Mixup-Inspired Video Class-Incremental Learning

1st Jinqiang Long
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
longjinqiang@ruc.edu.cn

2nd Yizhao Gao
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
gaoyizhao@ruc.edu.cn

3rd Zhiwu Lu
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
luzhiwu@ruc.edu.cn

***Abstract*—Continual learning aims to learn a sequence of tasks without forgetting the previously learned knowledge. Although existing memory-based approaches can be easily deployed for video Class-Incremental Learning (CIL), little efforts have been made to explore how to better exploit the data from the previous work (in the memory) for alleviating the catastrophic forgetting. In this work, we thus propose a simple yet effective framework called Mixup-Inspired Video Class-Incremental Learning (MIVCIL). The core idea of our MIVCIL framework is to impose mixup on the current video data and the previous video data (from the memory buffer) to mitigate the catastrophic forgetting. By exploring different mixup strategies on the video data, our MIVCIL framework has three instantiations for video class-incremental learning. We further provide a detailed analysis of the performance and computational overhead of the three instantiations on the latest benchmark vCLIMB. Experimental results show that all three instantiations achieve significant improvements over the representative/state-of-the-art methods.**

***Index Terms*—Continual learning, Video class-incremental learning, Catastrophic forgetting, Mixup**

I. Introduction

In recent years, deep learning has achieved a series of state-of-the-art performance in computer vision with non-streaming data [1], [2]. However, in real-world scenarios, a deep learning model needs to evolve with streaming data, which means that the data for different tasks often arrive in a certain order. Owing to the fact that storing all of the old data and training from scratch with both of the old data and new data are expensive and time-consuming, it is a natural idea to fine-tune the model trained on the old data with the newly-arrived data. Under this setting, when the model is fine-tuned with newly-arrived data to adapt to new tasks, the performance of the model on the previously learned tasks will decline sharply. This phenomenon is well known as *catastrophic forgetting* in continual learning [3]–[6]. Continual learning enables the model trained on a sequence of tasks to mitigate the catastrophic forgetting so that the model can not only perform well on the new tasks but also can maintain good performance on the previously learned tasks. In this work, we focus on a specific case of continual learning: class-incremental learning (**CIL**), where the task-ids is invisible to the model and the model needs to recognize the correct task-id and the correct class-id for the data inputted into the model.

Existing methods for mitigating the catastrophic forgetting in CIL can be divided into three groups: (1) Regularization-based methods add a regularization term to the loss function to penalize the catastrophic forgetting [3], [4]; (2) Memory-based methods store some data samples [5], [6]or generate pseudo data of the previously learned tasks [7], [8] for data replaying; (3) Architecture-based methods assign different subsets of model parameters to different tasks or extend network for newly arrived tasks [9], [10]. More details are given in **Section II**. Most of existing methods are designed for continual learning with image data. However, continual learning is still under-explored in the video domain. Villa et al. [11] extended some methods from image domain to video domain and evaluate them on a standard benchmark. Note that the latest works on video CIL make main attempts to overcome the catastrophic forgetting by designing new model structures or new training strategies. Although memory-based methods can be easily deployed for video CIL, little efforts have been made to explore *how to better exploit* the data from the previously learned tasks (in the memory) for alleviating the catastrophic forgetting.

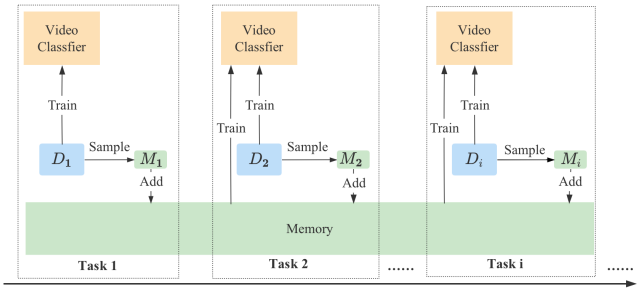

Fig. 1. **Schematic illustration of video class-incremental learning through memory-based methods**. A memory buffer with a fixed maximum size is maintained to store video examples sampled from the data of the previous tasks. When the model is trained on the current task, the data in the memory buffer and the data of the current task will be mixed together and then fed into the model for training.

When the model learns a new task in video CIL, existing memory-based methods simply mix the video data in memory with the video data for the current task to train the model, as shown in **Fig. 1**. We believe that this simple mixing does not effectively utilize the video data in memory that contains information from the previous tasks. To make more efficient use of the video data in memory and further mitigate the catastrophic forgetting, we attempt to perform interpolation on the new video data (for the current task) and the old video

data (from the memory buffer) instead of the simple mixing operation. With such data interpolation, we thus propose a simple yet effective framework called Mixup-Inspired Video Class-Incremental Learning (MIVCIL) by imposing mixup [12] on the current video data and the old video data (from the memory buffer). When different mixup strategies are applied over the video data, our MIVCIL framework has three instantiations for video CIL. Note that our MIVCIL framework is simple to implement and is flexible to be fused with any memory-based methods. For fair comparison, we evaluate the three instantiations of our MIVCIL on the latest video CIL benchmark vCLIMB [11], and find that our proposed MIVCIL methods achieve significant improvements over the state-of-the-art methods under the video CIL setting.

The main contributions of this work can be summarized as follow: **(1)** Following memory-based methods, we propose a simple yet effective framework called MIVCIL to mitigate the catastrophic forgetting for video CIL, which applies mixup [12] to the training data through three different strategies. **(2)** By evaluating the obtained three instantiations of MIVCIL on the latest benchmark vCLIMB, we demonstrate the effectiveness of our MIVCIL in video CIL. **(3)** We provide a detailed analysis of the performance and computation overhead of the three instantiations of our MIVCIL.

II. Related work

Continual Learning. Continual Learning aims to learn novel concepts continually on a sequence of tasks without forgetting the previously learned knowledge. Existing continual learning approaches can be divided into three categories: **(1) Regularization-based methods** [3], [4] attempt to penalize the catastrophic forgetting by adding a regularization term to the loss function to constrain the updating of each parameter of the model. Different methods make difference in how to estimate the importance of the parameters. Memory Aware Synapses (MAS) [3] measures the importance of parameters according to the influence of parameter's change on the output of model, while Elastic Weight Consolidation (EWC) [4] uses the second derivative of loss function with respect to the parameters. **(2) Memory-based methods** [5]–[7] maintain a memory buffer to store a few of examples sampled from the data of the previous tasks or generate pseudo data for data replay (see **Fig. 1**). Existing methods try to find better strategies to manage the memory buffer. Incremental Classifier and Representation Learning (iCaRL) [5] selects samples that are closest to the feature mean of each class and uses a nearest-mean-of-exemplars classifier. Bias Correction (BiC) [6] proposes that there will be an imbalance between old and new classes on large-scale data and adds a bias correction layer based on the iCaRL model to mitigate the class imbalance in large-scale data. While iCaRL [5] and BiC [6] sample from real data, DGR [7] uses a generative model to generate pseudo data for data replay. **(3) Architecture-based methods** [9], [10] assign different subsets of model parameters to different tasks or extend networks for new tasks. In Progressive Neural Networks (PNN) [9], the network trained on each task is frozen and its parameters will not be further updated. When a new task arrives, a new network will be instantiated.

Video CIL. Although continual learning has been thriving in the field of image processing, video CIL is a new continual learning setting that needs to be further explored [13], [14]. Note that a new video CIL benchmark vCLIMB has been proposed in [11]. Since video examples contain a temporal dimension whose size could be greatly various from each other unlike image instances, vCLIMB redefines the memory size in terms of the stored frames for memory-based methods to favor fair comparison among them. Moreover, vCLIMB extends four widely-used continual learning methods (i.e., iCaRL [5], BiC [6], MAS [3], and EWC [4]) to the video CIL setting, and also propose a stronger baseline which adds a temporal consistency loss item to the loss function to reduce the gap between the representation of the origin video clip and the temporally down-sampled version. In this paper, we also make performance evaluation on the benchmark vCLIMB.

Mixup. Mixup [12] is an effective strategy for data augmentation in computer vision research. For two examples (x_1, y_1) and (x_2, y_2) randomly selected from the dataset, the following operation is performed for mixup:

$$\tilde{x} = \lambda x_1 + (1 - \lambda)x_2 \tag{1}$$

$$\tilde{y} = \lambda y_1 + (1 - \lambda)y_2 \tag{2}$$

where $\lambda \sim Beta(\alpha, \alpha)$, for any $\alpha \in (0, \infty)$. Through the above simple linear transformation of the input examples, the generalization ability of a model can be increased, and the robustness of the model against adversarial attack can be improved simultaneously. Even though mixup has already been widely used in traditional image tasks, there is still no good strategy to apply mixup to the video CIL setting at present, which is the focus of our work.

III. Preliminaries

A. Problem Definition

Considering a continual learning problem in video domain, we train a neural network on a continuum of data consisting of T tasks. Each task τ ($\tau = 1, 2, ..., T$) contains its corresponding dataset $D_\tau = \{(x_1, y_1), (x_2, y_2), ..., (x_{n_\tau}, y_{n_\tau})\}$ with n_τ examples, where x_i is the input video, and y_i is the corresponding label, for each example $(x_i, y_i) \in D_\tau$. And each video $x = \{f_1, f_2, ..., f_{n_x}\}$ consists of n_x frames. Considering a neural network f_Θ parameterized by Θ and a classifier g_Φ parameterized by Φ, we can define the cross-entropy (CE) loss on D_τ as follows:

$$\mathcal{L} = \sum_{(x_i, y_i) \in D_\tau} CE(g_\Phi(f_\Theta(x_i)), y_i) \tag{3}$$

Our work focus on the memory-based methods for class-incremental learning (**CIL**), which will select and store examples from dataset D_τ into a memory buffer M_τ with a fixed maximum size S_m. When the model is train on task τ, the memory can be defined as $M_\tau = \{(x_1, y_1), (x_2, y_2), ..., (x_{m_\tau}, y_{m_\tau})\}$ with m_τ examples, where m_τ should be limited by the constraint $m_\tau \leq S_m$.

B. Evaluation Metrics

We use two standard metrics proposed to evaluate the model performance: Final Average Accuracy (**Acc**) and Backward Forgetting (**BWF**).

Final Average Accuracy is the average test accuracy of all the tasks after the last task τ_T is completed. Let $a_{\tau',\tau}$ denotes the accuracy of task τ after the task τ' is completed. The Acc can be defined as follows:

$$\text{Acc} = \frac{1}{T}\sum_{\tau=1}^{T} a_{T,\tau} \tag{4}$$

Backward Forgetting can evaluate average accuracy decrease between the maximum accuracy and the minimum accuracy of each task after the last task is completed. BWF can be defined as follows:

$$\text{BWF} = \frac{1}{T-1}\sum_{\tau=1}^{T-1} \max_{\tau' \in \{1,\dots,T\}} (a_{\tau',\tau} - a_{T,\tau}) \tag{5}$$

IV. Methodology

The memory-based methods maintain a memory buffer that stores some samples from the previous tasks for data replay to mitigate the catastrophic forgetting. Since the memory buffer is empty when the model is trained on the first task ($\tau = 1$), we focus on how to impose mixup when the model is trained on task τ ($\tau \in \{2, ..., T\}$).

Specifically, when the model is being trained on task τ, the training data consists of two parts: the data of the current task $D_\tau = \{(x_{1,\mathcal{D}}, y_{1,\mathcal{D}}), (x_{2,\mathcal{D}}, y_{2,\mathcal{D}}), ..., (x_{n_\tau,\mathcal{D}}, y_{n_\tau,\mathcal{D}})\}$, and the examples stored in the memory buffer $M_\tau = \{(x_{1,\mathcal{M}}, y_{1,\mathcal{M}}), (x_{2,\mathcal{M}}, y_{2,\mathcal{M}}), ..., (x_{m_\tau,\mathcal{M}}, y_{m_\tau,\mathcal{M}})\}$. The traditional memory-based methods [5], [6] choose to simply mix D_τ and M_τ together, and then train the model with the mixed data, which can effectively mitigate the catastrophic forgetting in continual learning. In this work, inspired by such memory-based methods, we propose a stronger framework **MIVCIL** by imposing mixup [12] on the data in M_τ and D_τ by three different strategies instead of the traditional simply mixing.

A. Mixup on Data

The first strategy distinguishes between the data in D_τ and M_τ, and utilizes **M**ixup **o**n the **D**ata of the current task (**MoD**). We do not make any modification to the data in the memory buffer, while augmenting the data of the current task by implementing mixup during training.

For each example of the current task $(x_{i,\mathcal{D}}, y_{i,\mathcal{D}}) \in D_\tau$, we select an example $(x_{j,\mathcal{M}}, y_{j,\mathcal{M}}) \in M_\tau$ randomly from the memory buffer and interpolate between them as:

$$\tilde{x}_{i,\mathcal{D}} = \lambda x_{i,\mathcal{D}} + (1-\lambda) x_{j,\mathcal{M}} \tag{6}$$

where $\lambda \sim Beta\ (\alpha, \alpha)$, for any $\alpha \in (0, \infty)$. Whereupon, we train the model on the data in the memory buffer and the interpolated data of the current task (see **Fig. 2**). The loss function for the interpolated data can be defined as:

$$\begin{aligned}\mathcal{L}_{mixup} = &\sum_{i,j} \lambda CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{D}})), y_{i,\mathcal{D}}) \\ &+ (1-\lambda) CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{D}})), y_{j,\mathcal{M}})\end{aligned} \tag{7}$$

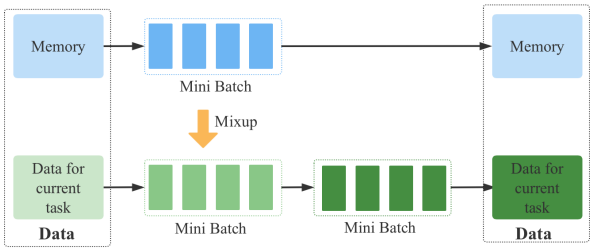

Fig. 2. **Illustration of MoD.** Given a data batch from the current task, **MoD** randomly samples a data batch from the memory buffer, and then implements mixup between the two batches to obtain the interpolated data. The model is trained with the interpolated data and the data in memory buffer.

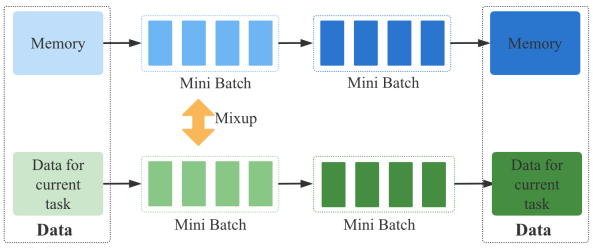

Fig. 3. **Illustration of MoDaM.** Based on the setting of **MoD**, **MoDaM** additionally samples a data batch from the current task and implements mixup between this batch and the data batch from the memory buffer.

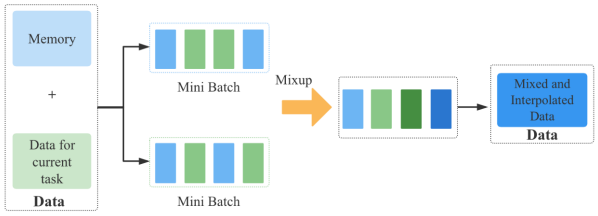

Fig. 4. **Illustration of MaM. MaM** directly mixes the data from the current task and the data from the memory buffer together (without any modification), and then implements mixup between the mixed data.

B. Mixup on Data and Memory

The second strategy also distinguishes between the data in D_τ and M_τ, but implements **M**ixup **o**n the **D**ata of the current task **a**nd the data in **M**emory buffer (**MoDaM**), which is different from the first strategy.

For each example $(x_{i,\mathcal{M}}, y_{i,\mathcal{M}}) \in M_\tau$, we select an example $(x_{j,\mathcal{D}}, y_{j,\mathcal{D}}) \in D_\tau$ from the data of the current task to implement mixup on them. Simultaneously, we make the same modification to the data of the current task just as MoD. MoDaM (see **Fig. 3**) can be defined as:

$$\tilde{x}_{i,\mathcal{M}} = \lambda_1 x_{i,\mathcal{M}} + (1-\lambda_1) x_{j,\mathcal{D}} \tag{8}$$

$$\tilde{x}_{i,\mathcal{D}} = \lambda_2 x_{i,\mathcal{D}} + (1-\lambda_2) x_{j,\mathcal{M}} \tag{9}$$

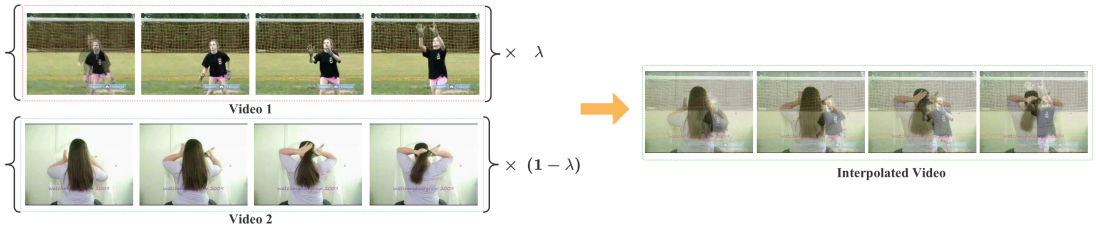

Fig. 5. **Illustration of mixup between two videos.** In practice, each video will be represented by the same number of frames (we show 4 frames per video here as an example). We interpolate the corresponding frames of the two videos with the scale coefficient λ.

The loss function for the interpolated data is defined as:

$$\mathcal{L}_{mixup,\mathcal{M}} = \sum_{i,j} \lambda_1 CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{M}})), y_{i,\mathcal{M}}) + (1-\lambda_1) CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{M}})), y_{j,\mathcal{D}}) \quad (10)$$

$$\mathcal{L}_{mixup,\mathcal{D}} = \sum_{i,j} \lambda_2 CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{D}})), y_{i,\mathcal{D}}) + (1-\lambda_2) CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{D}})), y_{j,\mathcal{M}}) \quad (11)$$

C. Mix and Mixup

Different from the above two mixup strategies, the third strategy does not distinguish between the data in D_τ and M_τ, which chooses to **M**ix them **a**nd implement **M**ixup (**MaM**). Specifically, we simply mix the two parts of data just the same as the traditional memory-based methods, where the $D_{union,\tau} = D_\tau \cup M_\tau$. For each example $(x_{i,\mathcal{D}_u}, y_{i,\mathcal{D}_u}) \in D_{union,\tau}$, MaM randomly selects another example $(x_{j,\mathcal{D}_u}, y_{j,\mathcal{D}_u}) \in D_{union,\tau}$ from the mixed data and interpolate between them as follows:

$$\tilde{x}_{i,\mathcal{D}_u} = \lambda x_{i,\mathcal{D}_u} + (1-\lambda) x_{j,\mathcal{D}_u} \quad (12)$$

The model is trained on the mixed and interpolated data (see **Fig. 4**). The loss function for the interpolated data is:

$$\mathcal{L}_{mixup} = \sum_{i,j} \lambda CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{D}_u})), y_{i,\mathcal{D}_u}) + (1-\lambda) CE(g_\Phi(f_\Theta(\tilde{x}_{i,\mathcal{D}_u})), y_{j,\mathcal{D}_u}) \quad (13)$$

For easy understanding, the details of the mixup operation over two videos in Equation (12) are given as follows. For $\text{video}_i = \{F_1, F_2, ..., F_{n_i}\}$ and $\text{video}_j = \{F_1, F_2, ..., F_{n_j}\}$, we select the same number of frames to form the representation of the videos, i.e., we have $\text{video}_i = \{F_{1,i}, F_{2,i}, ..., F_{m,i}\}$ and $\text{video}_j = \{F_{1,j}, F_{2,j}, ..., F_{m,j}\}$. The mixup operation over two videos is then defined as:

$$F_k = \lambda F_{k,i} + (1-\lambda) F_{k,j}, k \in \{1, 2, ..., m\} \quad (14)$$

which is also illustrated in **Fig. 5**.

Overall, the aforementioned three strategies utilize mixup on the video data in the memory and the video data of the current task, which can mitigate catastrophic forgetting and increase the robustness of the model. With these mixup strategies, we can obtain three instantiations of our MIVCIL. Since our MIVCIL is model-independent, it can be applied to any memory-based models for video CIL. In addition, our MIVCIL does not require the tuning of additional hyper-parameter so that it is very easy to implement.

V. Experiments

A. Experimental Setup

Baselines. In this work, we evaluate our MIVCIL on the new video CIL benchmark vCLIMB [11]. We apply our MIVCIL to a stronger baseline proposed in vCLIMB, which adds the temporal consistency loss to that of iCaRL [5] (iCaRL+TC [11]). We make comparison to two regularization-based methods [3], [4] and two memory-based methods [5], [6], as in vCLIMB. Additionally, we also provide a detailed analysis of the performance and computation overhead of the three instantiations of our MIVCIL.

Datasets and Tasks. To explore the effectiveness and scalability of our MIVCIL, we make evaluation on UCF101 [15] and HMDB51 [16], which are two widely-used benchmark datasets for video classification. **(1) UCF101** is an action video dataset for action recognition, which contains 13.3K videos from 101 action classes. Each video in UCF101 has an average of 182 frames. **(2) HMDB51** is a video dataset for action recognition as well, which contains 6.7K videos from 51 action classes. Each video has an average of 95 frames. We split the two datasets into 10 tasks, where each task has 10 classes for UCF101 and 5 classes for HMDB51 (but the first task has one more class).

Implement Details. (1) Model Details. We adopt Temporal Segment Networks (TSN) [17] with ResNet-34 pretrained on ImageNet as the backbone, as in vCLIMB [11]. We follow the same temporal data augmentation proposed in TSN [17], using 8 segments per video and 1 frame per segment. The factor λ of mixup [12] obeys beta distribution ($\lambda \sim Beta\ (\alpha, \alpha)$), where we set $\alpha = 0.2$ in this paper. For the factor of the temporal consistency loss, we use $\lambda_{tc} = 0.5$, following VCILMB [11]. For the size of memory buffer, we use the same number of example per class as vCLIMB [11], which is set to 20 when the model has learned all the tasks. Therefore, the maximum size of memory buffer should be 2,020 and 1,020 for UCF101 [15] and HMDB51 [16], respectively. **(2) Training Details.** We train the model in a supervised fashion for 50 epochs per task. We make use of the optimizer Adam with the initial learning rate of 1×10^{-3} and the learning schedule with $\text{milestone} = [10, 20]$ to optimize the model.

B. Main Results

The comparative results of video CIL on UCF101 [15] and HMDB51 [16] are shown in **Table I**. It can be clearly seen that the performance of regularization-based methods [3], [4]

TABLE I

COMPARATIVE RESULTS OF VIDEO CIL ON UCF101 AND HMDB51. MOD, MODAM, AND MAM ARE THE THREE INSTANTIATIONS OF OUR MIVCIL. *Frames per Video* DENOTES THE NUMBER OF FRAMES STORED IN THE MEMORY BUFFER FOR EACH VIDEO, AND THE SIZE OF *Memory Buffer* IS MEASURED BY THE NUMBER OF STORED FRAMES. WE REPORT THE FINAL AVERAGE ACCURACY (ACC) AND BACKWARD FORGETTING (BWF) FOR PERFORMANCE EVALUATION. THE BEST SCORES ARE HIGHLIGHTED IN BOLD, AND THE SECOND BEST SCORES ARE UNDERLINED.

Model	Frames per Video	UCF101			HMDB51		
		Memory Buffer	Acc ↑	BWF ↓	Memory Buffer	Acc ↑	BWF ↓
EWC [4]	none	none	9.51%	98.95%	None	2.31%	74.74%
MAS [3]	none	none	10.89%	**11.11%**	None	9.12%	29.20%
Bic [6]	all	3.69×10^5	78.16%	18.49%	9.67×10^4	35.71%	**16.66%**
iCaRL [5]	4	8.08×10^3	59.98%	34.17%	4.08×10^3	21.99%	29.41%
	8	16.16×10^3	60.03%	32.24%	8.16×10^3	21.05%	29.19%
	16	32.32×10^3	62.26%	31.91%	16.32×10^3	20.60%	31.53%
iCaRL+TC [11]	4	8.08×10^3	72.19%	25.37%	4.08×10^3	26.11%	32.84%
	8	16.16×10^3	74.70%	22.31%	8.16×10^3	25.65%	28.62%
	16	32.32×10^3	76.10%	19.78%	16.32×10^3	26.70%	28.99%
MoD (ours)	4	8.08×10^3	79.68%	16.93%	4.08×10^3	34.84%	20.18%
	8	16.16×10^3	<u>80.21%</u>	16.64%	8.16×10^3	34.83%	19.90%
	16	32.32×10^3	**80.81%**	14.87%	16.32×10^3	36.27%	17.20%
MoDaM (ours)	4	8.08×10^3	72.56%	14.73%	4.08×10^3	36.28%	18.36%
	8	16.16×10^3	73.76%	13.82%	8.16×10^3	<u>36.49%</u>	<u>17.05%</u>
	16	32.32×10^3	74.66%	<u>13.18%</u>	16.32×10^3	**37.07%**	17.27%
MaM (ours)	4	8.08×10^3	77.27%	17.50%	4.08×10^3	28.66%	29.35%
	8	16.16×10^3	76.27%	17.89%	8.16×10^3	35.62%	23.53%
	16	32.32×10^3	77.59%	16.80%	16.32×10^3	34.51%	25.72%

is extremely poor on either datasets, although they can lead to very low backward forgetting (BWF) sometimes. Therefore, we pay more attention to comparing our MIVCIL with the memory-based methods [5], [6], [11].

Results on UCF101. Table I (columns3-5) shows the comparative results of different methods on UCF101 [15]. We can observe that: (1) Our MoD and MaM achieve much better results than other methods on the final average accuracy (Acc) with the same memory buffer size. Concretely, our MoD outperforms recent methods by large margins on all settings (7.49% on 4 frames, 5.51% on 8 frames, and 4.71% on 16 frames). It has an average increase of 5.90% in accuracy compared to iCaRL+TC [11] and an average increase of 19.47% compared to iCaRL [5]. (2) With only 4 frames for each video in memory buffer, our MoD achieves higher accuracy than BiC [6], which stores *all* frames per video and requires about 45 times of memory space capacity. (3) Although MAS [3] achieves the lowest BWF, it almost fails in terms of Acc. This means that BWF should be only used as an auxiliary metric in addition to Acc.

Results on HMDB51. Table I (columns 6-8) shows the comparative results on HMDB51 [16]. It can be seen that: (1) All of our three methods outperform the other methods (except BiC [6]), where Acc has been improved and BWF has declined to a large extent. (2) Among our three methods, MoDaM performs the best on HMDB51, which leads to an average increase of 10.46% in Acc and an average decrease of 12.59% in BWF compared with iCaRL+TC [11]. (3) Although the MoD shows slightly weaker performance among our proposed strategies, it still outperforms the baselines with a distinct margin. Concretely, it has an average increase of 6.78% in accuracy compared to iCaRL+TC [11] and and 11.72% in accuracy compared to iCaRL [5]. (4) Compared to BiC [6] with a much larger memory buffer (using all frames per video), our MoDaM achieves higher Acc scores with similar BWF scores (using 4 frames per video), which only requires 1/23 of Bic's memory buffer.

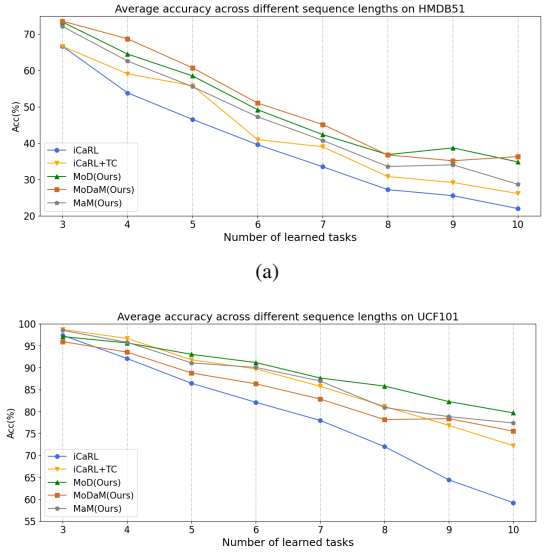

Fig. 6. **Illustration of the average accuracy (Acc) on learned tasks after learning task sequences of different length on HMDB51 (*a*) and UCF101 (*b*).** We set the number of frames per video stored in the memory buffer to 4.

C. Further Evaluation

To further demonstrate the superiority of our methods, we show the average accuracy of the model on all learned tasks after it is trained sequentially on each task of HMDB51 (*a*) and UCF101 (*b*) in **Fig.** 6 . Since the average accuracy results acquired after the first two tasks are of minor difference across

TABLE II
RESULTS OF COMPUTATION OVERHEAD ANALYSIS. WE REPORT THE TRAINING TIME AND THE MAXIMUM GPU OCCUPATION DURING THE TRAINING PROCESS FOR EACH METHOD. WE SET THE NUMBER OF FRAMES PER VIDEO STORED IN THE MEMORY BUFFER TO 4.

Model	UCF101			HMDB51		
	Time	GPU	Acc	Time	GPU	Acc
iCaRL+TC [11]	10h	18G	72.19%	4h	18G	26.11%
MoD (ours)	24h	20G	79.68%	7h	20G	34.84%
MoDaM (ours)	40h	24G	75.51%	14h	24G	36.28%
MaM (ours)	11h	20G	77.27%	5h	20G	28.66%

different approaches, we show the results from the end of the third task for clear comparison. It can be observed that: (1) On HMDB51, all of our three methods keep reaching higher average accuracy on all learned tasks than the other competitors after each task (i.e. with different task sequence lengths). (2) On UCF101, although our proposed MoD and MaM shows less superiority over the baselines when the number of learned tasks is small, they outperform the baselines with distinct larger margins when the number of learned tasks grows larger. The above two observations prove the good generalization ability of our method on different length of task sequence and the superiority in learning on challenging long task sequence. Overall, these two advantages provide further evidence that our MIVCIL indeed achieves great success in mitigating the catastrophic forgetting in video CIL scenario.

D. Computation Overhead Analysis

Table II provides the comparison on the computation overhead of our three methods and iCaRL+TC [11]. Consistent to the no-free-lunch theorem, the performance improvement of our methods comes from the better utilization of the information/knowledge of memory data, which leads to more training time and GPU occupation on both UCF101 and HMDB51. Concretely, our MoD needs more GPU space (2G) than iCaRL+TC but outperforms it by large margins (7.49% on UCF101 and 8.73% on HMDB51). In addition, our MoDaM requires 6G more GPU space with much higher Acc scores (especially 36.28% vs. 26.11% on HMDB51).

Note that the increase of computation overhead is not surprising, since our MIVCIL imposes mixup [12] on almost every example in the dataset. More specifically, MoDaM needs to: (1) sample examples from memory buffer M_τ for each data batch from the current task D_τ; (2) sample examples from D_τ for each data batch from M_τ to implement mixup twice, which leads to the most computation overhead. MoD requires less overhead than MoDaM due to the lack of the second part, while the cost of MaM is the least because the process of sampling for each batch is not required. In practice, we have to balance performance improvement and computing overhead to choose the best instantiation of our MIVCIL.

VI. CONCLUSION

In this paper, we propose a simple yet effective framework called MIVCIL, which has three instantiations to deploy mixup [12] for video CIL based on memory buffer. We evaluate our MIVCIL on a novel benchmark vCLIMB, and show that our MIVCIL can significantly mitigate the catastrophic forgetting in video CIL. Additionally, we provide a detailed analysis of the performance and computation overhead of the three instantiations of our MIVCIL. Although our MIVCIL only utilizes mixup at the raw video level, we believe that the use of mixup at the feature level also has great potential. In our ongoing work, we will explore more approaches.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China (61976220 and 62376274). Zhiwu Lu is the corresponding author.

REFERENCES

[1] M. Bain, A. Nagrani, G. Varol, and A. Zisserman, "Frozen in time: A joint video and image encoder for end-to-end retrieval," in *IEEE International Conference on Computer Vision*, 2021, pp. 1728–1738.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[3] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *European Conference on Computer Vision*, 2018, pp. 139–154.

[4] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[5] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[6] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 374–382.

[7] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[8] Y. Xiang, Y. Fu, P. Ji, and H. Huang, "Incremental learning using conditional adversarial networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6619–6628.

[9] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[10] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[11] A. Villa, K. Alhamoud, V. Escorcia, F. Caba, J. L. Alcázar, and B. Ghanem, "vclimb: A novel video class incremental learning benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 035–19 044.

[12] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[13] J. Park, M. Kang, and B. Han, "Class-incremental learning for action recognition in videos," in *IEEE International Conference on Computer Vision*, 2021, pp. 13 698–13 707.

[14] H. Zhao, X. Qin, S. Su, Y. Fu, Z. Lin, and X. Li, "When video classification meets incremental classes," in *ACM Multimedia*, 2021, pp. 880–889.

[15] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[16] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *IEEE International Conference on Computer Vision*, 2011, pp. 2556–2563.

[17] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2740–2755, 2018.