AdaptCMVC: Robust Adaption to Incremental Views in Continual Multi-view Clustering

Supplementary Material

6. Introduction

In this supplementary, we provide additional implementation details of our experiments, such as the hyperparameter configurations across all datasets. We also elaborate on the evaluation metrics used in our paper. Furthermore, we provide additional analysis of the noise robustness of AdaptCMVC and updating of the similarity matrix, new comparison, parameter sensitivity analysis, additional examples of the influence of the view order, as well as a discussion on the computational complexity. Finally, we comment on the possible limitations of our work.

7. Experiment

7.1. Implementation Details

Network architectures. We utilize convolutional neural networks to construct the encoder and decoder module following [13]. The encoder module contains four stacked encoder blocks which are comprised of two convolutional layers, two batch normalization layers, and a dropout module. The decoder maintains a symmetric structure with the encoder.

Other hyperparameters. In this work, we have five hyperparameters and Table 6 lists all hyperparameters used for our proposed AdaptCMVC. We also investigate the parameter sensitivity in the proposed method. Figure 6 represents the change of accuracies with different loss weights, which indicates that our method is relatively insensitive to the parameters λ_1, λ_2, and λ_3.

Datasets	λ_1	λ_2	λ_3	β	α
E-MNIST	0.6	5	0.3	0.7	0.999
E-FMNIST	0.01	0.7	0.5	0.001	0.999
Office-31	0.01	1	0.3	0.01	0.999
COIL-100	0.1	0.5	0.5	0.2	0.999
COIL-20	0.9	4	0.3	0.5	0.999
PatchedMNIST	0.5	0.4	0.5	0.2	0.999

Table 6. Hyperparameters used to train the AdaptCMVC. λ_1, λ_2 and, λ_3 are the trade-off hyperparameters of $\mathcal{L}_c$,$\mathcal{L}_s$, and, $\mathcal{L}_r$, respectively. β and α are mentioned in Eq. 7 and Eq. 10.

7.2. Evaluation Metrics

We employ the Accuracy (ACC) and the Normalized Mutual Information (NMI) to evaluate our model and baselines, which are defined as below:

$$\mathbf{ACC} = \max_{m \in \mathcal{M}} \frac{\sum_{j=1}^{n} \delta(m(\hat{y}_j) - y_j)}{n} \tag{11}$$

$$\delta(x, y) = \begin{cases} 0 & x \neq y \\ 1 & x = y \end{cases} \tag{12}$$

where $\hat{y}_j$ is the predicted clustering assignment of sample j, y_j is the ground truth label of sample j. The maximum runs over $\mathcal{M}$, which is the set of all bijective mappings from $1, ..., k$ to itself.

$$\mathbf{NMI} = \frac{2MI(\hat{\mathbf{y}}, \mathbf{y})}{H(\hat{\mathbf{y}}) + H(\mathbf{y})} \tag{13}$$

where $MI(\cdot, \cdot)$ and $H(\cdot)$ represent the mutual information and entropy. $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_n]$,$\mathbf{y} = [y_1, \ldots, y_n]$.

7.3. Noise Robust Analysis

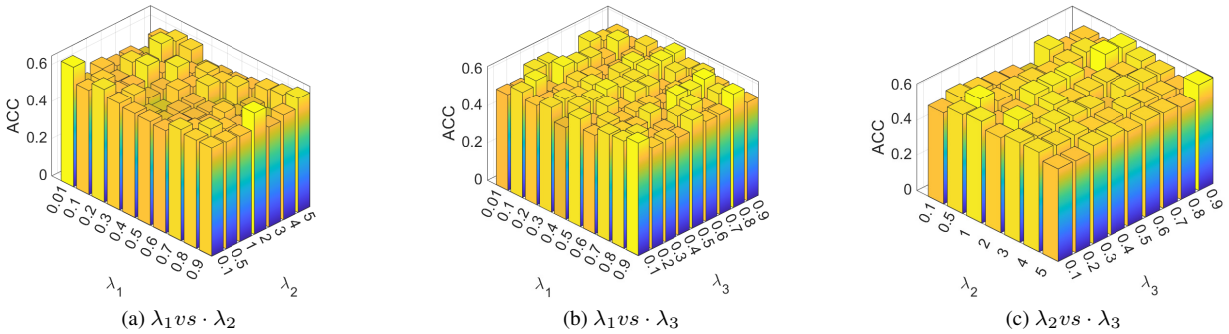
In this section, we compare our model with other CMVC baselines using noisy data to evaluate the effectiveness of the noise robust consistency loss proposed in Eq.6. Specifically, all methods start with the first normal view and we then add Gaussian noise to the subsequent session view, where the Gaussian noise is applied in three different severities. As shown in Table 7, our proposed AdaptCMVC outperforms CAC and CMVC with different Gaussian noise severities. It indicates that our model is more robust to view-specific noise than other CMVC methods.

PatchedMNIST	**CMVC**		**CAC**		**AdaptCMVC**	
	ACC	NMI	ACC	NMI	ACC	NMI
Severity=1	0.6144	0.2667	0.6112	0.2608	**0.7409**	**0.3701**
Severity=2	0.5867	0.1849	0.5991	0.1975	**0.6515**	**0.2588**
Severity=3	0.5777	0.1622	0.5791	0.1628	**0.5904**	**0.2623**
Office-31	**CMVC**		**CAC**		**AdaptCMVC**	
	ACC	NMI	ACC	NMI	ACC	NMI
Severity=1	0.1506	0.1732	0.1525	0.1732	**0.2210**	**0.3288**
Severity=2	0.1534	0.1785	0.1516	0.1772	**0.2068**	**0.3084**
Severity=3	0.1475	0.1691	0.1451	0.1656	**0.2170**	**0.2976**

Table 7. Noise robust analysis on the PatchedMNIST and Office-31 datasets.

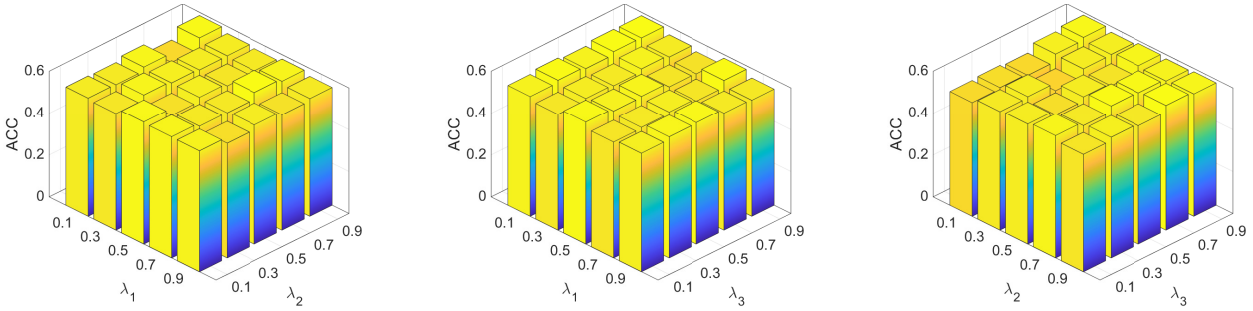
7.4. Analysis of Similarity Matrix Update

To improve the noise robustness of AdaptCMVC, the similarity matrix $\mathbf{S}^{v-1}$ is updated iteratively to better capture the joint group structure. Notably, when all prior similarity



(a) $\lambda_1 vs \cdot \lambda_2$ (b) $\lambda_1 vs \cdot \lambda_3$ (c) $\lambda_2 vs \cdot \lambda_3$

Figure 5. The influence of λ_1, λ_2, and λ_3 on COIL-20 dataset.



(a) $\lambda_1 vs \cdot \lambda_2$ (b) $\lambda_1 vs \cdot \lambda_3$ (c) $\lambda_2 vs \cdot \lambda_3$

Figure 6. The influence of λ_1, λ_2, and λ_3 on COIL-100 dataset.

matrices are incorporated simultaneously, the average ACC across all datasets decreases from 0.5830 to 0.4863. This suggests that incorporating all prior similarity matrices at once may introduce biases toward the initial views, thereby negatively impacting overall performance.

7.5. Additional Comparison to L2SC [58]

As demonstrated in the main paper, our proposed method surpasses state-of-the-art CMVC approaches, including [42, 58]. Here, we provide additional results comparing with L2SC [58], which notably is not scalable to larger datasets. We therefore only evaluate the performance on the three small benchmark datasets: COIL-20 (Ours: **0.6432**, L2SC: 0.5364), Office-31 (Ours: **0.2281** , L2SC: 0.1718), and COIL-100 (Ours: **0.5712**, L2SC: 0.4547). The results consistently demonstrate the effectiveness of AdaptCMVC also compared to L2SC.

7.6. Impact of View Order

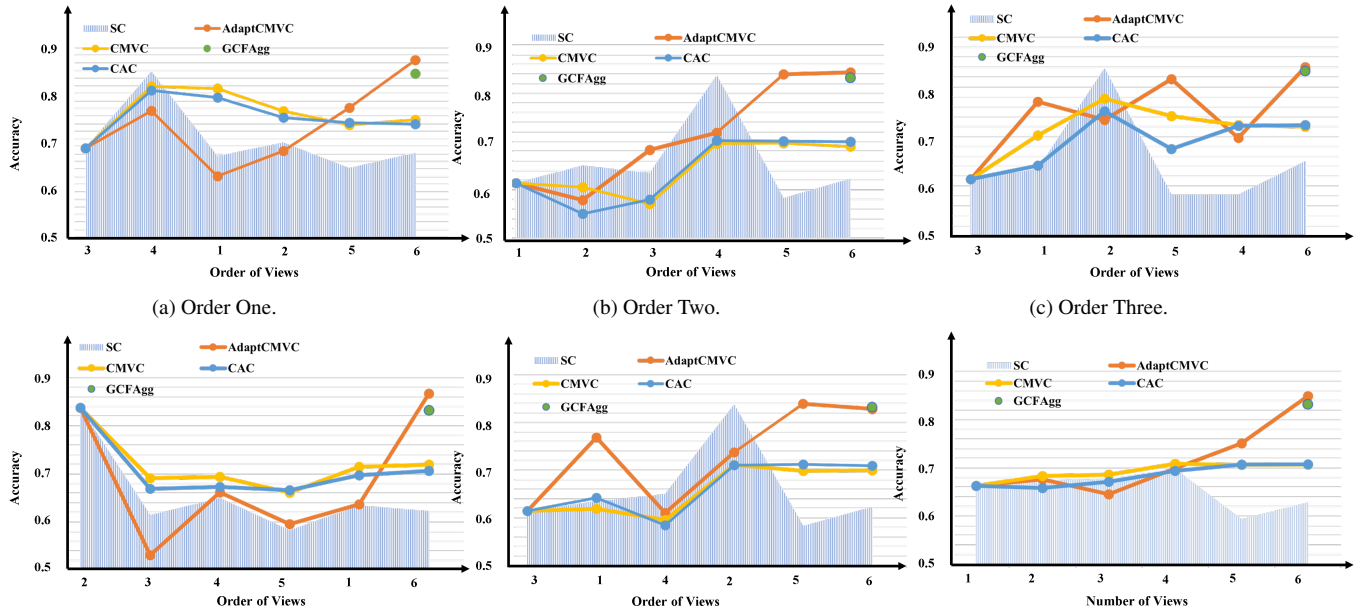
In this section, we add the result of the PatchedMNIST dataset to analyze the impact of the view order mentioned in 4.4. The PatchedMNIST has six views of data, which can provide 6! number of orders. We randomly select 5 kinds of orders. From Figure 7a, 7b, 7c, 7d, 7e, we can see that the order of views has a slight impact on AdaptCMVC, CAC, and CMVC. However, our AdaptCMVC typically maintains a consistent upward trajectory in the continual clustering process with incremental views and achieves good performance. The average plot provided in Figure 7f illustrates the average over these five different orders, where we clearly see that AdaptCMVC is able to leverage the views more efficiently as the number of views increases.

7.7. Discussion of Complexity

In this section, we present an analysis of the computational complexity of our proposed AdaptCMVC framework in comparison with conventional deep MVC approaches. The key advantage of our CMVC methodology lies in its sequential encoder training mechanism, which significantly reduces memory complexity to a constant space requirement of $O(1)$, as opposed to the linear space complexity of $O(m)$ typically required by parallel deep MVC methods that process all views simultaneously. Note, here m denotes the number of views. Although the sequential training paradigm generally results in longer training times compared to joint training approaches, it does exhibit linear scalability to the number of modalities.

7.8. Potential Limitations

A limitation of our domain adaptation inspired approach is that the expected input at the different sessions should have the same input dimension d. While this is not necessarily a problem for general multi-view computer vision tasks,



(a) Order One. (b) Order Two. (c) Order Three.

(d) Order Four. (e) Order Five. (f) Average over Orders.

Figure 7. The impact of view order on our proposed AdaptCMVC. SC represents single-view clustering. CMVC and CAC are other continual multi-view clustering baselines. GCFAgg means simultaneously using all six views to clustering by GCFAgg.

where this can for instance be addressed by rescaling etc., it can restrict its use in multi-modal settings where different modalities (e.g. image and text) should be combined. One exciting direction for future work is to address these settings by decomposing the view-encoder into a shared and modality-specific component.