global minimum of Eq. (5) is obtained, applying the HAC with linkage in Eq. (8) groups similar examples in the same clusters and separates dissimilar examples since both Eq. (3) and Eq. (4) are satisfied.

3.4. Extending related work to our task

We compare BCL with various metric learning approaches commonly used in face verification tasks.

Triplet Loss [36] tries to preserve the order of distances between similar pairs (x_i, x_j) and dissimilar pairs (x_i, x_u): $\mathcal{L}_{triplet} = \sum_{\substack{y_i = y_j \\ y_i \neq y_u}} \left[\mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_j) - \mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_u) + m\right]_+$. While the loss ensures that positives are closer than negatives by a margin m, there is no constraint on the distance between positive samples. Thus, we are unable to directly use the margin as a threshold for stopping the HAC algorithm.

Threshold strategy. We choose a threshold based on a validation set: we apply the HAC algorithm and pick the threshold that predicts the ground truth number of validation clusters. Even for the baselines that learn a threshold, this strategy worked better than using the learned threshold. Thus, we report scores using this strategy for all baselines.

Contrastive Loss [5] considers pairwise constraints. For any pair of samples (x_i, x_j), and $y_{ij} = 1$ when they are similar and 0 otherwise, the contrastive loss between them is $\mathcal{L}_{cont} = \frac{y_{ij}}{2} \mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_j) + \frac{1}{2}(1 - y_{ij}) \left[m - \mathsf{d}(\mathbf{f}_i, \mathbf{f}_j)\right]_+^2$. This aims to make dissimilar samples at least m distance apart. While m is usually a fixed hyperparameter, we treat it as a trainable value in the same way as b in BCL.

Logistic Discriminant Metric Learning (LDML) [12] maps distances to a probability score via the sigmoid function $\sigma(\cdot)$. It can be written as $p_{ij} = p(y_i = y_j | \mathbf{f}_i, \mathbf{f}_j, \beta) = \sigma\left(\beta - \mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_j)\right)$, where β is a threshold trained to distinguish similar from dissimilar pairs. The loss is formulated as binary cross-entropy and is minimized: $\mathcal{L}_{ldml} = -\sum_{y_i = y_j} \log p_{ij} - \sum_{y_i \neq y_j} \log(1 - p_{ij})$.

Prototypical Networks [41] If both Eq. (2) and Eq. (4) are satisfied, the following order is obtained: $\forall x_i \in \mathcal{C}_k$, $v \neq k$, $\mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_k) - b \leq \mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_v) - \gamma$. To satisfy this relative constraint, we formulate the cross entropy loss:

$$\mathcal{L}_{proto} = -\frac{1}{N} \sum_{i \in \mathcal{C}_k} \frac{1}{|\mathcal{C}_k|} \log\left(p(y_i = k | \mathbf{f}_i)\right) \tag{9}$$

where $p(y_i = k | \mathbf{f}_i)$ is the posterior probability:

$$\frac{\exp(-\mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_k) + b)}{\exp(-\mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_k) + b) + \sum_{v \neq k} \exp(-\mathsf{d}^2(\mathbf{f}_i, \boldsymbol{\mu}_v) + \gamma)}. \tag{10}$$

The vanilla Prototypical Networks [41] correspond to $\mathcal{L}_{proto}$ when $b = \gamma = 0$. NormFace [46] is similar to [41], with one main difference that representations are ℓ_2-normalized. We report scores for $b = \gamma = 0$ since we experimentally found that it returns the best results with our threshold strategy.

4. Video Face Track Clustering with BCL

We discuss how BCL can be applied to face track clustering. Each sample represents a face track and is associated with a specific identity. Our goal is to create clusters such that tracks with the same identity are grouped together.

During training, we create mini-batches by uniformly sampling a fixed number of tracks. As the training data contains several identities with very few (1-2) tracks, and many others with hundreds or thousands of tracks, uniform random sampling preserves the skewed distribution of cluster membership within the mini-batch (see Fig. 4). From each track, we randomly choose one face image (which serves as data augmentation) and use a pre-trained and fixed CNN to extract a face representation x_i. We will refer to this as the base CNN representation. At test time, we average the base representations of all face images in the track and apply HAC after computing embeddings. This makes the track feature robust, while keeping it in the same space as the training samples. Other track-level representations such as [23, 30, 48, 53] are out of scope of this work.

Base CNN is a 50-layer ResNet [16] with squeeze-and-excitation (SE) blocks [17]. The model is pre-trained on the MS-1M dataset [13], and fine-tuned on the VGGFace2 dataset [4] with cross-entropy loss to predict over 8000 identities. We obtain features in $\mathbb{R}^{256}$ from the last layer (before the classifier). This model is named SE-ResNet50-256.[2] We will show that our methods work equally well when using a different base CNN. We do not fine-tune the CNN.

Model. Our model φ_θ is a stack of 4 linear layers with ReLU non-linearity (MLP) in between and is applied on top of the base CNN representation. When not stated otherwise, the hidden layers have 256, 128, and 64 nodes, and the final embedding dimension $D = 64$.

Our constraints require that $b > 0$. To this end, we use the `softplus` operator defined as $b = \log(1 + e^{\hat{b}})$, and train $\hat{b} \in \mathbb{R}$ as a model parameter. We balance the similar and dissimilar losses with $\alpha = 4$ based on the performance on a validation set.

Learning. We find that the loss for our model can be reduced dramatically (to ϵ) by mapping all samples to the same point and learning the squared radius to be close to 0. We prevent the learning process from reducing the radius to 0, by freezing it for the first 5 epochs. Subsequently, the loss parameter $\hat{b}$ updates slowly, 0.1 times the learning rate used for MLP weights. We employ SGD with 0.9 momentum at a learning rate of 0.003, and a 0.9× decay every 10 epochs to update the weights of the MLP. We use mini-batches of 2000 samples (tracks) when not stated otherwise.

[2]We use the pre-trained PyTorch model provided by https://github.com/ox-vgg/vgg_face2.

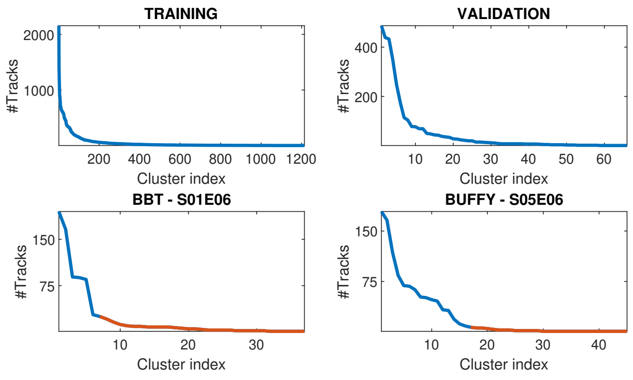


Figure 4. Number of tracks in clusters. Orange lines in BBT and BUFFY indicate track counts for unknown/background characters.

5. Evaluation

We first present the datasets and metrics used in our experiments. Then, we perform ablation studies on the validation split and finally show and discuss our results on the test set.

5.1. Datasets and metrics

We use face tracks from several movies and TV series as part of training and evaluation.

Train and validation splits consist of face tracks and ground-truth identity labels provided for 51 movies from the MovieGraphs dataset [45]. Like most previous work, the dataset contains annotations for main characters only, and does not disambiguate between background characters. Nevertheless, it is suitable for training, and we obtain 65,076 tracks that are mapped to 1,280 unique actors using IMDb. As Fig. 4 indicates, many actors have few (and even one) tracks making the training distribution similar to test.

We reserve 5% of the actors for validation and ensure that actors appearing in the test data are *not* seen during train or validation. This results in 61,774 tracks (1,214 actors) for the train split and 3,302 tracks (66 actors) for validation.

Test split. Our evaluation is on six episodes each of two TV series: The Big Bang Theory (BBT) and Buffy the Vampire Slayer (BUFFY). Both have been actively used in person identification and clustering [3, 18, 39, 55].[3] We wish to emphasize that most previous approaches for face clustering only consider primary (recurring) characters and know the number of clusters. We adopt a more practical setting where the number of characters is *not* known, and tracks for *all* (secondary as well as background characters) are included. We painstakingly resolve faces of background characters and assign unique identifiers to them. This is difficult even for humans, but is achieved through a combination of facial (hair) and non-facial (clothing, spatial location) cues. Finally, we also evaluate on *combined tracks* from several episodes

[3]We use an updated version of the tracks that does not discard background characters and small/profile faces.

	One cluster	N clusters	Base K known	Ours $\tau = 4b$
#Cl	1	3302	66	69
NMI	0	0	68.91	77.09
WCP	14.75	100.0	76.53	85.65

Table 1. Performance on the validation set showing the impact of putting all samples in the same or their own clusters. We also present performance of base CNN features when the number of clusters is known. The validation set has 66 ground-truth clusters.

Base CNN	Dim	#P	#Cl	NMI	WCP
SE-ResNet50-256	256	26.5M	69	**77.09**	85.65
ResNet50	2048	41.1M	80	76.74	**87.67**

Table 2. Performance on validation set for different base CNN models. The 4-layer MLP used for the ResNet50 model is 2048→512→256→128→64.

Dim	256	128	64	32	16	8
#P	263K	132K	111K	109K	108K	107K
#Cl	45	62	69	**68**	72	29
NMI	76.68	76.89	**77.09**	75.48	68.72	47.62
WCP	81.98	85.52	85.65	**85.89**	79.35	50.79

Table 3. Performance on the validation set for varying embedding dimension. #P indicates the number of parameters in the MLPs.

(and series) to mimic additional challenging scenarios. Each face in Fig. 1 represents a different track from BBT-S1E1.

Metrics. We adopt three primary metrics to evaluate performance: (i) #Cl: is the number of predicted clusters, and should be close to the ground-truth number of identities.
(ii) Normalized Mutual Information (NMI) [25]: for a given set of class labels Y and cluster predictions C, NMI is calculated as $2I(Y;C)/(H(Y)+H(C))$, where $H(\cdot)$ is entropy, and $I(\cdot;\cdot)$ is mutual information. NMI is a balanced metric and scores 0 when all samples are either in one cluster or their own clusters (see Table 1). All model checkpoints are chosen to maximize NMI on the validation set.
(iii) Weighted Clustering Purity (WCP) [43]: Also called clustering accuracy [55], WCP combines purity (fraction of samples that belong to the same class) of the clustering by weighting with the number of samples in the cluster.

5.2. Ablation Studies

We make several design choices that are motivated in the following. Table 1 provides insights into the validation split by showing the extreme ends of the clustering. We also demonstrate that our model outperforms the base CNN descriptors even when the base model is assumed to know the actual number of clusters (Base K known). Throughout this section, the ideal number of clusters on validation is 66.

Base CNN model. We demonstrate that the choice of the CNN model does not directly influence performance. In fact, our base model SE-ResNet50-256 has an output space $x_i \in \mathbb{R}^{256}$ while the ResNet50 base model produces $x_i \in \mathbb{R}^{2048}$. Table 2 shows that both models exhibit similar performance.

	Batch size	500	1000	2000	4000
#Cl	in batch	220	330	450	600
(approx)	> 5 samples	15	45	90	150
Performance	#Cl	88	91	**69**	29
on	NMI	72.13	74.63	**77.09**	76.55
Validation	WCP	83.77	**87.28**	85.65	79.68

Table 4. Ablation studies on mini-batch size. The first half of the table reports the number of the clusters in the batch, and those that have more than 5 samples. In the second half, we report performance on validation.

Embedding dimension. From the results in Table 3, we can infer that choosing too small an embedding dimension reduces performance dramatically. However, setting $D \geq 32$ achieves comparable similar performance.

Batch size. Our model learns to satisfy the constraints and perform clustering on data within each mini-batch. When batches are small (*e.g.* less than 50) it is likely that most clusters have only one sample. This automatically satisfies positive constraints and gradients are 0. Making the mini-batches too large incurs a computational cost and reduces the number of parameter updates; the model requires many more epochs to reach a similar performance. In Table 4, we first report the approximate number of clusters in a batch, and the number of clusters with more than 5 samples that can be assumed to have meaningful centroids (> 5). Notice how this can be quite small even for a batch of 500 samples. We find that a batch of 2000 samples is a decent trade off that achieves good performance.

ℓ_2 normalized embeddings $\mathbf{f}_i$ (*i.e.* $\mathcal{F} = \mathbb{S}^{D-1}$) help improve performance and are used in our model. Without the ℓ_2 normalization, our method creates 71 clusters with NMI: 74.57 and WCP: 83.07 (∼2.5% lower).

Single face image at training. We average base CNN representations of face images in a track at test time, while at training, we feed single images. This seems conflicting. However, when we choose to average a random half subset of track images during training, the performance is much worse with 124 clusters and a 7% lower NMI (absolute).

Complexity. During BCL training, each sample is compared only to the centers of clusters/categories. Thus, BCL has complexity linear in the number of samples and number of categories. This is much lower than most baselines that compare samples with samples. We report the wall clock time (average of 3 runs) taken to compute various losses for one epoch – Prototypical: 12.3s; Contrastive: 15.5s; LDML: 15.5s; Triplet: 50.8s; and BCL: 9.9s.

5.3. Evaluation on Test Set

We present statistics of the test set episodes in Table 5, rows 1-6. In particular, note how some episodes have a large

		BBT						BUFFY						BBT	BUFFY	BOTH
		S1E1	S1E2	S1E3	S1E4	S1E5	S1E6	S5E1	S5E2	S5E3	S5E4	S5E5	S5E6	6 ep.	6 ep.	12 ep.
1	#Ch	8	6	26	28	25	37	13	22	15	32	38	45	103	109	212
2	#Named Ch	6	5	7	8	6	6	11	12	13	14	13	17	11	26	37
3	#Unk Ch	2	1	19	20	19	31	2	10	2	18	25	28	92	83	175
4	#T	656	615	660	613	524	840	795	993	1194	898	840	1112	3908	5832	9740
5	#Named T	647	613	562	568	463	651	786	866	1185	852	733	1055	3504	5477	8981
6	#Unk T	9	2	98	45	61	189	9	127	9	46	107	57	404	355	759
CrossEntropy Loss																
7	#Cl	23	24	37	38	**26**	**37**	43	39	58	56	49	**52**	130	194	323
8	NMI	67.42	64.57	64.87	69.73	72.52	63.02	63.14	59.58	59.07	61.44	60.52	61.78	57.91	55.58	60.33
9	WCP	96.80	90.57	86.36	87.93	86.83	73.81	86.67	69.99	78.48	79.73	78.10	70.68	86.59	74.57	76.05
Logistic Discriminant Metric Learning [12]																
10	#Cl	14	15	19	25	20	30	25	31	28	**29**	31	30	62	82	116
11	NMI	66.42	53.21	66.59	65.33	73.06	55.77	63.57	53.38	58.54	59.52	52.68	56.50	53.15	50.65	51.97
12	WCP	92.23	82.28	74.70	79.61	86.07	62.86	83.02	58.71	71.69	67.59	59.17	59.80	74.33	61.01	58.14
Contrastive Loss [5]																
13	#Cl	14	13	17	22	19	32	22	30	26	**29**	29	27	60	71	110
14	NMI	62.45	63.69	61.77	65.55	71.38	55.68	61.00	53.94	58.15	53.42	53.59	52.01	58.94	49.15	51.53
15	WCP	90.70	86.99	64.85	76.35	75.57	65.95	77.86	56.09	67.59	60.80	62.02	50.36	77.53	57.30	48.81
Triplet Loss [36]																
16	#Cl	**9**	12	15	16	13	23	23	**24**	25	22	23	26	51	73	111
17	NMI	88.13	71.23	79.83	76.71	85.77	69.34	73.60	64.22	66.24	63.61	67.88	65.49	67.94	59.74	64.79
18	WCP	98.48	95.28	90.15	83.69	89.69	76.67	88.68	67.77	81.99	69.71	77.74	68.71	87.31	68.69	71.34
Prototypical Loss [41]																
19	#Cl	12	15	**22**	**28**	18	41	32	32	20	**35**	**40**	36	**87**	**123**	**197**
20	NMI	82.29	75.12	83.74	**80.29**	91.36	**74.32**	74.23	71.02	76.16	70.46	76.63	73.47	70.43	64.99	70.23
21	WCP	96.19	97.56	**93.79**	**91.03**	**94.66**	**86.67**	90.19	**80.16**	82.50	81.85	88.69	78.24	**90.56**	80.52	**82.80**
Ball Cluster Learning (Ours)																
22	#Cl	**7**	**8**	16	18	11	23	**17**	16	**18**	22	26	22	47	71	116
23	NMI	**95.81**	**87.25**	**88.38**	76.59	**92.21**	74.19	**81.78**	**77.60**	**77.64**	**78.13**	**79.72**	**78.15**	**73.22**	**71.23**	**75.32**
24	WCP	**98.63**	**98.54**	90.61	86.95	89.12	81.07	**92.08**	79.76	**84.00**	**84.97**	**89.05**	**80.58**	89.36	**83.62**	**82.81**
Ball Cluster Learning (Ours) + Fine-tune with automatically obtained positive/negative pairs																
25	#Cl	9	8	24	24	21	36	23	27	25	36	38	40	69	78	126
26	NMI	97.34	97.80	94.00	90.42	95.83	83.32	84.59	82.59	78.76	77.58	81.71	79.51	88.26	77.05	80.42
27	WCP	99.24	99.67	96.06	96.08	97.71	90.36	94.97	88.12	90.28	86.19	90.24	88.13	94.11	86.64	85.84

Table 5. Clustering performance on episodes of the test set. S1E2 corresponds to season 1 and episode 2. The last three columns show results on datasets created by combining tracks from several episodes. *Name* refers to primary and secondary named characters; *Unk* refers to background characters; #Ch is number of characters; #T is number of tracks; and #Cl is number of predicted clusters and should be close to the number of characters (row 1). Read this table by looking at each column, and seeing which method is able to predict the number of clusters and has high NMI and WCP scores.

		BBT 6 ep.	BUFFY 6 ep.	ALL 12 ep.
BCL	K-means	60.5 (92.0)	66.7 (**87.3**)	68.7 (**88.0**)
BCL	HAC	**70.6** (**93.0**)	**69.1** (85.3)	**72.5** (86.2)
PRO	K-means	60.7 (91.3)	64.5 (85.4)	66.8 (85.6)
PRO	HAC	68.3 (91.1)	65.8 (80.0)	70.3 (83.3)

Table 6. NMI and WCP performances of our approach (BCL) and prototypical loss (PRO) when the number of clusters is known.

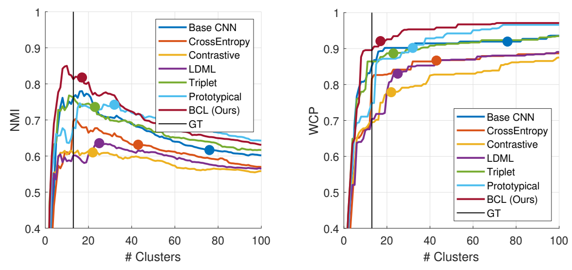


Figure 5. NMI and WCP vs. number of clusters on Buffy S5E1. Circles indicate operating points (*i.e.* number of predicted clusters for the methods), our method uses the HAC threshold 4b, while all others are using the threshold tuned to give 66 clusters on the validation set. Best seen in color.

number of background characters (*e.g.* 31 for BBT-S1E6) while others do not (*e.g.* 2 for BUFFY-S5E1). The last three columns refer to larger and arguably harder[4] datasets created by combining tracks of several episodes. In addition to Table 5, we also plot NMI and WCP vs. number of clusters in Fig. 5. Below, we discuss each loss in detail.

CrossEntropy loss (CE). CE can be seen as a (logistic) regression problem that merges all the similar examples to a single one-hot vector. We believe this is a reason why base CNN representations are quite good at clustering (blue curve in Fig. 5) when the number of characters is known. However, using a threshold on validation to choose an operating point results in much lower performance (76 clusters instead of 13). To further test this hypothesis, we train an MLP φ_θ to classify among our training set of actors, and use activations from the last layer as embeddings. The orange curve in Fig. 5 is lower than the base model (blue) indicating that training with more characters may have helped the base model. Nevertheless, choosing an operating point is difficult. Results in Table 5

[4]The combined episode datasets have many more background characters, while tracks from recurring characters collapse onto each other. This further skews cluster membership, with the largest cluster being several thousand tracks, and the smallest still having one track.

rows 7-9 show that the CE over-clusters (create many more clusters than GT). Directly using base CNN representations also results in many clusters (see supplementary).

Verification losses. Next, we analyze LDML, contrastive, and triplet losses (Table 5 rows 10-18). While these losses are often used to perform clustering, they are not designed for it [47]. We see two major features: (i) unlike BCL, estimating the number of clusters is not a built-in feature and requires choosing a threshold on the validation set that may be unreliable; and (ii) early errors in the iterative merging can really harm the overall composition. We observe that triplet loss consistently achieves higher NMI and better estimates for number of clusters than contrastive and LDML.

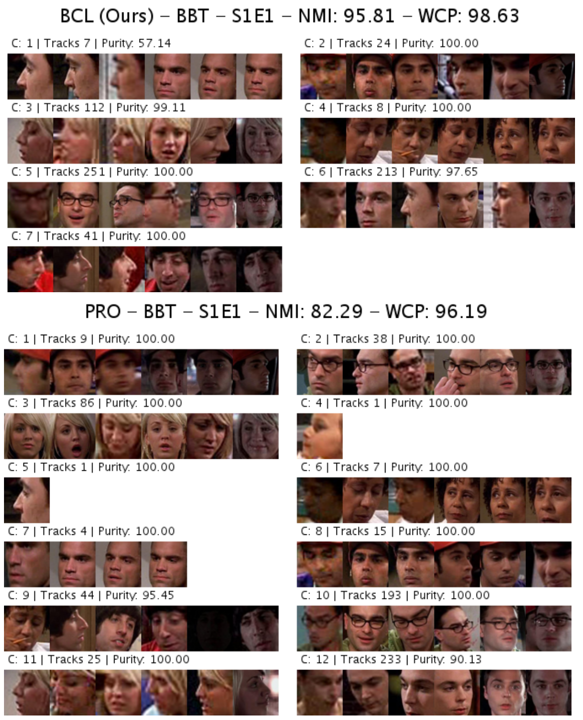
Prototypical loss (PRO) vs. BCL. Similar to verification losses (above), PRO works best when the number of clusters is known (*e.g.* for few-shot learning). The loss has strong ties with K-means, and optimizes the space to create well separated K clusters [20]. Interestingly, in our experiments, ℓ_2 normalizing embeddings reduced the performance of PRO by over 15% NMI. We report PRO scores for non-normalized representations, that are also more stable when transferring a threshold based on the validation set. In fact, by comparing Table 5 row 19 with row 1, we see that PRO over-estimates the number of clusters when there are few background characters (BCL, row 22, works well here), but performs better in episodes with several background characters.

While PRO estimates more clusters, that does not translate to better assignments. For example, on BUFFY-S5E4, PRO predicts 13 more clusters than BCL (35 vs. 22) and is closer to the ground-truth 32 clusters, but attains 7.7% lower NMI and 3% lower WCP. A lower purity while having more clusters is a strong indicator of bad clustering.

We also compare performance between PRO and BCL when the number of clusters K is known (see Table 6). BCL is able to consistently outperform both K-means or HAC clustering methods for the prototypical loss. We also tried extensions of K-means that automatically determine the number of clusters in an unsupervised way [14, 32] when the representations are fixed. Their performance was worse than our method of choosing a threshold on the validation set. Additional comparisons are in the supplementary material.

Qualitative. Fig. 6 visualizes clusters created by BCL (top) vs. those with PRO (bottom) on BBT-S1E1. BCL predicts 7 clusters in comparison to the ground-truth 8, and merges the singleton track of a background girl (C4 in PRO) with Penny (C3 in BCL). Both methods find the other unnamed character - C4 in BCL, C6 in PRO. While BCL merges few tracks of Sheldon and Kurt (C1), PRO is able to find Kurt (C7). However, PRO splits clusters for Raj (C1, C8), Penny (C3, C11), and Leonard (C2, C10).

Fine-tuning on each episode. Our model can be applied directly to several different datasets by using the learned threshold $4b$ without fine-tuning, this is a major advantage.


BCL (Ours) – BBT – S1E1 – NMI: 95.81 – WCP: 98.63
C: 1 | Tracks 7 | Purity: 57.14
C: 2 | Tracks 24 | Purity: 100.00
C: 3 | Tracks 112 | Purity: 99.11
C: 4 | Tracks 8 | Purity: 100.00
C: 5 | Tracks 251 | Purity: 100.00
C: 6 | Tracks 213 | Purity: 97.65
C: 7 | Tracks 41 | Purity: 100.00
PRO – BBT – S1E1 – NMI: 82.29 – WCP: 96.19
C: 1 | Tracks 9 | Purity: 100.00
C: 2 | Tracks 38 | Purity: 100.00
C: 3 | Tracks 86 | Purity: 100.00
C: 4 | Tracks 1 | Purity: 100.00
C: 5 | Tracks 1 | Purity: 100.00
C: 6 | Tracks 7 | Purity: 100.00
C: 7 | Tracks 4 | Purity: 100.00
C: 8 | Tracks 15 | Purity: 100.00
C: 9 | Tracks 44 | Purity: 95.45
C: 10 | Tracks 193 | Purity: 100.00
C: 11 | Tracks 25 | Purity: 100.00
C: 12 | Tracks 233 | Purity: 90.13
Figure 6. Visualizing clusters created by BCL (top) and PRO (bottom) for BBT-S1E1. Refer to supp. material for other episodes.

Following previous work that uses positive and negative pairs obtained automatically from each episode [6, 43, 54], BCL can be easily modified to fine-tune our model and make it cast-specific. Shots with background characters are often crowded (multiple faces), and negative constraints among them can help resolve confusion. Table 5 rows 25-27 show the overall performance improves after fine-tuning; importantly, the estimated number of characters (row 25) is much closer to the ground-truth (row 1). Details of the fine-tuning procedure and comparison against fine-tuned baselines is in the supplementary material.

6. Conclusion

We presented *Ball Cluster Learning* - a supervised approach to carve the representation space into balls of an equal radius. We showed how the radius is related to the stopping criterion used in agglomerative clustering methods, and evaluated this approach for clustering face tracks in videos. In particular, we considered a realistic setup where the number of clusters is not known, and tracks from all characters (main or otherwise) are included. We reviewed several metric learning approaches and adapted them to this clustering setup. BCL shows promising results, and to the best of our knowledge is the first approach that learns a threshold that can be used directly to estimate the number of clusters.

Acknowledgments. This work was supported by the MSR-Inria joint lab, the Louis-Vuitton - ENS Chair on AI, DARPA Explainable AI (XAI), NSERC, and Samsung.

References

[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Who's that Actor? Automatic Labelling of Actors in TV series starting from IMDB Images. In *Asian Conference on Computer Vision (ACCV)*, 2016. 1, 2

[2] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean Sum-of-squares Clustering. *Machine Learning*, 75(2):245–248, 2009. 4

[3] Martin Bäuml, Makarand Tapaswi, and Rainer Stiefelhagen. Semi-supervised Learning with Constraints for Person Identification in Multimedia Data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1, 2, 5

[4] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *Automatic Face and Gesture Recognition (FG)*, 2018. 1, 5

[5] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 2, 4, 7

[6] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Unsupervised Metric Learning for Face Identification in TV Video. In *International Conference on Computer Vision (ICCV)*, 2011. 1, 2, 8

[7] Timothee Cour, Benjamin Sapp, Akash Nagle, and Ben Taskar. Talking Pictures: Temporal Grouping and Dialog-Supervised Person Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 1, 2

[8] Daniel Defays. An Efficient Algorithm for a Complete Link Method. *The Computer Journal*, 20(4):364–366, 1977. 4

[9] Mark Everingham, Josef Sivic, and Andrew Zisserman. "Hello! My name is ... Buffy" – Automatic Naming of Characters in TV Video. In *British Machine Vision Conference (BMVC)*, 2006. 1, 2

[10] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. In *International Conference on Computer Vision (ICCV)*, 2017. 2

[11] Joris Guérin, Olivier Gibaru, Stéphane Thiery, and Eric Nyiri. CNN features are also great at unsupervised classification. *arXiv preprint, arXiv:1707.01700*, 2017. 2

[12] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? Metric Learning Approaches for Face Identification. In *International Conference on Computer Vision (ICCV)*, 2009. 2, 4, 7

[13] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. MS-Celeb-1M: A Dataset and Benchmark for Large Scale Face Recognition. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[14] Greg Hamerly and Charles Elkan. Learning the k in k-means. In *Advances in Neural Information Processing Systems (NIPS)*, 2004. 8

[15] Monica-Laura Haurilet, Makarand Tapaswi, Ziad Al-Halah, and Rainer Stiefelhagen. Naming TV Characters by Watching and Analyzing Dialogs. In *Winter Conference on Applications of Computer Vision (WACV)*, 2016. 1, 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[18] SouYoung Jin, Hang Su, Chris Stauffer, and Erik Learned-Miller. End-to-end Face Detection and Cast Grouping in Movies using Erdős–Rényi Clustering. In *International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 5

[19] Prakhar Kulshreshtha and Tanaya Guha. An Online Algorithm for Constrained Face Clustering in Videos. In *International Conference in Image Processing (ICIP)*, 2018. 2

[20] Marc T. Law, Jake Snell, Amir M. Farahmand, Raquel Urtasun, and Richard S Zemel. Dimensionality reduction for representing the knowledge of probabilistic models. In *International Conference on Learning Representations (ICLR)*, 2019. 2, 8

[21] Marc T. Law, Raquel Urtasun, and Richard S. Zemel. Deep spectral clustering learning. In *International Conference on Machine Learning (ICML)*, volume 70, pages 1985–1994, 2017. 2

[22] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. TVQA: Localized, Compositional Video Question Answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018. 1

[23] Yu Liu, Junjie Yan, and Wanli Ouyang. Quality Aware Network for Set to Set Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 5

[24] Stuart Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Info. Theory*, 28(2):129–137, 1982. 4

[25] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval (chapter 16)*. Cambridge University Press, 2008. 6

[26] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-zero Cost. In *European Conference on Computer Vision (ECCV)*, 2012. 2

[27] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018. 2

[28] Arsha Nagrani, Samuel Albanie, and Andrew Zisserman. Learnable PINs: Cross-Modal Embeddings for Person Identity. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[29] Arsha Nagrani and Andrew Zisserman. From Benedict Cumberbatch to Sherlock Holmes: Character Identification in TV series without a Script. In *British Machine Vision Conference (BMVC)*, 2017. 1, 2

[30] Omkar M. Parkhi, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. A Compact and Discriminative Face Track Descriptor. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2, 5

[31] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep Face Recognition. In *British Machine Vision Conference (BMVC)*, 2015. 2

[32] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *International Conference on Machine Learning (ICML)*, 2000. 8

[33] Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. Linking People in Videos with "Their" Names Using Coreference Resolution. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[34] Anna Rohrbach, Marcus Rohrbach, Siyu Tang, Seong Joon Oh, and Bernt Schiele. Generating Descriptions with Grounded and Co-Referenced People. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[35] Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. Movie Description. *International Journal of Computer Vision (IJCV)*, 123(1):94–120, 2017. 1

[36] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A Unified Embedding for Face Recognition and Clustering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 4, 7

[37] Matthew Schultz and Thorsten Joachims. Learning a Distance Metric from Relative Comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2004. 2

[38] Vivek Sharma, M. Saquib Sarfraz, and Rainer Stiefelhagen. A Simple and Effective Technique for Face Clustering in TV Series. In *Workshop at Conference on Computer Vision and Pattern Recognition (CVPRW)*, 2017. 1

[39] Vivek Sharma, Makarand Tapaswi, M. Saquib Sarfraz, and Rainer Stiefelhagen. Self-Supervised Learning of Face Representations for Video Face Clustering. In *Automatic Face and Gesture Recognition (FG)*, 2019. 2, 5

[40] Josef Sivic, Mark Everingham, and Andrew Zisserman. "Who are you?" – Learning person specific classifiers from video. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[41] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 2, 4, 7

[42] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. "Knock! Knock! Who is it?" Probabilistic Person Identification in TV series. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[43] Makarand Tapaswi, Omkar M. Parkhi, Esa Rahtu, Eric Sommerlade, Rainer Stiefelhagen, and Andrew Zisserman. Total Cluster: A person agnostic clustering method for broadcast videos. In *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, 2014. 2, 6, 8

[44] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. MovieQA: Understanding Stories in Movies through Question-Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[45] Paul Vicol, Makarand Tapaswi, Lluís Castrejón, and Sanja Fidler. MovieGraphs: Towards Understanding Human-Centric Situations from Videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 5

[46] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. NormFace: l2 Hypersphere Embedding for Face Verification. In *ACM International Conference on Multimedia (ACM MM)*, 2017. 2, 3, 4

[47] Jixuan Wang, Kuan-Chieh Wang, Marc T. Law, Frank Rudzicz, and Michael Brudno. Centroid-based deep metric learning for speaker recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019. 8

[48] Ruiping Wang, Huimin Guo, Larry S. Davis, and Qionghai Dai. Covariance Discriminative Learning: A Natural and Efficient Approach to Image Set Classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 5

[49] Baoyuan Wu, Siwei Lyu, Bao-Gang Hu, and Qiang Ji. Simultaneous Clustering and Tracklet Linking for Multi-face Tracking in Videos. In *International Conference on Computer Vision (ICCV)*, 2013. 2

[50] Baoyuan Wu, Yifan Zhang, Bao-Gang Hu, and Qiang Ji. Constrained Clustering and its Application to Face Clustering in Videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[51] Shijie Xiao, Mingkui Tan, and Dong Xu. Weighted Block-sparse Low Rank Representation for Face Clustering in Videos. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[52] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[53] Jiaolong Yang, Peiran Ren, Dongqing Zhang, Dong Chen, Fang Wen, Hongdong Li, and Gang Hua. Neural Aggregation Network for Video Face Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 5

[54] Shun Zhang, Yihong Gong, and Jinjun Wang. Deep Metric Learning with Improved Triplet Loss for Face Clustering in Videos. In *Pacific Rim Conference on Multimedia*, 2016. 2, 8

[55] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Joint Face Representation Adaptation and Clustering in Videos. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 5, 6

[56] Henry Zhou, Makarand Tapaswi, and Sanja Fidler. Now You Shake Me: Towards Automatic 4D Cinema. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1