Video Face Clustering with Unknown Number of Clusters
SUPPLEMENTARY MATERIAL

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

In this document, we discuss how we fine-tune our models on the test episodes using the BCL loss. We also briefly discuss the challenges to fine-tune using triplet or prototypical losses, but show that they can use the constraints inspired by BCL in Sec. A. Additionally, we will show that variants of K-means that aim to predict the number of clusters such as X-means [4] and G-means [3] perform worse than our proposed method (Sec. B). Finally, we present additional quantitative and qualitative results on the TV series episodes introduced in the main paper in Sec. C.

A. Fine-tuning on test episodes

As discussed in the related work section, many clustering approaches use unsupervised constraints that arise from the video to learn cast-specific metrics [2, 7, 8, 9, 10, 11, 12]. The positive constraints are obtained from face images within a track that are considered similar; and negative constraints from faces that appear at the same time in the video that can be assumed to be dissimilar. Note that most previous works know the number of clusters, and use the constraints to improve the distance metric.

In the following, we show how our method can be modified to work with such positive and negative constraints. We also discuss the limitations of the baselines for fine-tuning, but propose an alternative that uses BCL pair-wise constraints.

Ball Cluster Learning. Recall that our constraints are originally based on cluster samples and their centroids. For all $x_i \in \mathcal{C}_k$, BCL aims to satisfy $\mathsf{d}^2(\mathbf{f}_i, \mu_k) < b$ and $\mathsf{d}^2(\mathbf{f}_i, \mu_v) > \gamma$, where $\gamma = 9b + \epsilon$ and $x_i \notin \mathcal{C}_v$.

For a positive pair $x_i \in \mathcal{C}_k, x_j \in \mathcal{C}_k$, and a negative counterpart $x_u \in \mathcal{C}_v$, the centroid constraints can be modified as:

$$\mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_j) < 4b \quad \text{and} \quad \mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_u) > 4b + \epsilon. \tag{1}$$

In practice, as the model is already trained, we wish to only fine-tune on the positive and negative pairs. In most cases, the positive constraints are already satisfied, and using the relaxed constraint hurt performance. Thus, we use: $\mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_j) < \min\left(\mathsf{d}^2_{\text{ori}}(\mathbf{f}_i, \mathbf{f}_j), 4b\right)$, where $\mathsf{d}^2_{\text{ori}}(\mathbf{f}_i, \mathbf{f}_j)$ is the distance between the pair prior to fine-tuning. The constraints are formulated as loss functions to train the model by using the $[\cdot]_+ = \max(0, \cdot)$ operator as before.

We tried an analogous strategy for dissimilar pairs (using $\max\left(\mathsf{d}^2_{\text{ori}}(\mathbf{f}_i, \mathbf{f}_u), 4b + \epsilon\right)$) but it did not provide significant performance improvement. Thus, we ignored it.

During training, we freeze the ball squared-radius b, use a learning rate of 0.0003 (0.1 times the original), and select a random face image from each track in the pairs (about 1,000 pairs for each episode). The model parameters are updated for a fixed number of iterations (2,000 for all single episodes). As before, we perform clustering by using Hierarchical Agglomerative Clustering (HAC) with complete linkage and distance threshold $\tau = 4b$.

Triplet loss [5] is well suited to train a model with above-mentioned automatically obtained positive and negative pairs. However, it does not involve learning a threshold that can be used directly with HAC (or any clustering). As we are fine-tuning on test episodes, we do not have access to a validation set that would allow us to obtain such a threshold. Furthermore, optimizing performance on each test episode by choosing a new *best* threshold is inappropriate.

To circumvent this, we use the original threshold learned on the validation set τ and formulate pair-wise constraints in a similar manner to BCL. In particular, the positive pairs follow $\mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_j) < \tau$ and negative pairs $\mathsf{d}^2(\mathbf{f}_i, \mathbf{f}_u) > \tau$. Thus, we fine-tune the model checkpoint trained using the triplet loss with the BCL loss. All other implementation details (learning rate, number of iterations, *etc*.) are same as those used for BCL.

Prototypical loss [6]. Unlike the triplet loss that is designed to work with samples (triplets), the prototypical loss needs class/cluster centroids. It also faces the same challenge

of not knowing the number of clusters, or not having a clear stopping criterion for HAC. Nevertheless, as discussed for triplet loss above, we adopt BCL pair-wise constraints for the prototypical loss with the threshold τ chosen on validation. All other details are kept same.

Evaluation. Table 1 presents results of fine-tuning on each episode. For each episode, we see that BCL fine-tuned model (BCL-FT) is able to predict the number of clusters quite accurately. We believe this can be attributed to background characters often appearing simultaneously, providing sufficient negative constraints. On the combined episodes datasets (last three columns of Table 1), the negative constraints are within each episode, and it is not possible to distinguish between background characters across episodes. This explains why BCL predicts fewer clusters than ground-truth.

Both baselines (TRI-FT and PRO-FT) also show good performance improvements after fine-tuning with BCL. With respect to predicting the number of clusters, PRO-FT and BCL-FT seem to have flipped roles, with PRO-FT now predicting fewer clusters after fine-tuning, but over-clustering before fine-tuning (see Table 5 of the main paper). However, note that BCL-FT with more clusters has higher purity, while PRO (no fine-tune) had lower purity even with more clusters (in Table 5).

Conclusion. This experiment emphasizes that BCL combines the best of all worlds. Models can be trained with both samples and centroids, or pairs and triplets. Most importantly, BCL learns a threshold to predict the number of clusters automatically. In addition, BCL pair-wise loss can be used to fine-tune models trained with other losses to achieve performance gains.

B. K-means variants

Owing to the popularity of the K-means approach for clustering, there is some work on automatically estimating the number of clusters while performing clustering based on some criterion. In this section, we will look at two such methods and analyze how they work when applied to our challenging datasets. Note that both these methods do not further learn an embedding, but rely on existing features. Thus, we evaluate performance on the Base CNN representations – that are actually quite good (see Table 3), as well as the features learned using our BCL loss function.

X-means [4] In this variant, clustering starts with all samples in the same cluster and splits until some stopping criterion. In particular, at each iteration, a cluster is split into two components. The Bayesian Information Criterion (BIC) is used to decide whether the newly created two clusters are preferred over the original single cluster. Clustering stops when a maximum number of clusters $K_{\max}$ has been crossed, or when further splitting any cluster would result in lowered BIC scores.

Table 2 rows 2-4 show the performance of X-means when using base CNN features, and rows 5-7 when using features trained with BCL loss. We choose $K_{\max}$ to be 40 for the BBT episodes, 80 for BUFFY, 150 for BBT (6 episodes combined) and BUFFY (6 episodes combined), and 300 for BOTH (all 12 episodes). These are strong upper bounds for all datasets. However, as seen from the results, the method stops the iterations only after crossing the maximum number of clusters in all datasets (*i.e*. all predicted number of clusters are higher than $K_{\max}$). This, together with the poor NMI scores, suggests that the using BIC may not be a sufficiently strong criterion for a complex dataset.

G-means [3] Similar to X-means, this approach also starts with all samples in one cluster, and iteratively splits them based on some criterion. Different to X-means, the stopping criterion used here determines the "Gaussian-ness" of samples around the cluster centroid. In particular, clusters that have a strong Gaussian shape are not split further, while others (*e.g*. those that may be bimodal) are split into two. This process repeats until no more clusters can be split. The Anderson-Darling test is used to determine whether a distribution is Gaussian.

We present the results of G-means in the second half of Table 2. We see that G-means also fails at reliably estimating the number of clusters, and prefers to overcluster all datasets.

C. Additional evaluation

Base CNN representations. In Table 3, we present the performance of base CNN representations with standard HAC clustering and using a threshold learned on the validation set. These results should be analyzed together with Table 5 of the main paper, but were omitted due to space constraints. Note that the threshold is chosen such that correct number of clusters are created on the validation set. While the base representation is quite good (as seen in NMI and WCP curves), choosing a threshold is an unreliable method and results in over-clustering on the test episodes.

When K is known. We also present results when (for some reason) the number of clusters K is known. We compare against best performing baselines: triplet and prototypical loss, on all episodes of the test set. Table 4 shows that our method is able to achieve higher NMI and WCP in most cases (12 out of 15). Note that this experiment is presented for completeness, as the main point of BCL is to automatically predict the number of clusters, when K is unknown. All results are without fine-tuning.

		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
Pre-trained Triplet Loss [5] + BCL Fine-tune																
2	#Cl	6	7	13	12	12	23	16	21	16	18	22	23	33	54	73
3	NMI	**97.98**	97.13	91.22	86.37	95.25	83.38	**85.91**	82.62	78.98	75.34	76.04	78.66	**89.09**	76.28	78.95
4	WCP	99.09	**99.84**	92.73	89.23	94.08	86.31	91.95	87.71	85.76	78.95	80.83	79.59	92.40	81.45	82.26
Pre-trained Prototypical Loss [6] + BCL Fine-tune																
5	#Cl	6	**6**	19	16	15	41	17	22	21	27	21	34	61	72	113
6	NMI	96.76	97.09	91.43	**90.83**	**95.84**	**85.38**	77.98	**83.01**	**79.29**	77.24	**81.74**	**82.31**	87.74	**79.90**	**82.81**
7	WCP	98.17	**99.67**	95.00	93.31	94.85	**93.69**	90.82	87.11	88.86	82.52	84.76	84.80	93.96	85.17	85.26
Pre-trained Ball Cluster Learning (Ours) + BCL Fine-tune																
8	#Cl	**9**	8	**24**	**24**	**21**	**36**	23	27	25	**36**	**38**	**40**	**69**	**78**	**126**
9	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
10	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 1. Clustering performance on episodes of the test set, with fine-tuning on the test set. The last three columns show results on datasets created by combining tracks from several episodes. #Ch is the ground-truth number of characters (row 1); and #Cl is number of predicted clusters and should be close to the number of characters. 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						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
X-Means on Base CNN representation																
2	#Cl	41	42	45	41	51	48	93	89	90	82	101	91	175	180	327
3	NMI	55.74	56.83	65.75	64.89	66.47	66.51	59.61	66.86	57.56	62.99	61.03	66.23	56.26	62.67	65.85
4	WCP	98.63	97.89	92.73	90.70	94.27	84.64	94.59	90.74	88.86	87.75	88.21	86.15	92.71	87.21	88.68
X-Means on features learned with BCL																
5	#Cl	42	25	45	34	43	44	97	41	81	84	80	91	162	163	313
6	NMI	56.90	62.46	69.82	68.35	71.46	70.08	60.12	72.16	59.99	65.90	66.59	69.72	58.62	63.91	66.63
7	WCP	99.24	97.89	95.45	91.35	96.18	87.38	95.85	88.62	91.04	91.31	92.02	89.48	93.22	88.73	88.87
G-Means on Base CNN representation																
8	#Cl	57	31	56	33	46	67	87	101	117	74	73	101	243	404	567
9	NMI	55.29	62.61	70.41	71.48	71.74	67.00	58.07	66.63	58.34	67.06	63.50	69.08	57.64	61.57	66.07
10	WCP	98.32	96.42	94.85	91.35	94.85	86.55	89.18	89.93	88.94	89.31	87.14	87.68	92.14	87.21	87.95
G-Means on features learned with BCL																
11	#Cl	23	39	41	35	45	79	42	54	75	63	72	106	137	337	481
12	NMI	68.51	64.54	72.74	70.38	73.22	66.18	67.87	61.43	62.88	65.70	66.91	69.09	60.86	61.63	64.42
13	WCP	98.93	98.05	92.42	91.35	94.85	87.02	91.07	71.30	84.00	86.41	88.45	87.86	89.94	85.51	84.06

Table 2. Clustering performance on episodes of the test set using two variants of K-means that predict the number of clusters.

Choosing a threshold on train set. As the training set is much larger than validation, it might seem that the baselines may perform better when choosing a threshold on the validation set. However, this is not the case as observed from Table 5. As the MLP φ_θ fits well to the training set a smaller cutoff threshold (distance) is selected resulting in more clusters on unseen data. Thus, it is important to have a separate validation set.

NMI and WCP vs. number of clusters. We plot the NMI and WCP curves for all methods for each episode of BBT in Fig. 1 and BUFFY in Fig. 2. All results are before fine-tuning. We wish to draw the reader to the following observations:

1. The threshold for prototypical loss is quite stable and is able to predict the number of clusters well (as was discussed in Table 5 of the main paper). However, the **purity is almost always lower than our method**, indicating that even though it makes more clusters, the formed clusters tend to be more heterogeneous (*i.e.* contain more samples from different categories).
2. Our method shows higher NMI and WCP irrespective of the operating point in most episodes.
3. The base representations have very good performance curves. However, their operating points (chosen based on the validation threshold) are far from the optimal number of clusters. Cross-entropy loss, especially when used with thousands of classes, seems to be effective at learning classification as well as clustering.

Qualitative visualization of clusters. Finally, we visualize the clusters created by Triplet loss (TRI), Prototypical

		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
		Base CNN representation														
2	#Cl	36	38	49	51	36	59	76	75	94	93	95	92	200	407	609
3	NMI	59.84	60.60	68.18	69.18	74.85	68.42	61.66	67.94	57.88	65.01	65.59	66.90	59.33	60.00	64.80
4	WCP	97.41	97.72	92.88	92.66	95.42	86.31	91.95	88.02	88.27	89.64	91.31	85.52	93.19	88.08	89.14

Table 3. Clustering performance on episodes of the test set when using base CNN representation.

Method	Metric	BBT						BUFFY						BBT	BUFFY	ALL
		S1E1	S1E2	S1E3	S1E4	S1E5	S1E6	S5E1	S5E2	S5E3	S5E4	S5E5	S5E6	6 ep.	6 ep.	12 ep.
Triplet Loss [5]																
KM	NMI	73.2	83.4	74.7	70.5	75.5	67.8	74.3	64.7	68.7	64.6	68.8	66.0	57.8	59.2	62.6
KM	WCP	93.6	96.7	93.2	91.4	93.7	83.7	88.1	72.3	80.8	80.1	86.1	78.0	89.7	77.7	80.5
HAC	NMI	88.3	69.0	79.1	76.6	81.9	70.8	76.2	64.1	66.9	64.9	70.7	67.6	64.0	60.1	65.3
HAC	WCP	**98.5**	79.2	93.0	91.5	94.7	80.4	86.4	67.4	77.3	73.3	84.0	76.5	90.2	72.7	76.9
Prototypical Loss [6]																
KM	NMI	80.8	82.5	76.0	72.3	76.0	69.4	79.5	74.6	74.3	71.9	71.4	72.5	60.7	64.5	66.8
KM	WCP	95.0	95.6	93.6	**91.7**	94.3	83.7	89.4	84.2	**83.7**	86.0	88.9	84.7	91.3	85.4	85.6
HAC	NMI	87.6	86.6	83.1	**80.3**	**89.2**	74.0	67.9	68.4	77.4	70.6	76.5	73.8	68.3	65.8	70.3
HAC	WCP	94.7	94.8	**94.2**	91.0	**96.2**	85.2	74.8	73.0	79.9	81.2	85.1	82.8	91.1	80.0	83.3
Ball Cluster Learning (Ours)																
KM	NMI	83.9	90.1	73.2	70.2	76.9	71.6	78.9	**79.1**	72.0	72.9	71.6	74.8	60.5	66.7	68.7
KM	WCP	**98.5**	99.2	92.9	91.2	93.5	86.1	**91.6**	**87.4**	81.6	**87.6**	88.8	87.7	92.0	**87.3**	**88.0**
HAC	NMI	**92.8**	**91.9**	**84.3**	78.5	86.1	**76.1**	**81.3**	75.3	**77.9**	**75.9**	**76.9**	**78.6**	**70.6**	**69.1**	**72.5**
HAC	WCP	**98.6**	**98.2**	92.6	**91.7**	96.0	**86.7**	89.6	81.4	81.4	87.3	**91.2**	**88.5**	**93.0**	85.3	86.2

Table 4. Comparison between models trained with triplet, prototypical, and our approach when the number of clusters is known. We evaluate both K-means (KM) as well as Hierarchical Agglomerative Clustering (HAC) to obtain the appropriate number of clusters. A short version of this appeared as Table 6 in the main paper.

Loss (PRO), and our method Ball Cluster Learning (BCL) on one episode of BBT (Fig. 3) and BUFFY (Fig. 4). Each cluster is visualized by selecting 6 random face tracks (when available), and one face image per track. All results are without fine-tuning.

These figures also throw light on the difficulty of our dataset that includes wide variations in illumination and pose. Tracks, their labels and features, and our implementation of BCL is available at https://github.com/makarandtapaswi/BallClustering_ICCV2019.

In Fig. 3, BCL achieves close to the correct number of clusters, and separates the unknown character with just 2 tracks (C:2). Both triplet and prototypical losses lead to over clustering. *E.g.* Leonard is split to C:1, C:6, C:7 and C:12 in triplet loss, and C:1, C:4, C:6, C:11, C:13 when using prototypical loss.

BUFFY-S5E3 (Fig. 4) is a unique episode in which one of the lead characters *Xander* is duplicated due to a magic spell (the duplicate is played by the actor's identical twin). Nevertheless, we see that BCL achieves reasonable performance, and is able to find minor characters (Joyce C:9, the building manager C:10), as well as isolate one of the background characters (C:11).

References

[1] 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. 5

[2] 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

[3] Greg Hamerly and Charles Elkan. Learning the k in k-means. In *Advances in Neural Information Processing Systems (NIPS)*, 2004. 1, 2

[4] 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. 1, 2

[5] 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. 1, 3, 4, 5

[6] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1, 3, 4, 5

[7] 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. 1

[8] Baoyuan Wu, Siwei Lyu, Bao-Gang Hu, and Qiang Ji. Simultaneous Clustering and Tracklet Linking for Multi-face

Thresh	BBT						BUFFY					
Set	S1E1	S1E2	S1E3	S1E4	S1E5	S1E6	S5E1	S5E2	S5E3	S5E4	S5E5	S5E6
#Ch	8	6	26	28	25	37	13	22	15	32	38	45
Contrastive Loss [1]												
train	374 (40.6)	382 (41.0)	411 (52.5)	443 (51.7)	341 (54.3)	629 (56.7)	605 (48.1)	721 (55.0)	862 (46.8)	694 (53.5)	608 (53.7)	725 (57.9)
val	14 (62.5)	13 (63.7)	17 (61.8)	22 (65.6)	19 (71.4)	32 (55.7)	22 (61.0)	30 (53.9)	26 (58.2)	29 (53.4)	29 (53.6)	27 (52.0)
Triplet Loss [5]												
train	28 (65.5)	31 (63.4)	38 (74.6)	44 (74.1)	39 (78.2)	72 (68.3)	64 (65.0)	73 (64.9)	77 (60.1)	67 (63.6)	66 (69.4)	79 (68.4)
val	9 (88.1)	12 (71.2)	15 (79.8)	16 (76.7)	13 (85.8)	23 (69.3)	23 (73.6)	24 (64.2)	25 (66.2)	22 (63.6)	23 (67.9)	26 (65.5)
Prototypical Loss [6]												
train	19 (74.6)	25 (69.3)	35 (77.4)	39 (76.6)	27 (86.6)	63 (73.9)	50 (70.7)	55 (69.7)	43 (67.8)	60 (69.7)	63 (72.7)	65 (74.3)
val	12 (82.3)	15 (75.1)	22 (83.7)	28 (80.3)	18 (91.4)	41 (74.3)	32 (74.2)	32 (71.0)	20 (76.2)	35 (70.5)	40 (76.6)	36 (73.5)

Table 5. Choosing the HAC threshold on train vs. validation set. Showing the number of predicted clusters and NMI. Ideal number of clusters is presented in the first row. Note how it is beneficial to have a separate validation set, as overfitting on training can lead to selection of smaller thresholds.

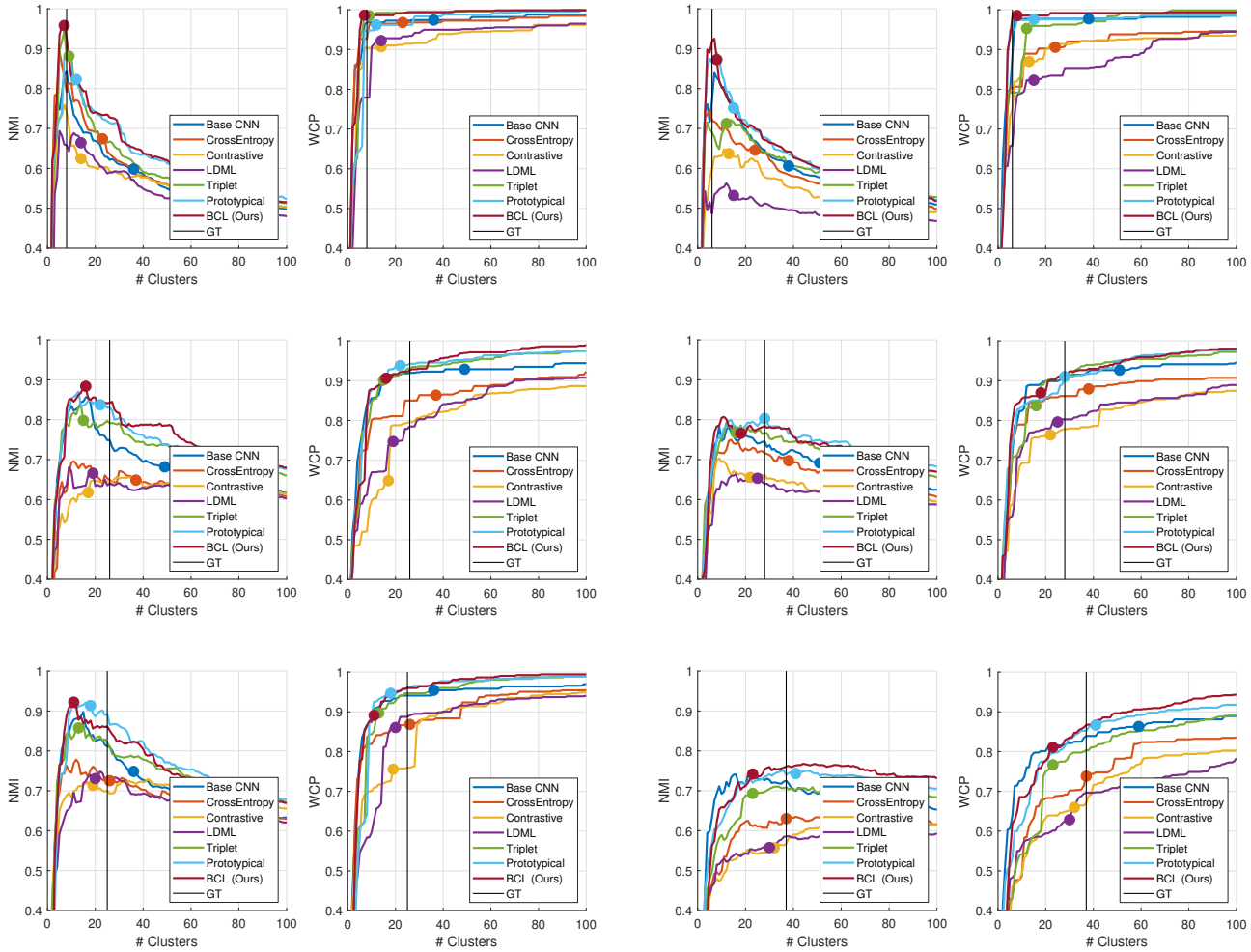

Figure 1. NMI and WCP vs. number of clusters for BBT-S1E1 to S1E6 (left to right, top to bottom). Circles indicate operating points (*i.e.* number of predicted clusters for the methods), our method uses the HAC threshold 4*b*, while all others are using the threshold tuned to give 66 clusters on the validation set. Best seen in color.

Tracking in Videos. In *International Conference on Computer Vision (ICCV)*, 2013. 1

[9] 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. 1

[10] 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. 1

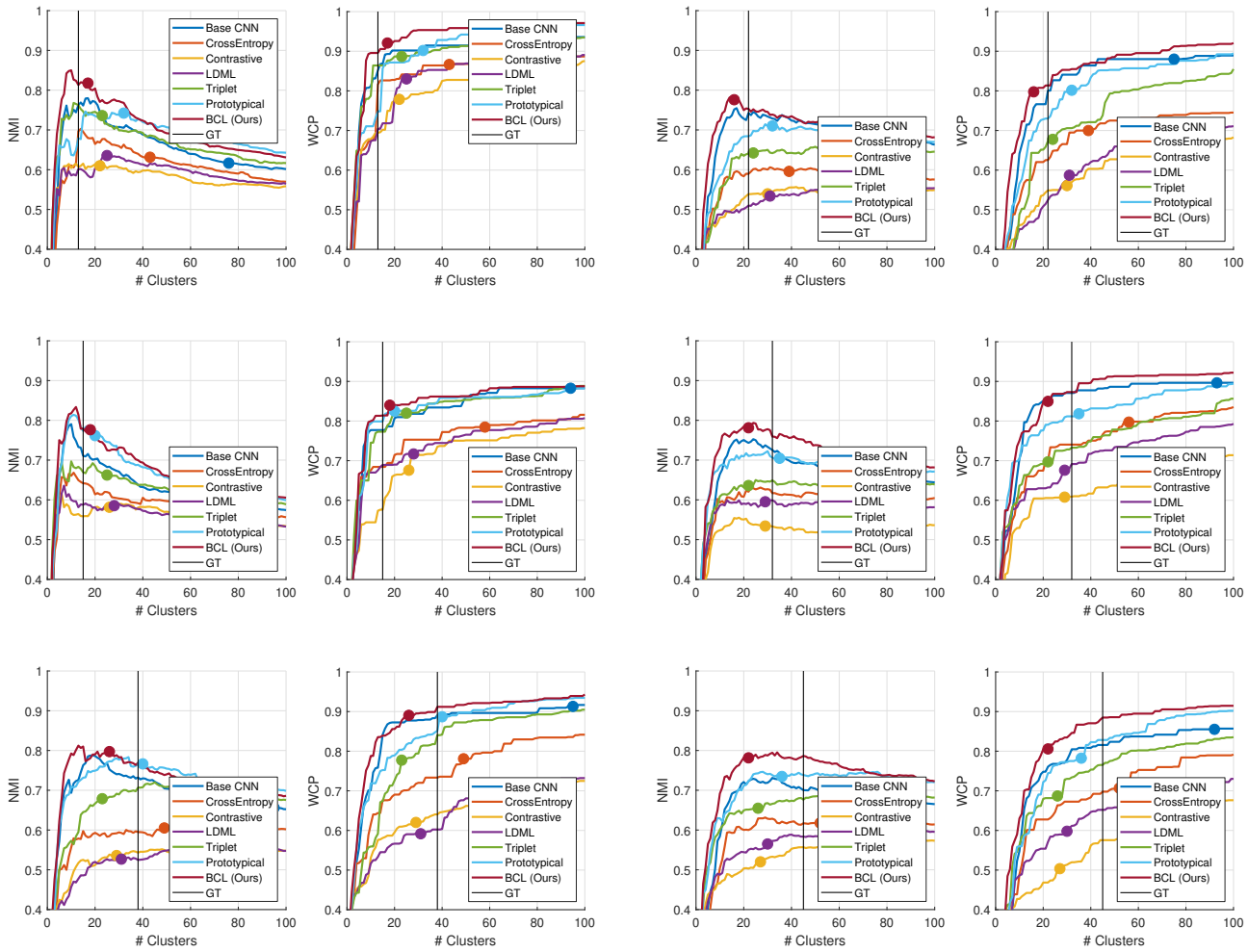

Figure 2. NMI and WCP vs. number of clusters for BUFFY-S5E1 to S5E6 (left to right, top to bottom). Circles indicate operating points (*i.e.* number of predicted clusters for the methods), our method uses the HAC threshold 4*b*, while all others are using the threshold tuned to give 66 clusters on the validation set. Best seen in color.

[11] 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. 1

[12] 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

Figure 3. Clusters created by triplet loss (TRI, top), prototypical loss (PRO, middle), and BCL (bottom) on BBT-S1E2. The correct number of clusters is 6.

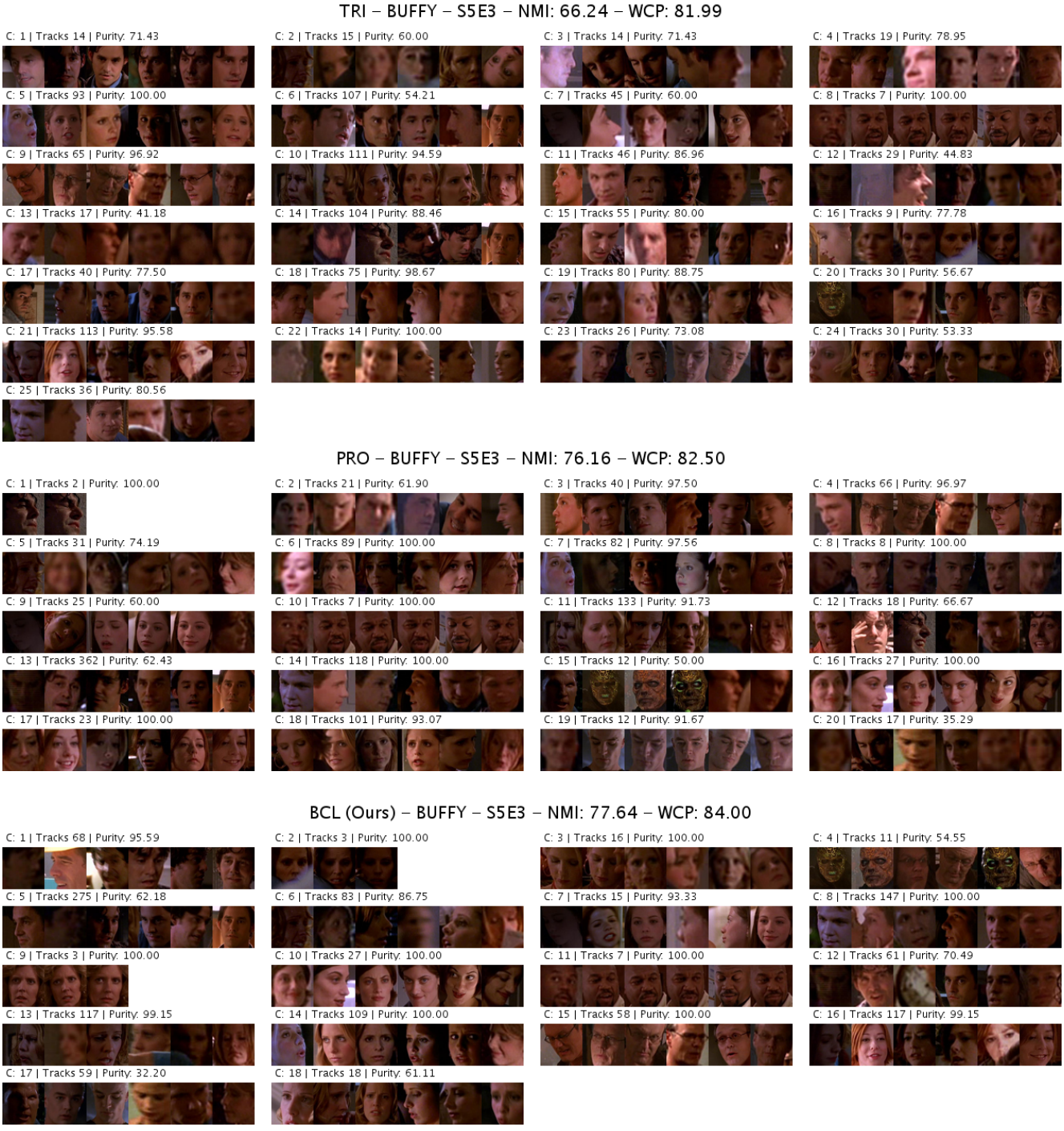

Figure 4. Clusters created by triplet loss (TRI, top), prototypical loss (PRO, middle), and BCL (bottom) on BUFFY-S5E3. The correct number of clusters is 15.