Time-series Clustering with Jointly Learning Deep Representations, Clusters and Temporal Boundaries

Panagiotis Tzirakis[1], Mihalis A. Nicolaou[2], Björn Schuller[1,3] and Stefanos Zafeiriou[1,4]
[1] Department of Computing, Imperial College London, UK
[2] Computation-based Science and Technology Research Centre, The Cyprus Institute, Cyprus
[3] ZD.B Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany
[4] Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland

***Abstract*— Clustering and segmentation of temporal data is an important task across several fields, with prominent applications in computer vision and machine learning such as face and gesture segmentation. Several related methods have been proposed in literature, focusing on learning temporal boundaries and clusters, with recent works focusing on learning deep representations for clustering. However, none of the proposed methods is suitable for jointly learning segments, clusters, as well as representations. In this paper, we propose the first methodology that simultaneously discovers suitable deep representations, as well as clusters and temporal boundaries, with the clustering process providing supervisory cues for updating temporal boundaries and training the proposed deep learning architecture. We demonstrate the power of the proposed approach on a human motion segmentation task using the CMU-MMAC database. Our method provides the best results with respect to normalized mutual information compared to other clustering algorithms.**

I. INTRODUCTION

Data clustering has been a popular, challenging topic in machine learning and computer vision for more than 50 years [3], [12], [2], with a wide range of applications that include face and gesture analysis [16], [9], as well as segmentation and motion pattern detection [15], [10], [13], [25]. A popular application of clustering lies in problems arising in the context of human behaviour analysis, while the setting becomes even more challenging when considering time-series data [6], [11]. Clustering of human behaviour aims at grouping the visual information in such a way that the behavioural segments that belong to the same group are similar while those that belong to different groups are dissimilar. It also refers to grouping and (automatically) segmenting various videos to different behaviours, or human actions.

The task of clustering human behaviour introduces many challenges, including (a) dealing with high dimensional data, (b) presence of outliers (which are in abundance in visual data and cannot be easily modelled), as well as, (c) temporal misalignment, i.e., data of different lengths and phases. In more detail, challenge (c) renders standard distance metrics, such as Euclidean distance, not applicable to measuring the similarity between data. In order to alleviate this issue, temporal alignment and warping techniques such as Dynamic Time Alignment Kernel (DTAK) [17] are often utilized.

Many studies in literature focus on temporally segment the time series and compute clusters [21] by either utilizing DTAK [27] or similarity matrices [8]. Other studies focus on learning deep representations (e.g. autoencoders) and clusters [4]. However, none of the studies try to jointly learn representations, segments and clusters.

In this work, we propose the first, to the best of our knowledge, methodology that simultaneously: (a) clusters images from raw dynamic data in an hierarchical manner, (b) learns deep representations in an end-to-end manner utilizing convolution neural networks (CNNs), and (c) identifies the temporal boundaries of segments. For our purposes, we utilise the graph degree linkage clustering algorithms. The intuition behind our choice is that small temporal scales can be extracted (*e.g.* moving arm for human motion) before longer ones (*e.g* walking). The similarity between different length segments is computed with DTAK [17]. The CNN parameters are learned with supervisory cues from the clustering results. The last step of our algorithm alters the temporal boundaries based on the clustering results with a coordinate-descent algorithm. The effectiveness of our method is shown on human motion segmentation task using the CMU-MMAC database [5].

II. RELATED WORK

Our work is most closely related to temporal segmentation, and learning deep representations and clusters in a unified framework. To this end, our related work is focused on these studies.

Temporal Clustering. A popular temporal clustering approach is the change-point detection [7] which tries to identify the location of timely-unknown changes in the time series. Another method is the switching linear dynamical system (SLDS) [14] which switches linear dynamical systems over time to capture the contextual information in the time series. Two recent algorithms are the Aligned Cluster Analysis (ACA) [27] and Hierarchical Aligned Cluster Analysis (HACA) [28], which are an extension of kernel k-means and spectral clustering for temporal data. More specifically, these methods segment the time series and compute the clusters in a unified framework utilising dynamic programming. In a different study, Krüger et al. [8] solves the problem by utilising neighbourhood graphs and similarity information in the graphs. All of the aforementioned studies find clusters and

segments but none tries to also learn deep representations.

Deep Representation Learning. Several studies have been proposed that train deep representations and perform clustering. For example in [18], [19] the authors propose deep semi-NMF, a model that decomposes observations into multiple factors. In another study, Wang et al. [20] use sparse coding to extract image-based features and subsequently learn deep representations utilizing a cluster-oriented loss. Other studies utilise raw pixel intensities to jointly learn deep representations and clusters. For example, Yang et al. [23] use an agglomerative clustering method and the clustering loss to train their model. In a more recent study [22] the authors use K-means and further utilise an autoencoder reconstruction penalty, with clustering performed in the bottleneck space. Although the aforementioned studies learn deep representations, none of them finds temporal segments in time series.

III. BACKGROUND

To make the paper self-complete, we briefly review the Dynamic Time Alignment Kernel (DTAK) [17] which is used to compute distances between segments of different length. The kernel extends the dynamic time warping (DTW) in that it satisfies the triangle inequality [24].

More particularly, given two sequences $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ (of length N) and $\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_K]$ (of length K) where $\mathbf{x}_i$, $\mathbf{z}_j$ are the frames at time i, j, the DTAK first computes the frame kernel matrix $\mathbf{K}$ as follows:

$$\mathbf{K}_{ij} = exp(-\frac{||\mathbf{x}_i - \mathbf{z}_j||^2}{2\sigma^2}) \tag{1}$$

where σ is the bandwidth of the kernel. Then the similarity between the segments is defined as follows

$$\psi(\mathbf{X}, \mathbf{Z}) = \frac{\mathbf{S}_{NK}}{N + K} \tag{2}$$

where $\mathbf{S}$ is a cumulative kernel matrix computed in a recursive manner as follows

$$\mathbf{S}_{ij} = max \begin{cases} \mathbf{S}_{i-1,j} + \mathbf{K}_{ij} \\ \mathbf{S}_{i-1,j-1} + 2\mathbf{K}_{ij} \\ \mathbf{S}_{i,j-1} + \mathbf{K}_{ij} \end{cases} \tag{3}$$

where $\mathbf{S}_{11} = 2\mathbf{K}_{11}$.

IV. PROPOSED METHOD

Our method can be split in three different parts: (i) agglomerative clustering, (ii) temporal segmentation, and (iii) representation learning. The unified objective function can be defined as

$$\arg\min_{\mathbf{y},\theta,\mathbf{s}} \mathcal{L}(\mathbf{y}, \theta, \mathbf{s}|\mathbf{X}), \tag{4}$$

where $\mathbf{y}$ are the cluster labels, θ are the CNN parameters, $\mathbf{s}$ is a vector containing the segments of the sequence $\mathbf{X}$, and $\mathcal{L}$ is the loss function. Each of the three parts is presented in the following subsections along with the recurrent framework that minimises the objective function by combining them together.

A. Agglomerative Clustering Algorithm

Agglomerative clustering is a bottom up clustering approach, meaning that it starts with a large number of small clusters and iteratively merges two clusters that have the highest affinity. The process stops when some stopping criterion is met (*e.g.* a specified number of clusters). Mathematically, at each time step the method merges clusters $\mathbf{C}_a$ and $\mathbf{C}_b$ if

$$\{\mathbf{C}_a, \mathbf{C}_b\} = \arg \max_{\mathbf{C}_i, \mathbf{C}_j \in \mathbf{C}, i \neq j} \mathcal{A}(\mathbf{C}_i, \mathbf{C}_j) \tag{5}$$

where $\mathbf{C}$ is the set with all the clusters and $\mathcal{A}$ indicates the affinity between two clusters.

In our method, we exploit the benefits of the graph-degree linkage clustering algorithm [26]. The algorithm starts by creating a K-Nearest Neighbour (K-NN) graph, where the affinity between clusters is computed based on the indegree and outdegree of the vertices (samples) in the graph. The affinity is defined as follows:

$$\begin{aligned} \mathcal{A}(\mathbf{C}_a, \mathbf{C}_b) = &\frac{1}{|\mathbf{C}_a|^2} \mathbf{1}^T_{|\mathbf{C}_a|} \mathbf{W}_{\mathbf{C}_a, \mathbf{C}_b} \mathbf{W}_{\mathbf{C}_b, \mathbf{C}_a} \mathbf{1}_{|\mathbf{C}_a|} + \\ &\frac{1}{|\mathbf{C}_b|^2} \mathbf{1}^T_{|\mathbf{C}_b|} \mathbf{W}_{\mathbf{C}_b, \mathbf{C}_a} \mathbf{W}_{\mathbf{C}_a, \mathbf{C}_b} \mathbf{1}_{|\mathbf{C}_b|} \end{aligned} \tag{6}$$

where $\mathbf{W}$ is an adjacent matrix and $\mathbf{W}_{\mathbf{C}_b, \mathbf{C}_a}$ corresponds to the $\mathbf{C}_b$ row and $\mathbf{C}_a$ column. We should note that in our method segments are considered as the samples of the agglomerative clustering algorithm. To this end, each element in $\mathbf{W}$ is computed by using the DTAK. Figure 1 (middle) depicts a toy example of the 1-NN graph the algorithm constructs, where each sample is a segment. For more details of this type of clustering the interested reader is referred to [26].

B. Temporal Segmentation

Our method performs temporal segmentation by utilising a coordinate-descent optimization method, inspired by [27]. In particular, by keeping the parameters of the CNN fixed, the method alternates between optimizing segments and clusters, i. e.,

$$\arg\min_{\mathbf{y},\mathbf{s}} \mathcal{L}(\mathbf{y}, \mathbf{s}|\mathbf{X}.\theta) \tag{7}$$

As in [27] we introduce an auxiliary function $\mathcal{L} : [1, N] \rightarrow \mathbb{R}$:

$$\mathcal{L}(v) = \min_{\mathbf{y},\mathbf{s}} \mathcal{L}(\mathbf{y}, \mathbf{s}|\theta)|_{\mathbf{X}_{[1,v]}} \tag{8}$$

where v is a position in the sequence $\mathbf{X}$, and $\mathbf{X}_{[1,v]}$ denotes the segment $[1, v]$ in $\mathbf{X}$. We can further justify that if there is a position i in $\mathbf{X}$ such that $i < v$ then the optimal decomposition of the sequence $\mathbf{X}_{[1,v]}$ is achieved when both segments $\mathbf{X}_{[1,i-1]}$ and $\mathbf{X}_{[i,v]}$ are optimal. To minimise the function we utilise Bellman's equation [28], i. e.,

$$\mathcal{L}(v|\theta) = \min_{v-n_{max}<i\leq v}(\mathcal{L}(v-1|\theta) + \min_{\mathbf{y},\mathbf{s}} \mathcal{L}(\mathbf{y},\mathbf{s}|\theta)|\mathbf{x}_{[i,v]}) \tag{9}$$

where n_{max} is a constraint parameter that constraints the maximum length to search for an optimal segment starting at position i. Optimizing Eq. 9 can be performed in two steps: (a) the *forward pass*, where the clusters are kept fixed and temporal boundaries are determined, and (b) the *backward pass*, where the segments are kept fixed and the clusters are updated.

More particularly, in the *forward pass* the sequence is scanned from the beginning ($v = 1$) to its end ($v = N$), and for each position v new segments are constructed with starting position the $r = i - n_{max}$-th frame and ending the v frame, i.,e. $\mathbf{X}[r, v]$. The similarity between each of these segments and all segments in the clusters is computed. The label (y^*) and starting position (r^*) found to have the lowest error are stored for each position v. Figure 1 (right) shows an example of this step where at timestep $v = 20$ the segment $X_{[i,v]}$ is compared with all the segments in the clusters.

The *backward pass* starts from the end of the sequence, i.,e. $v = N$, and a new segment is created with starting position the r_N^* that was found in the forward pass. The method continues progressively from the starting position of each newly created segment and creates new segments until the beginning of the sequence, i. e. $v = 1$ (e. g. for the previous case it will start at position r_N^* to create the segment $X[r_{N-1}^*, r_N^*]$). For more details, the interested reader is referred to [27].

C. Representation Learning

To learn suitable representations from high-dimensional data we utilise a Convolution Neural Network (CNN) which comprises of convolution and max-pooling layers. Figure 1 (left) shows our architecture. To minimise our loss, we keep the segments and clusters fix and minimise only with respect to the parameters θ of the network, i. e.

$$\min_{\theta} \mathcal{L}(\theta|\mathbf{X}, \mathbf{y}, \mathbf{s}) \tag{10}$$

The CNN learns representations on a sequence of images $\mathbf{X}$ supervised by the cluster labels $\mathbf{y}$ that have been merged.

D. Recurrent Framework

We combine all of the parts of our method (clustering, temporal segmentation and representation learning) in a recurrent framework by partially unrolling the timesteps into multiple periods [23], where in each period we update the CNN parameters and the segments. The duration of each period p is $n_p = ceil(\eta \times n_c^{t_s})$ steps, where η is a hyperparameter and $n_c^{t_s}$ is the number of clusters at the beginning of the period.

The framework starts by initialising the segments and clusters. This can be accomplished either randomly or with temporal clustering algorithm like ACA. In our case we use random segmentation. After the initial segmentation is obtained, clusters are merged until the end of period p is reached. Then, based on the label clusters the parameters of our model are updated and the DTAK kernel matrix are recomputed. Finally, the segments are refined utilizing the forward-backward algorithm. This process is repeated until the desired number of clusters is reached.

V. Objective Function

The loss is accumulated over all timesteps and it can be computed as follows

$$\mathcal{L}(\mathbf{y}, \theta, \mathbf{s}|\mathbf{X}) = \sum_{t=1}^{T} \mathcal{L}^t(\mathbf{y}^t, \theta^t, \mathbf{s}^t|\mathbf{y}^{t-1}, \mathbf{s}^{t-1}, \mathbf{X}) \tag{11}$$

where y^0, s^0 indicates the initial clusters and segments, respectively, at timestep $t = 0$, and T is the total number of timesteps required for the algorithm to finish. We define the loss at each timestep t as

$$\begin{aligned}\mathcal{L}^t(\mathbf{y}^t, \theta^t, \mathbf{s}^t|\mathbf{y}^{t-1}, \mathbf{X}) &= -\mathcal{A}(\mathbf{C}_i^t, \mathbf{N}_{\mathbf{C}_i^t}^{K_c}[1]) \\ &- \frac{\lambda}{K_c - 1}\sum_{k=2}^{K_c}(\mathcal{A}(\mathbf{C}_i^t, \mathbf{N}_{\mathbf{C}_i^t}^{K_c}[1]) - \mathcal{A}(\mathbf{C}_i^t, \mathbf{N}_{\mathbf{C}_i^t}^{K_c}[k]))\end{aligned} \tag{12}$$

where λ is a weight, $\mathbf{N}_{\mathbf{C}_i^t}$ is a sorted vector of the closest clusters of cluster $\mathbf{C}_i$, and K_c are the nearest clusters of cluster $\mathbf{C}_i$. The first term of the loss measures the affinity of cluster $\mathbf{C}_i$ and its nearest neighbour, and the second term, which takes the local structure into account, measures difference in the affinity of $\mathbf{C}_i$ to its nearest neighbour and affinities of $\mathbf{C}_i$ to its other neighbour clusters.

The minimisation of this loss is accomplished in three steps. First agglomerative clustering is performed by fixing the parameters of the CNN and the segments. Second the clusters and the segments are fixed, and the CNN parameters are updated. Finally, both the segments and the clusters are refined by keeping the CNN parameters fixed. Particularly, the loss for each step is be defined as follows.

Cluster labels:

$$\mathcal{L}^p(\mathbf{Y}^p|\theta^p, \mathbf{X}) = \sum_{t=t_p^s}^{t_p^e} \mathcal{L}^t(\mathbf{y}^t|\theta^p, \mathbf{y}^{t-1}, \mathbf{X}) \tag{13}$$

Time Boundaries:

$$\mathcal{L}(\mathbf{Y}, \mathbf{s}|\theta, \mathbf{X}) = \sum_{k=1}^{p} \mathcal{L}^k(\mathbf{Y}^k, \mathbf{s}^k|\theta, \mathbf{X}) \tag{14}$$

CNN parameters:

$$\mathcal{L}(\theta|\mathbf{Y}, \mathbf{s}, \mathbf{X}) = \sum_{k=1}^{p} \mathcal{L}^k(\theta|\mathbf{Y}^k, \mathbf{s}^k, \mathbf{X}) \tag{15}$$

where t_p^s, t_p^e denote the start and end of the period respectively, and $\mathbf{Y}^p$ is the image label sequence at period p.

To compute the loss, the entire dataset is required and as such it is difficult to use batch-based optimisation. However, we can approximate this by computing affinities between data points for each period separately [23], i. e.,

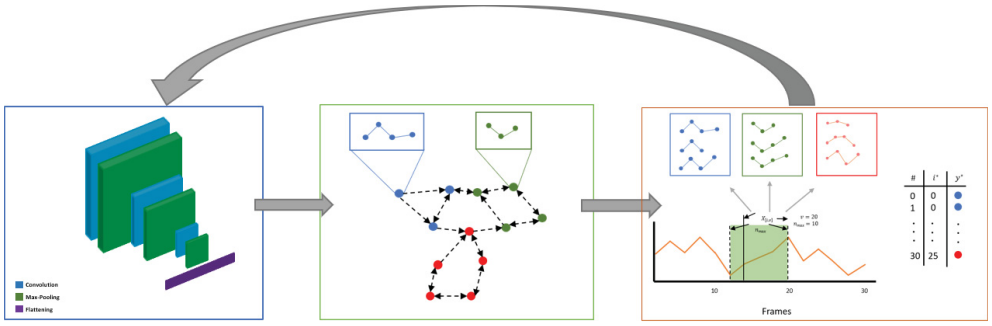

Fig. 1. Depicting the recurrent framework: (a) CNN network (left) , (b) agglomerative clustering (middle) and (c) temporal segmentation (right). - Best viewed in color

$$\mathcal{L}(\theta|\mathbf{y}^{t_p^e}, \mathbf{s}^{t_p^e}, X) = -\frac{\lambda}{K_c - 1}\sum_{ijk}(\gamma\mathcal{A}(\mathbf{x}_i, \mathbf{x}_j) - \mathcal{A}(\mathbf{x}_i, \mathbf{x}_k)) \tag{16}$$

where γ is a weight and $\mathbf{x}_i, \mathbf{x}_j$ are images from the same cluster and $\mathbf{x}_i, \mathbf{x}_k$ from different ones.

VI. Experiments

A. Dataset

Temporal clustering of human actions is tested on the CMU Multi-Modal Activity Database (CMU-MMAC) [5] database which contains human activity of subjects performing tasks involved in cooking and food preparation. Our method was tested on the first four videos where the subjects cook a brownie as these labels are available. Table I shows the number of frames and the number of clusters for each video.

TABLE I

Showing the subjects used from the CMU-MMAC database along with the number of frames and clusters of each video.

Subject ID	# Frames	# Clusters
S07	9800	32
S08	8699	30
S09	13107	34
S12	14832	33

B. Experimental Setup

We initialise the agglomerative clusteirng parameters as follows $\eta = 0.5$, the number of nearest clusters $K_c = 5$, $\lambda = 5$, and $\alpha = 1$.

For our experiments we used Tensorflow [1]. The CNN architecture is comprised three layers of convolution with ReLU activation and max-pooling between the convolution layers. The kernel of the convolution is 3×3, with stride 1 and padding 0, and the number of filters is set to 50. The max-pooling layer is comprised of a kernel of size 2×2 and stride 2. Finally, the last features from the last max-pooling layer are flatten and a L2-normalization layer is applied. Finally, the input to the CNN are images of size $80 \times 60 \times 3$.

We train the network using sequences of the data of length 100 so that neighbour frames are processed from the network. We use stochastic gradient descent with batch size of 3 with $momentum = 0.9$ and inverse learning rate decay with initial learning rate value of 0.01, with $gamma = 0.0001$ and $power = 0.75$.

C. Results

We compare our method with both temporal clustering alorithms, namely, ACA and HACA, the conventional clustering algorithm Gaussian Mixture Models (GMM), and a deep approach proposed by [22]. The inputs to the algorithms are the raw pixel intensities. Table II depicts the results for the CMU-MMAC database with respect to the NMI. Our method provides the best results in all four videos. As expected GMM algorithm provides the worst results.

TABLE II

Results on the CMU-MMAC database (wrt NMI).

Subject ID	GMM	ACA	HACA	[22]	Ours
S07	0.6530	0.6629	0.6684	0.501	**0.6839**
S08	0.6265	0.6653	0.6708	0.452	**0.6714**
S09	0.5111	0.5840	0.5481	0.428	**0.6171**
S12	0.5518	0.5725	0.5929	0.397	**0.6279**

VII. Conclusions and Future Works

In this paper, we propose a clustering algorithm that can jointly learn deep representations, clusters and temporal boundaries of dynamic data using raw intensity pixels. More particularly, we learn the parameters of a CNN network by utilising supervisory cues from the clustering-based loss function, and the temporal boundaries of the segments are learned with a coordinate-descent optimization algorithm. Our method provides the best results, with respect to normalised mutual information, compared to other clustering algorithms.

VIII. Acknowledgments

The support of the EPSRC Center for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1) is gratefully acknowledged. Dr. Zafeiriou acknowledges support from a Google Faculty award, as well as from the EPSRC fellowship Deform (EP/S010203/1).

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[4] B. L. Bhatnagar, S. Singh, C. Arora, C. Jawahar, and K. CVIT. Unsupervised learning of deep feature representation for clustering egocentric actions. In *International Journal of Artificial Intelligence*, pages 1447–1453, 2017.

[5] F. de la Torre, J. K. Hodgins, J. Montano, and S. Valcarcel. Detailed human data acquisition of kitchen activities: the cmu-multimodal activity database (cmu-mmac). In *CHI 2009 Workshop. Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*, 2009.

[6] D. Gong, G. Medioni, and X. Zhao. Structured time series analysis for human action segmentation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1414–1427, 2014.

[7] Z. Harchaoui, E. Moulines, and F. R. Bach. Kernel change-point analysis. In *Neural Information Processing Systems*, pages 609–616, 2009.

[8] B. Krüger, A. Vögele, T. Willig, A. Yao, R. Klein, and A. Weber. Efficient unsupervised temporal segmentation of motion data. *Transactions on Multimedia*, 19(4):797–812, 2017.

[9] W.-A. Lin, J.-C. Chen, and R. Chellappa. A proximity-aware hierarchical clustering of faces. In *FG*, pages 294–301. IEEE, 2017.

[10] C. Lu and N. J. Ferrier. Repetitive motion analysis: Segmentation and event classification. *Transactions on Pattern Analysis & Machine Intelligence*, (2):258–263, 2004.

[11] N. Lv, Y. Huang, Z. Feng, and J. Peng. A genetic algorithm approach to human motion capture data segmentation. *Computer Animation and Virtual Worlds*, 25(3-4):281–290, 2014.

[12] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems*, pages 849–856, 2002.

[13] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3):103–124, 2008.

[14] V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *Neural Information Processing Systems*, pages 981–987, 2001.

[15] Y. Rui and P. Anandan. Segmenting visual actions based on spatio-temporal motion patterns. In *Computer Vision & Pattern Recognition*, volume 1, pages 111–118. IEEE, 2000.

[16] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Computer Vision & Pattern Recognition*, pages 815–823, 2015.

[17] H. Shimodaira, K.-i. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Neural Information Processing Systems*, pages 921–928, 2002.

[18] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.

[19] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller. A deep matrix factorization method for learning attribute representations. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):417–429, 2017.

[20] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang. Learning a task-specific deep architecture for clustering. In *ICDM*, pages 369–377. SIAM, 2016.

[21] G. Xia, H. Sun, L. Feng, G. Zhang, and Y. Liu. Human motion segmentation via robust kernel sparse subspace clustering. *IEEE Transactions on Image Processing*, 27(1):135–150, 2018.

[22] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *arXiv preprint arXiv:1610.04794*, 2016.

[23] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Computer Vision & Pattern Recognition*, pages 5147–5156, 2016.

[24] B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *International Conference on Data Engineering*, pages 201–208. IEEE, 1998.

[25] L. Zelnik-Manor and M. Irani. Statistical analysis of dynamic actions. *Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1530–1535, 2006.

[26] W. Zhang, X. Wang, D. Zhao, and X. Tang. Graph degree linkage: Agglomerative clustering on a directed graph. In *ECCV*, pages 428–441. Springer, 2012.

[27] F. Zhou, F. De la Torre, and J. K. Hodgins. Aligned cluster analysis for temporal segmentation of human motion. In *Automatic Face and Gesture Recognition*, pages 1–7. IEEE, 2008.

[28] F. Zhou, F. De la Torre, and J. K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *Transactions on Pattern Analysis & Machine Intelligence*, 35(3):582–596, 2013.