Live and Learn: Continual Action Clustering with Incremental Views

Xiaoqiang Yan[1], Yingtao Gan[1], Yiqiao Mao[1], Yangdong Ye[1]*, Hui Yu[2]

[1] School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, China
[2] School of Creative Technologies, University of Portsmouth, PO1 2DJ, United Kingdom
iexqyan@zzu.edu.cn, ieytgan@outlook.com, ieyqmao@gs.zzu.edu.cn, ieydye@zzu.edu.cn, hui.yu@port.ac.uk

Abstract

Multi-view action clustering leverages the complementary information from different camera views to enhance the clustering performance. Although existing approaches have achieved significant progress, they assume all camera views are available in advance, which is impractical when the camera view is incremental over time. Besides, learning the invariant information among multiple camera views is still a challenging issue, especially in continual learning scenario. Aiming at these problems, we propose a novel continual action clustering (CAC) method, which is capable of learning action categories in a continual learning manner. To be specific, we first devise a category memory library, which captures and stores the learned categories from historical views. Then, as a new camera view arrives, we only need to maintain a consensus partition matrix, which can be updated by leveraging the incoming new camera view rather than keeping all of them. Finally, a three-step alternate optimization is proposed, in which the category memory library and consensus partition matrix are optimized. The empirical experimental results on 6 realistic multi-view action collections demonstrate the excellent clustering performance and time/space efficiency of the CAC compared with 15 state-of-the-art baselines.

Introduction

Recognizing human actions from videos is a fundamental and active research topic in various applications. In recent years, action recognition has achieved significant attentions and promising progresses due to the explosive growth of action data and the advancement of relevant learning models. Usually, most existing methods focus on supervised learning scenarios which leverage sufficient data labels to train learning models. However, labelling large amounts of data is time-consuming and labor-intensive. In addition, due to the different backgrounds and prior knowledge of annotators, they cannot assure to uniformly assign action labels to similar video samples. Thus, it is necessary to resort to unsupervised clustering approaches to automatically learn action categories.

In last decades, numerous action clustering approaches have been devised and achieved some satisfactory performances. For instance, (Jones and Shao 2014) designs a dual assignment k-means to perform two co-occurring action clustering tasks. (Bhatnagar et al. 2017) proposes a generic unsupervised approach to learn feature representation for first person action clustering. (Yan, Hu, and Ye 2017) proposes to perform action clustering in a multi-task clustering setting. (Peng et al. 2020) proposes to improve the performance of action clustering with contextual information of actions in a recursive constrained framework. (Kumar et al. 2022) performs representation learning and online action clustering for unsupervised activity segmentation. However, existing action clustering faces the following issues: 1) Difficulty in action category discovery. Most current approaches focus on making better use of spatio-temporal information to describe the actions. However, discovering action categories remains a challenging problem due to camera motions, changes of viewpoints, cluttered background. 2) Ignoring the correlations between multiple camera views. Due to the occlusion and deformation of the viewpoints, single-view action clustering cannot effectively capture the complementary information between multiple views. For example, it is impossible to recognize applauding solely from the camera behind a person. Thus, it is natural to perform multi-view clustering on the task of recognizing human actions.

Multi-view clustering (MVC) leverages the complementary information from different camera views to enhance the performance of action clustering (Xia et al. 2022). According to the strategy of complementary information extraction, existing MVC approaches can be classified into the following types: subspace clustering (Zhang et al. 2019; Mao et al. 2021; Tang et al. 2023; Yan et al. 2023a; Xu et al. 2023; Zhao, Yang, and Nie 2023; Yan et al. 2023c), consensus clustering (Wang et al. 2019; Liu et al. 2021; Huang, Wang, and Lai 2023; Yan et al. 2023b) and multi-kernel clustering (Liu et al. 2016; Guo and Ye 2019; Zhang et al. 2023; Liu 2023). Although existing MVC approaches have achieved some promising results, they assume that all camera views are available in advance, which is impractical when the camera view is incremental over time. Besides, learning the invariant information among multiple camera views is a difficult and challenging issue in continual learning scenario due to the following reasons. First, there is a large divergence between historical and new coming views usually. Second, the new coming views are often unknown and unpredictable

*Corresponding author

to historical views. Third, it is impractical to store all historical views for directly employing existing MVC approaches to re-access all views simultaneously.

To solve these problems, we propose a novel continual action clustering (CAC) method, which is capable of achieving never-ending knowledge transfer between historical views and the new coming ones (as shown in Figure 1). To achieve this target, we first design a category memory library, which learns and stores the learned action categories from historical views. When a new camera view arrives, CAC constructs its basic clustering partitions based on kernel learning. Then, we only need to maintain a consensus partition matrix for historical views, which can be updated by leveraging the incoming new camera view rather than keeping all of them. Finally, a three-step alternate optimization is proposed, in which the category memory library and consensus partition matrix are alternately optimized. Extensive experimental results demonstrate the clustering performance and time/space efficiency of the proposed CAC. The contributions of this study can be summarized as follows:

- A novel continual action clustering method named CAC is proposed, which can achieve never-ending knowledge transfer between historical views and the new coming ones, and improve the clustering performance accordingly. To the best of our knowledge, this is the first work to explore the task of multi-view action clustering in incremental learning scenario.
- We design a category memory library to learn and store the learned action categories from historical views. When new action view is coming, we only need to maintain a consensus partition matrix for historical views, which can be updated by leveraging the incoming new camera view rather than keeping all of them.
- A three-step alternate optimization is proposed, in which the category memory library and consensus partition matrix are alternately optimized. The experimental results strongly support the proposed CAC method.

Related Works and Backgrounds

Incremental Multi-view Clustering

In recent several years, there are few efforts towards solving incremental MVC so far (Zhou et al. 2019; Yin et al. 2021; Sun et al. 2022; Wan et al. 2022). Specifically, (Zhou et al. 2019) divides all views into several incremental subspaces and then performs k-means iteratively by increasing the size of the subspaces. (Yin et al. 2021) extends spectral clustering to incremental multi-view clustering setting, which finds a ensemble kernel from existing basic kernels and a new coming view. (Wan et al. 2022) first constructs a consensus similarity matrix of all available views, and reconstructs the consensus similarity matrix when newly views are available. Despite the success of existing incremental MVC, they seem far from fully being explored and can still be improved from the following considerations. First of all, with limited computational resources, it would be difficult to access and recompute all samples in historical views. Secondly, the new coming views are often unknown and unpredictable to historical views, so it is challenging to capture the large divergence between them.

Multi-kernel k-means

Traditional k-means cannot effectively deal with the feature transformation of nonlinear data. Thus, kernel k-means (Liu 2023) has been proposed to map nonlinear data into a high dimensional space, in which the data can be separated. Kernel k-means has also been applied to the task of multi-view clustering and obtained promising performance, i.e., multi-kernel k-means (Wang et al. 2019; Liu et al. 2021).

Let $\mathbf{X} = \{X^i\}_{i=1}^m$ be a data collection with m views, each view contains n data samples $\{\mathbf{x}_j\}_{j=1}^n$, where n is the number of data samples. We use $\phi_p(\cdot) : \mathbf{x} \in \mathcal{X} \rightarrow \mathcal{H}_p$ to denote the mapping function that transforms the data samples in the p-th view into reproducing kernel Hilbert space $\mathcal{H}_p (1 \leq p \leq m)$. In multi-kernel condition, the data samples of the m views can be represented as $\phi_{\boldsymbol{\beta}}(\mathbf{x}) = [\beta_1\phi_1(\mathbf{x})^T, \cdots, \beta_m\phi_m(\mathbf{x})^T]^T$, where $\boldsymbol{\beta} = [\beta_1, \cdots, \beta_m]^T$ is the coefficients of m basic kernels $\{\mathcal{K}_p(\cdot,\cdot)\}_{p=1}^m$. According to the definition of $\phi_{\boldsymbol{\beta}}(\mathbf{x})$, the kernel function $\mathcal{K}_{\boldsymbol{\beta}}(\mathbf{x}_i, \mathbf{x}_j)\}$ can be defined as follows,

$$\mathcal{K}_\beta(\mathbf{x}_i, \mathbf{x}_j) = \phi_\beta(\mathbf{x}_i)^T\phi_\beta(\mathbf{x}_j) = \sum_{p=1}^m \beta_p^2\mathcal{K}_p(\mathbf{x}_i, \mathbf{x}_j) \quad (1)$$

where a kernel matrix $\mathbf{K}_{\boldsymbol{\beta}}$ can be obtained by applying the kernel function on the data collection $\mathbf{X} = \{X^i\}_{i=1}^m$.

Multi-kernel k-means supposes that the optimal kernel matrix can be characterized by a linear combination of multiple kernel matrices, i.e., $\mathbf{K}_{\boldsymbol{\beta}} = \sum_{p=1}^m \beta_p^2\mathbf{K}_p$. Thus, we can obtain the objective function of multi-kernel k-means,

$$\begin{aligned} &\min_{\mathbf{H},\boldsymbol{\beta}} \mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\beta}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^T)\right), \\ &\mathrm{s.t.}\mathbf{H} \in \mathbb{R}^{n\times k}, \mathbf{H}^T\mathbf{H} = \mathbf{I}_k, \boldsymbol{\beta}^T\mathbf{1}_m = 1, \boldsymbol{\beta}_p \geq 0, \forall p. \end{aligned} \quad (2)$$

where $\mathbf{H}$ is the clustering partition matrix, which can be obtained by alternately optimizing $\boldsymbol{\beta}$ and $\mathbf{H}$.

Late Fusion Multi-view Clustering

Recently, late fusion MVC methods have been proposed to address the issues of high complexity and cumbersome optimization process of multi-kernel k-means. Differently, late fusion MVC separately performs kernel k-means or spectral clustering on the kernel matrix $\{\mathbf{K}\}_{p=1}^m$ to generate basic clustering partitions $\mathbf{H} = \{\mathrm{H}_p\}_{p=1}^m$ for each view. Then, it searches an optimal partition matrix as follows,

$$\begin{aligned} &\max_{\mathbf{H},\boldsymbol{\beta}} \mathrm{Tr}(\mathbf{H}^{*T}\mathbf{X}) + \lambda\mathrm{Tr}(\mathbf{H}^{*T}\mathbf{M}), \\ &\mathrm{s.t.}\mathbf{H}^{*T}\mathbf{H}^* = \mathbf{I}_k, \mathbf{W}_p^T\mathbf{W}_p = \mathbf{I}_k, \\ &\sum_{p=1}^m \beta_p^2 = 1, \beta_p \geq 0, \mathbf{X} = \sum_{p=1}^m \beta_p\mathbf{H}_p\mathbf{W}_p. \end{aligned} \quad (3)$$

where β_p is the weight of the p-th view, $\mathbf{W}_p$ is the permutation matrix of the p-th view which can align the basic partition matrices of different views. $\mathbf{M}$ is the averaged partition

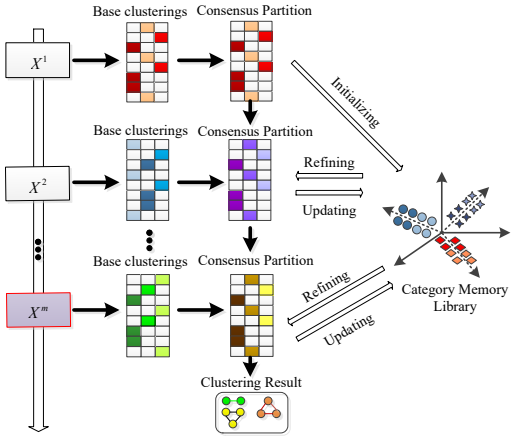


Figure 1: Framework of CAC. When a new view arrives, we only need to maintain a consensus partition matrix to achieve never-ending knowledge transfer between historical and new coming views with the category memory library.

matrix, $\mathbf{X} = \sum_{p=1}^{m} \beta_p \mathbf{H}_p \mathbf{W}_p$ is the linear combination of the partition matrices from different views. $\mathrm{Tr}(\mathbf{H}^{*T}\mathbf{M})$ is a regularization term that prevents the optimal partition matrix from being too far away from the average partition matrix $\mathbf{M}$.

Continual Action Clustering with Incremental Views

In this section, we first devise a category memory library to learn and store the learned action categories from historical views. Then, the objective function of CAC is provided, which is solved by a three-step alternate optimization. Finally, the convergence and complexity is analyzed.

Problem Statement

Given a data collection with m views $\mathbf{X} = \{X^i\}_{i=1}^m$, each view contains n data samples $\{\mathbf{x}_j\}_{j=1}^n$. The original intention of late fusion multi-view clustering based on multi-kernel k-means is to uncover the correlations among all views and generate the final partition matrix $\mathbf{H}^*$ by combining the basic partition matrices $\mathbf{H} = \{\mathrm{H}_p\}_{p=1}^m$ of each view. However, this learning strategy may be impractical in the incremental scenario of multi-view action clustering, in which the camera view is incremental over time. In this study, we propose a continual action clustering (CAC) method with incremental views. In incremental multi-view clustering, the CAC encounters a series of views $X^1, X^2, \cdots, X^m$. When it receives a view of data for clustering in each period, this system should meet the following requirements. 1) **Continual clustering**. It should achieve never-ending knowledge transfer between historical views and the new coming ones. 2) **Clustering performance**. The final clustering partition matrix $\mathbf{H}^*$ by continually combining the basic ones $\mathbf{H} = \{\mathrm{H}_p\}_{p=1}^m$ should be better that the clustering quality of traditional multi-view clustering. 3) **Computational speed**. New basic clustering assignment should be arbitrarily and efficiently added when the continual clustering system faces with new views.

Category Memory Library

To conduct the action clustering with incremental views, an important issue is to learn and store the categories (or clustering assignments) from historical views. The objective function Eq. 3 is a widely used formulation for late fusion multi-view clustering. Except for its simplicity and efficiency, it can also maximize the alignment of the optimal partition matrix by linearly combining the basic partition matrices for each view. However, the linear combination fails to integrate the basic partition matrices one by one and cannot reveal a more intrinsic clustering structure during the optimization process, which means that it cannot be directly applied to incremental multi-view clustering.

To solve this issue, we devise a category memory library to learn and store the latent cluster centroids of the basic partition matrix learned from the historical views. Specifically, we decompose the optimal partition matrix in Eq. 3 into two sub matrices as follows,

$$\mathbf{H}^* = \mathbf{E}\mathbf{B}, \tag{4}$$

where $\mathbf{E} \in \{0,1\}^{n\times k}$ indicates which category the data belongs to, $\mathbf{B} \in \mathbb{R}^{k\times k}$ is our category memory library that stores the cluster centroids in optimal clustering partition matrix. By storing and optimizing the category memory library, it can serve continual action clustering. According to Eq. 3 and Eq. 4, we can formulate the objective function of late fusion multi-view clustering with a category memory library as follows,

$$\begin{aligned} &\max_{\mathbf{E},\mathbf{B},\{\mathbf{W}_p\}_{p=1}^m,\boldsymbol{\beta}} \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\mathbf{X}) + \lambda \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\mathbf{M}), \\ &\mathrm{s.t.}\mathbf{B}^T\mathbf{B} = \mathbf{I}_k, \mathbf{W}_p^T\mathbf{W}_p = \mathbf{I}_k, \mathbf{E} \in \{0,1\}^{n\times k}, \\ &\sum_{p=1}^{m} \beta_p^2 = 1, \beta_p \le 0, \mathbf{X} = \sum_{p=1}^{m} \beta_p \mathbf{H}_p \mathbf{W}_p. \end{aligned} \tag{5}$$

The orthogonal constraints imposed on $\mathbf{B}$ make each cluster center of $\mathbf{B}$ independent and orthogonal to each other, facilitating subsequent optimization. Thus, the category memory library $\mathbf{B}$ can be used to store more refined clustering structures, achieving better clustering results while preserving learned knowledge from historical views.

Continual Action Clustering

Although Eq. 5 can store the learned action categories, it still requires to fuse the basic clustering partition matrices of all views at once without considering the significant consumption of space and time in the case of incremental multi-view clustering. As discussed, existing multi-view action clustering methods improve the clustering performance by fusing information from all views. However, such methods cannot handle the situations where the camera views continue to increase over time. For example, we can obtain new action data when a new camera is installed. Traditional methods require to fuse all views in one time, which is impractical since it wastes a lot of time and space resources.

To address this issue, the proposed CAC creatively combines multi-view action clustering with continual learning. Inspired by the multi-view clustering approaches based on the late fusion strategy, the proposed CAC only needs to maintain a consensus partition matrix, which can be updated by leveraging the incoming new camera view rather than keeping all of them.

Suppose we have collected the action data of the t-th view camera, the basic clustering partition matrix $\mathbf{H}_t$ can be generated by applying kernel k-means on the t-th view. At this time, suppose we already have the consensus partition matrix $\mathbf{H}^*_{t-1}$ of the previous $t-1$ cameras. To improve the efficiency of incremental MVC, we just use the optimal permutation matrix $\mathbf{W}_t$ of the basic partition matrix $\mathbf{H}_t$ to align the partition matrices. Besides, we treat the consensus partition matrix $\mathbf{H}^*_{t-1}$ of the previous $t-1$ view cameras as the regularization term rather than the averaged partition matrix $\mathbf{M}$. Thus, the optimal permutation matrix $\mathbf{W}_t$ can be obtained by learning the basic partition $\mathbf{H}_t$ after a linear transformation $\mathbf{H}_t\mathbf{W}_t$ as follows,

$$\begin{aligned} &\max_{\mathbf{E},\mathbf{B},\mathbf{W}_t} \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\mathbf{H}_t\mathbf{W}_t) + \lambda \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\mathbf{H}^*_{t-1}), \\ &\mathrm{s.t.}\mathbf{B}^T\mathbf{B} = \mathbf{I}_k, \mathbf{W}_t^T\mathbf{W}_t = \mathbf{I}_k, \mathbf{E} \in \{0,1\}^{n\times k} \end{aligned} \tag{6}$$

As shown in Eq. 6, the correlation between linear transformation $\mathbf{H}_t\mathbf{W}_t$ and the consensus partition matrix $\mathbf{H}^*_{t-1}$ can be further refined by the category memory library $\mathbf{B}$. After obtaining the optimal permutation matrix $\mathbf{W}_t$, CAC method can combine $\mathbf{H}_t\mathbf{W}_t$ and $\mathbf{H}^*_{t-1}$ as follows,

$$\mathbf{H}^*_t = \mathbf{H}^*_{t-1} + \mathbf{H}_t\mathbf{W}_t \tag{7}$$

By normalizing $\tilde{\mathbf{H}}^*_{t-1} = \mathbf{H}^*_t/(t-1)$, the final version of the CAC objective function can be formulated as follows,

$$\begin{aligned} &\max_{\mathbf{E},\mathbf{B},\mathbf{W}_p} \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\mathbf{H}_t\mathbf{W}_t) + \lambda \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\tilde{\mathbf{H}}^*_{t-1}), \\ &\mathrm{s.t.}\mathbf{B}^T\mathbf{B} = \mathbf{I}_k, \mathbf{W}_t^T\mathbf{W}_t = \mathbf{I}_k, \mathbf{E} \in \{0,1\}^{n\times k} \end{aligned} \tag{8}$$

Now, the proposed CAC can deal with the views one by one with a category memory library.

Three-step Alternate Optimization

In the objective function Eq. 8, there are three terms that needs to be optimized, i.e., $\mathbf{E}$, $\mathbf{B}$ and $\mathbf{W}_t$. To this end, we design a three-step alternate optimization to solve them.

Optimizing E Fixing $\mathbf{B}$ and $\mathbf{W}_t$, the optimization of Eq. 8 can be formulated as follows,

$$\max_{\mathbf{E}} \mathrm{Tr}(\mathbf{E}\mathbf{A}^T) \mathrm{s.t.} \mathbf{E} \in \{0,1\}^{n\times k} \tag{9}$$

where $\mathbf{A} = (\mathbf{H}_t\mathbf{W}_t + \lambda\tilde{\mathbf{H}}^*_{t-1})\mathbf{B}^T$. Thus, the optimal $\mathbf{E}$ can be represented as follows

$$\mathbf{E}(i,j) = 1 \tag{10}$$

where $j = \arg\max A(i,:)$.

Algorithm 1: Continual Action Clustering (CAC) Algorithm

Input: The basic partition matrices $\{\mathbf{H}_t\}_{t=1}^m$ tasks with T datasets $X = \{X^1, X^2, \cdots, X^T\}$, categories number k, regularization parameter λ, convergence boundary ε
Initialization: Initialize the category memory library $\mathbf{B}$ by performing k-means on $\mathbf{H}_1$
Initialize the consensus partition matrix by $\mathbf{H}^*_1 = \mathbf{H}_1$
for $t = 2$ to m **do**
  $\tilde{\mathbf{H}}^*_{t-1} = \mathbf{H}^*_t/(t-1)$
  Initialize $W_t = I_k$, $i = 1$
  while not convergence **do**
    Fix $\mathbf{B}$ and $\mathbf{W}_t$, update $\mathbf{E}$ by solving Eq. 9
    Fix $\mathbf{E}$ and $\mathbf{W}_t$, update $\mathbf{B}$ by solving Eq. 11
    Fix $\mathbf{E}$ and $\mathbf{B}$, update $\mathbf{W}_t$ by solving Eq. 13
    $i = i + 1$
  end while$(obj^i - obj^{i-1})/obj^i \le \varepsilon$
  Update $\mathbf{H}^*_t$ via Eq. 7
end for
Output: The final consensus partition matrix $\mathbf{H}^*_m$.

Optimizing B Fixing $\mathbf{E}$ and $\mathbf{W}_t$, the optimization of CAC objective function Eq. 8 can be formulated as follows,

$$\max_{\mathbf{B}} \mathrm{Tr}(\mathbf{B}^T\mathbf{D}) \mathrm{s.t.} \mathbf{B}^T\mathbf{B} = \mathbf{I}_k, \tag{11}$$

where $\mathbf{D} = E^T(\mathbf{H}_t\mathbf{W}_t + \lambda\tilde{\mathbf{H}}^*_{t-1})$. The matrix $\mathbf{D}$ can be formulated as $\mathbf{D} = \mathbf{S}\sum\mathbf{V}^T$ by singular value decomposition (SVD), the optimization of $\mathbf{B}$ can be rewritten as follows

$$\mathbf{B} = \mathbf{S}\mathbf{V}^T \tag{12}$$

Optimizing $\mathbf{W}_t$ Fixing $\mathbf{E}$ and $\mathbf{B}$, the optimization of CAC objective function Eq. 8 can be formulated as follows,

$$\max_{\mathbf{W}_t} \mathrm{Tr}(\mathbf{W}_t^T\mathbf{R}) \tag{13}$$

where $\mathbf{R} = \mathbf{H}_t^T\mathbf{E}\mathbf{B}$. Similar with Eq. 11, Eq. 13 can also be solved by SVD.

Now, we present the optimization process in Algorithm 1.

Theoretical Analysis

Convergence The objective function Eq. 8 can be separated into two terms as follows

$$\begin{cases} \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\mathbf{H}_t\mathbf{W}_t) \\ \mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\tilde{\mathbf{H}}^*_{t-1}) \end{cases} \tag{14}$$

By Cauchy-Schwartz inequality (Bhatia and Davis 1995), we know $\mathrm{Tr}(\mathbf{B}^T\mathbf{E}^T\mathbf{H}_t\mathbf{W}_t) \le ||\mathbf{B}^T\mathbf{E}^T||_F||(\mathbf{H}_t\mathbf{W}_t)||_F = k$, where k is the number of categories.

For the second term, $\mathbf{B}^T\mathbf{E}^T\tilde{\mathbf{H}}^*_{t-1} \le ||\mathbf{B}^T\mathbf{E}^T||_F||\tilde{\mathbf{H}}^*_{t-1}||_F$, since $||\tilde{\mathbf{H}}^*_{t-1}||_F$ is a constant, we can obtain that $||\mathbf{B}^T\mathbf{E}^T||_F||\tilde{\mathbf{H}}^*_{t-1}||_F \le a\sqrt{k}$ where $||\tilde{\mathbf{H}}^*_{t-1}||_F = a$. Thus, we can obtain the upper bound of the CAC objective function Eq. 8 as $k + a\sqrt{k}$. Besides, because our optimization method solves one variable while keeping other variables

Figure 2: Exemplar frames in WVU dataset

fixed, resulting in a monotonic increase in the value of objective function. Therefore, our algorithm is convergent. The experimental evidence also verifies the convergence of our algorithm.

Complexity According to the optimization process in Algorithm, the time complexity in each iteration is $O(nk^2+n)$, where n is the number of data samples, k is the number of clusters. There are m views in the multi-view clustering setting, so the final time complexity of CAC algorithm can be seen as $O(lm(nk^2+n))$, where l is the iteration number until convergence. For the space complexity, CAC algorithm only needs to store the category memory library $\mathbf{B}$ and consensus partition matrix $\mathbf{H}_t^*$. Thus, the space complexity is $O(nk+k^2)$.

Experiments

Datasets

We adopt the following widely-used multi-view human action datasets to evaluate the effectiveness of the CAC. They are: **IXMAS** (Ramagiri, Kavi, and Kulathumani 2011) consists of 11 different action categories, which has a total of 1, 695 video clips that recoded from 5 viewpoints. Due to the unavoidable partial occlusion, we select 4 views except for the top-down view. **MMI** (Liu et al. 2017) consists of 22 interactive action categories with 1760 clips recorded in dark, light and cluttered environments. **MSR** (Xu et al. 2016) consists of 20 human action categories, each category is performed 2 or 3 times by 10 subjects. **WVU** (Weinland, Boyer, and Ronfard 2007) contains 11 action categories and each category consists of 65 view clips. WVU is recorded from 8 cameras organized in a rectangular region. In this study, we select the actions that recorded non adjacent cameras, i.e., camera 2, 4, 6, 8, to make the incremental MVC more challenging. **UCLA** (Wang et al. 2014) consists of 10 daily actions recorded by 3 cameras. In total, it consists of 1475 RGB videos with depth and skeleton information. **MV-TJU** (Liu et al. 2015) consists of 22 human action categories and each category is performed 2 times by 20 subjects. Totally, it contains 3520 sequences with depth and skeleton modalities. Figure 2 shows exemplar frames in WVU dataset.

Experimental Setup

We implement the deep MVC baselines on PyTorch toolbox. Our proposed CAC and other multi-view clustering baselines are conducted on Matlab 2022a. All experiments are performed on a desktop computer with RTX 4090 GPU, i7-13700K CPU, 64G RAM, Windows 10 system.

Baselines To fully demonstrate the effectiveness of the proposed CAC method, we compare it with the following baselines: late fusion MVC (LFMVC, IJCAI'19) (Wang et al. 2019), one pass late fusion MVC (OPMVC, ICML'21) (Liu et al. 2021), fast multi-view clustering via ensembles (Fastmice, TKDE'23) (Huang, Wang, and Lai 2023), multi-view subspace clustering via adaptive graph learning and late fusion alignment (AGLLFA, NN'23) (Tang et al. 2023), multi-view clustering with local kernel alignment (LKA, TMM'23) (Zhang et al. 2023), simple multi-kernel k-means (SMKKM, TPAMI'23) (Liu 2023), binary multi-view clustering (BMVC, TPAMI'19) (Zhang et al. 2019), anchor-based partial multi-view clustering (APMC, AAAI'19) (Guo and Ye 2019), multi-kernel k-means clustering with matrix-induced regularization (Mir, AAAI'16) (Liu et al. 2016), global and cross-view feature aggregation (GCFAgg, CVPR'23) (Yan et al. 2023a), adaptive feature projection with distribution alignment (APADC, TIP'23) (Xu et al. 2023), auto-weighted orthogonal and nonnegative graph reconstruction (AONGR, INS'23) (Zhao, Yang, and Nie 2023), lifelong clustering (L2SC, TPAMI'22) (Sun et al. 2022), incremental multi-view spectral clustering (SCGL, NN'21) (Yin et al. 2021), continual multi-view clustering (CMVC, MM'22) (Wan et al. 2022). Specifically, GCFAgg, APADC and AONGR are deep MVC baselines, L2SC, SCGL and CMVC are incremental MVC baselines and the remaining ones are MVC approaches based on shallow learning models.

Parameter Setting In our model, there is a regularization parameter λ that balances the weights between historical and new coming views. Similar with the late fusion MVC baselines (CMVC, LFMVC, LKA), we tune λ in the range of $2.\hat{}[-10,-9,\cdots,9,10]$. We perform all baselines according to the original parameter settings from their papers, and the source codes are available from original authors or websites. We adopt ACC, NMI and Purity to measure the clustering performance as in (Zhang et al. 2019; Liu 2023; Zhang et al. 2023). Higher value indicates better clustering performance. For fairness, we run all methods 10 times and report the average results.

Experimental Results and Analysis

Table 1 shows the comparison results of CAC and baselines on 6 human action data collections. We can obtain the following observation from these tables.

1) The proposed CAC outperforms the traditional MVC baselines. For example, as an extension of late fusion MVC (LFMVC), the proposed CAC obtains 3.03%, 3.52%, 7.18%, 8.16%, 12.5% and 5.51% improvements respectively on the 6 datasets used in this paper. This is mainly because the proposed CAC well capture the consistency between multiple views in a continual clustering manner.

2) Recently, utilizing DNNs to boost the performance of MVC has received attractive attentions. To further verify the effectiveness of the CAC, we compare it with several latest state-of-the-art deep MVC baselines. From Table 1, we can observe that the CAC also performs better than the deep MVC baselines. For example, the CAC obtains 5.16%,

	IXMAS			MMI			MSR			WVU			UCLA			MVTJU		
	ACC	NMI	Purity	ACC	NMI	Purity	ACC	NMI	Purity	ACC	NMI	Purity	ACC	NMI	Purity	ACC	NMI	Purity
LFMVC	65.76	**66.56**	65.76	43.44	53.97	44.61	41.88	48.21	45.31	65.69	62.42	65.69	32.29	37.54	36.72	68.18	79.63	71.16
OPMVC	62.73	64.65	64.85	39.23	49.80	42.15	40.63	47.46	45.31	49.85	50.14	52.46	40.89	42.30	43.75	64.89	75.78	65.82
Fastmice	56.67	61.05	59.57	38.19	49.8	40.42	40.62	48.11	45.18	58.46	62.10	60.58	37.13	43.55	42.44	64.43	77.68	66.57
AGLLFA	60.30	64.73	61.82	45.67	56.45	48.36	39.69	41.49	40.31	65.23	66.76	66.77	42.45	44.80	47.92	65.51	75.86	67.67
LKA	65.45	64.49	65.45	45.20	54.40	46.72	46.56	50.23	47.81	68.92	65.52	69.08	44.01	43.87	45.31	71.56	79.16	73.10
SMKKM	63.56	65.21	64.38	44.60	55.46	46.86	41.72	47.59	44.62	65.18	61.33	65.22	32.06	36.53	36.82	68.47	79.46	70.58
BMVC	57.64	60.32	63.12	44.26	55.14	47.12	40.32	44.36	44.23	56.12	53.12	65.12	30.26	31.26	34.26	68.45	78.56	70.78
APMC	58.79	61.50	65.76	46.60	56.41	48.12	42.19	45.97	43.44	55.69	56.51	64.00	29.95	31.56	34.11	70.94	79.68	75.54
MIR	62.12	63.64	62.73	43.44	53.97	44.61	42.81	46.62	46.56	65.85	62.59	65.54	32.55	37.87	36.98	69.80	79.46	70.57
GCFAgg	58.48	60.61	59.09	39.23	51.47	42.51	37.19	42.00	38.12	60.31	66.78	63.85	44.95	43.73	47.03	67.87	77.34	67.98
APADC	55.63	58.36	57.45	40.78	52.15	43.21	40.36	44.36	40.19	63.15	61.36	59.36	43.69	44.43	43.26	63.54	71.35	67.46
AONGR	43.96	49.29	45.15	29.31	40.60	30.98	39.06	46.80	42.80	39.38	55.74	44.92	38.02	44.46	44.27	63.40	77.67	66.70
L2SC	61.67	62.11	62.34	44.67	55.24	47.42	41.72	46.77	44.06	64.46	64.64	65.42	27.34	23.36	29.94	69.87	79.06	71.12
SCGL	61.52	61.85	61.82	42.27	54.30	43.21	40.63	45.29	44.06	66.00	66.60	66.77	36.20	37.88	40.36	68.21	79.30	69.72
CMVC	63.33	63.58	63.94	44.50	54.77	46.84	46.88	51.83	49.06	69.69	65.17	69.69	42.71	44.32	47.14	71.99	79.94	72.87
CAC	**68.79**	65.77	**68.79**	**46.96**	**56.63**	**48.48**	**49.06**	**52.30**	**51.25**	**73.85**	**70.53**	**73.85**	**44.79**	**48.52**	**48.44**	**73.69**	**80.30**	**74.18**

Table 1: Comparisons on the 6 datasets in terms of ACC, NMI and Purity (mean results). Bold font denotes best values.

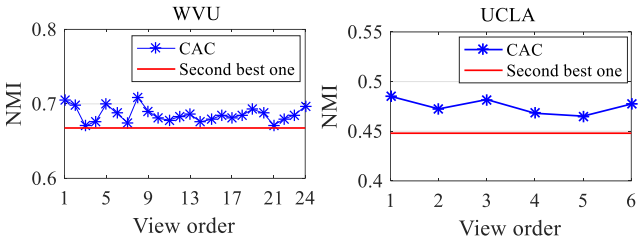


Figure 3: Impact of view order on CAC.

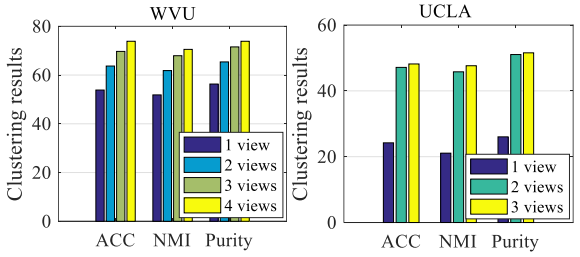


Figure 4: Impact of view number on CAC.

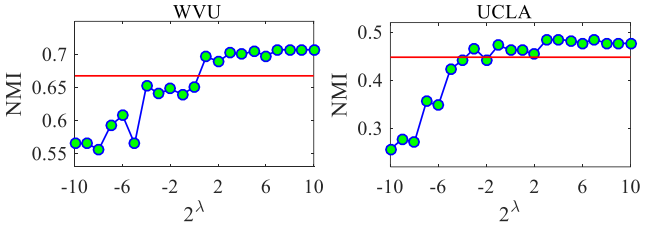


Figure 5: Various parameter λ on CAC.

7.41%, 16.48% improvements compared with GCFAgg, A-PADC and AONGR on IXMAS dataset in terms of NMI. This phenomenon further verifies the effectiveness of the CAC.

3) Compared with other incremental MVC baselines (L2SC, SCGL and CMVC), the CAC also achieves better performance or even a significant improvement on the 6 human action datasets. For example, CAC obtains 7.12%, 7.27% and 5.46% improvements compared with L2SC, SCGL and CMVC respectively on the IXMAS dataset in term of ACC metric. This is mainly because the knowledge of historical views is easily forgotten in the SCGL and CMVC, since they do not have a knowledge library to dynamically update the knowledge of historical views. L2SC requires to maintain two banks, i.e., feature bank and clustering partition bank, which is time-consuming making (see the subsection of running time analysis) and it is impratical for human action clustering applications. In this study, we design a category memory library. When new action view is coming, we only need to maintain a consensus partition matrix, which can be updated by leveraging the incoming new camera view rather than keeping all of them.

Impact of View Order

In the scenario of incremental MVC, the views are incremental over time. Thus, it is necessary to investigate the impact of the order of view arrival on the clustering performance of the CAC. For example, given a multi-view data collection with m views, there are $m! = m*(m-1)*\cdots*1$ number of orders that integrates the views. In this subsection, we report the clustering results of CAC with different view orders on WVU and UCLA datasets. As shown in Figure 3, the CAC outperforms consistently the second best baselines on these two datasets. We can conclude that the view order has little impact on the final clustering performance.

Impact of View Number

Since the CAC is an incremental MVC approach, which is capable of dealing with views over time. Ideally, as a new view joins, the clustering performance of incremental multi-

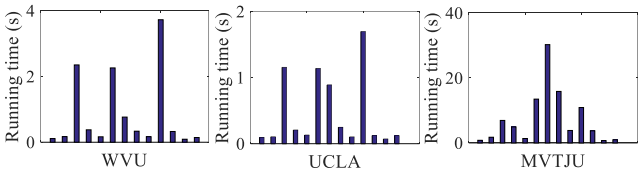
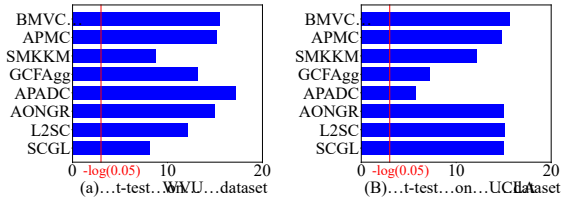


Figure 6: Running time comparison of our CAC with other baselines. In each sub-figure, the baselines in x-axis are LFMVC, OPMVC, Fastmice, AGLLFA, LKA, SSMKKM, BMVC, APMC, MIR, L2SC, SCGL, CMVC and our CAC.



(a) WVU (b) UCLA

Figure 7: Significance t-test on WVU and UCLA (ACC).

view clustering method should increase rather than decrease. In this subsection, we investigate the impact of view number on the clustering performance of CAC. We only report the experimental results on WVU and UCLA datasets due to page limit. From Figure 4, we can see that the performance of CAC unintentionally improves as the view number increases. This phenomenon verifies that the CAC can well leverage the knowledge of historical views to assist the clustering of new coming views.

Parameter Investigation

In this subsection, we conduct experiments to investigate the impact of the regularization parameter λ on the clustering performance of CAC method. Specifically, we vary the value of λ in the range of $2.^\wedge[-10, -9, \cdots, 9, 10]$. We only report the experimental results on WVU and UCLA due to page limit. Figure 5 provides the comparisons of CAC with the second best baseline. From this figure, we can observe that CAC trends to be better and stable when λ takes large value in the range. This experiment demonstrates that the CAC is stable and does not depend on precise parameter tuning.

Running Time Analysis

In this subsection, we investigate the running efficiency of the CAC method. For fairness, we only compare the running time of CAC with the baselines based on shallow learning models since deep MVC methods usually take too much time. As shown in Figure 6, although the CAC is not the fastest one, its running time is comparable with the most efficient ones. This verifies that the CAC takes much less time to deal with the multi-view data.

Significance Study

We carry out t-test study on WVU and UCLA with several state-of-the-art baselines in terms of ACC, in which the p-value is processed by $-\log(p)$ with a significance level of 0.05. From Figure 7, we observe that the CAC is statistically significant compared with these baselines.

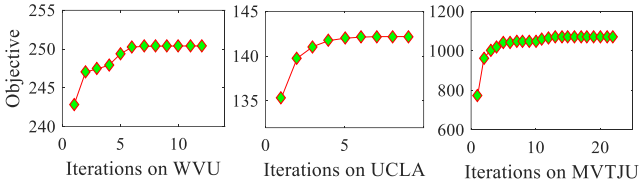


Figure 8: Convergence curves of the CAC on the task of incremental multi-view clustering.

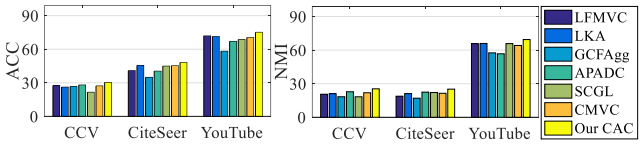


Figure 9: Comparison with representative traditional, deep and incremental MVC baselines on 3 MVC datasets.

Convergence

As shown in Figure 8, with the increase in experimental iterations, the objective function of the CAC monotonically approaches a fixed point on WVU, UCLA and MVTJU datasets. This phenomenon demonstrates that the CAC converges quickly and stably.

Adaptation to other MVC Datasets

To facilitate easier comparison for readers, we apply the CAC to more datasets widely used in the MVC baselines, i.e., CCV (Jiang et al. 2011), CiteSeer[1] and YouTube (Madani, Georg, and Ross 2013). Figure 9 shows the comparison results with several traditional, deep and incremental MVC baselines, which further verifies the effectiveness and robustness of the CAC.

Conclusions

This paper investigates a continual action clustering task with incremental views and proposes a novel CAC method, which achieves never-ending knowledge transfer between historical views and the new coming ones by leveraging late fusion MVC, and improves the clustering performance accordingly. Extensive experimental results demonstrate the superior performance and time/space efficiency of the CAC compared with 15 state-of-the-art baselines.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant No. 61906172, China Postdoctoral Science Foundation under grant No. 2020M682357 and Key Science and Technology Program of Henan Province under grant No. 222102210012.

[1]https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz

References

Bhatia, R.; and Davis, C. 1995. A Cauchy-Schwarz Inequality for Operators with Applications. *LAIA*, 223-224: 119–129.

Bhatnagar, B. L.; Singh, S.; Arora, C.; and Jawahar, C. V. 2017. Unsupervised Learning of Deep Feature Representation for Clustering Egocentric Actions. In *IJCAI*, 1447–1453.

Guo, J.; and Ye, J. 2019. Anchors Bring Ease: An Embarrassingly Simple Approach to Partial Multi-View Clustering. In *AAAI*, 118–125.

Huang, D.; Wang, C.; and Lai, J. 2023. Fast Multi-view Clustering via Ensembles: Towards Scalability, Superiority, and Simplicity. *IEEE TKDE*, Early access: 1–16.

Jiang, Y.; Ye, G.; Chang, S.; Ellis, D. P. W.; and Loui, A. C. 2011. Consumer Video Understanding: A Benchmark Database and An Evaluation of Human and Machine Performance. In *ICMR*, 1–8.

Jones, S.; and Shao, L. 2014. Unsupervised Spectral Dual Assignment Clustering of Human Actions in Context. In *IEEE CVPR*, 604–611.

Kumar, S.; Haresh, S.; Ahmed, A.; Konin, A.; Zia, M. Z.; and Tran, Q. 2022. Unsupervised Action Segmentation by Joint Representation Learning and Online Clustering. In *CVPR*, 20142–20153.

Liu, A.; Xu, N.; Nie, W.; Su, Y.; Wong, Y.; and Kankanhalli, M. S. 2017. Benchmarking a Multimodal and Multiview and Interactive Dataset for Human Action Recognition. *IEEE TCYB*, 47(7): 1781–1794.

Liu, A.; Xu, N.; Su, Y.; Lin, H.; Hao, T.; and Yang, Z. 2015. Single/multi-view Human Action Recognition via Regularized Multi-task Learning. *Neurocomputing*, 151: 544–553.

Liu, X. 2023. SimpleMKKM: Simple Multiple Kernel K-Means. *IEEE TPAMI*, 45(4): 5174–5186.

Liu, X.; Dou, Y.; Yin, J.; Wang, L.; and Zhu, E. 2016. Multiple Kernel k-Means Clustering with Matrix-Induced Regularization. In *AAAI*, 1888–1894.

Liu, X.; Liu, L.; Liao, Q.; Wang, S.; Zhang, Y.; Tu, W.; Tang, C.; Liu, J.; and Zhu, E. 2021. One Pass Late Fusion Multi-view Clustering. In *ICML*, 6850–6859.

Madani, O.; Georg, M.; and Ross, D. A. 2013. On Using Nearly-independent Feature Families for High Precision and Confidence. *Machine Learning*, 92(2-3): 457–477.

Mao, Y.; Yan, X.; Guo, Q.; and Ye, Y. 2021. Deep Mutual Information Maximin for Cross-Modal Clustering. In *AAAI*, 8893–8901.

Peng, B.; Lei, J.; Fu, H.; Shao, L.; and Huang, Q. 2020. A Recursive Constrained Framework for Unsupervised Video Action Clustering. *IEEE TII*, 16(1): 555–565.

Ramagiri, S.; Kavi, R.; and Kulathumani, V. 2011. Real-time Multi-view Human Action Recognition using a Wireless Camera Network. In *ICDSC*, 1–6.

Sun, G.; Cong, Y.; Dong, J.; Liu, Y.; Ding, Z.; and Yu, H. 2022. What and How: Generalized Lifelong Spectral Clustering via Dual Memory. *IEEE TPAMI*, 44(7): 3895–3908.

Tang, C.; Sun, K.; Tang, C.; Zheng, X.; Liu, X.; Huang, J.-J.; and Zhang, W. 2023. Multi-view Subspace Clustering via Adaptive Graph Learning and Late Fusion Alignment. *Neural Networks*, 165: 333–343.

Wan, X.; Liu, J.; Liang, W.; Liu, X.; Wen, Y.; and Zhu, E. 2022. Continual Multi-view Clustering. In *ACM MM*, 3676–3684.

Wang, J.; Nie, X.; Xia, Y.; Wu, Y.; and Zhu, S. 2014. Cross-View Action Modeling, Learning, and Recognition. In *CVPR*, 2649–2656.

Wang, S.; Liu, X.; Zhu, E.; Tang, C.; Liu, J.; Hu, J.; Xia, J.; and Yin, J. 2019. Multi-view Clustering via Late Fusion Alignment Maximization. In *IJCAI*, 3778–3784.

Weinland, D.; Boyer, E.; and Ronfard, R. 2007. Action Recognition from Arbitrary Views using 3D Exemplars. In *ICCV*, 1–7.

Xia, W.; Wang, Q.; Gao, Q.; Zhang, X.; and Gao, X. 2022. Self-Supervised Graph Convolutional Network for Multi-View Clustering. *IEEE TMM*, 24: 3182–3192.

Xu, J.; Li, C.; Peng, L.; Ren, Y.; Shi, X.; Shen, H. T.; and Zhu, X. 2023. Adaptive Feature Projection With Distribution Alignment for Deep Incomplete Multi-View Clustering. *IEEE TIP*, 32: 1354–1366.

Xu, J.; Mei, T.; Yao, T.; and Rui, Y. 2016. MSR-VTT: A Large Video Description Dataset for Bridging Video and Language. In *CVPR*, 5288–5296.

Yan, W.; Zhang, Y.; Lv, C.; Tang, C.; Yue, G.; Liao, L.; and Lin, W. 2023a. GCFAgg: Global and Cross-View Feature Aggregation for Multi-View Clustering. In *CVPR*, 19863–19872.

Yan, X.; Hu, S.; and Ye, Y. 2017. Multi-task clustering of human actions by sharing information. In *CVPR*, 6401–6409.

Yan, X.; Mao, Y.; Li, M.; Ye, Y.; and Yu, H. 2023b. Multitask Image Clustering via Deep Information Bottleneck. *IEEE TCYB*, Early access: 1–14.

Yan, X.; Mao, Y.; Ye, Y.; and Yu, H. 2023c. Cross-Modal Clustering With Deep Correlated Information Bottleneck Method. *IEEE TNNLS*, Early access: 1–15.

Yin, H.; Hu, W.; Zhang, Z.; Lou, J.; and Miao, M. 2021. Incremental Multi-view Spectral Clustering with Sparse and Connected Graph Learning. *NN*, 144: 260–270.

Zhang, T.; Liu, X.; Gong, L.; Wang, S.; Niu, X.; and Shen, L. 2023. Late Fusion Multiple Kernel Clustering With Local Kernel Alignment Maximization. *IEEE TMM*, 25: 993–1007.

Zhang, Z.; Liu, L.; Shen, F.; Shen, H. T.; and Shao, L. 2019. Binary Multi-View Clustering. *IEEE TPAMI*, 41(7): 1774–1782.

Zhao, M.; Yang, W.; and Nie, F. 2023. Auto-weighted Orthogonal and Nonnegative Graph Reconstruction for Multi-view Clustering. *Information Sciences*, 632: 324–339.

Zhou, P.; Shen, Y.; Du, L.; Ye, F.; and Li, X. 2019. Incremental Multi-view Spectral Clustering. *KBS*, 174: 73–86.