A Additional Experiments for Different Memory Budgets

This is supplementary to Section 4.2 "Comparison with State-of-the-art Results". Limited by space, we only summarize the results of our approach using the same exemplar memory size per class as others and the experiment results of "Memory budget comparison" on UCF101 with 10 steps in the main body. Table A1 supplements the average accuracy of FrameMaker with different memory budgets under all benchmarks of UCF101 and HMDB51 datasets, which is compared with the best performance of TCD [5].

Table A1: Comparison with the state-of-the-art approaches over class-incremental action recognition performance on UCF101 and HMDB51 with different memory budgets.

<table>
<thead>
<tr>
<th>Num. of Classes</th>
<th>Memory Per Class</th>
<th>UCF101</th>
<th>HMDB51</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 × 5 stages</td>
<td>5 × 10 stages</td>
<td>2 × 25 stages</td>
</tr>
<tr>
<td></td>
<td>CNN NME</td>
<td>CNN NME</td>
<td>CNN NME</td>
</tr>
<tr>
<td>8F × 5V=6.0Mb</td>
<td>74.89 77.16</td>
<td>73.43 75.35</td>
<td>72.19 74.01</td>
</tr>
<tr>
<td>1F × 1V=0.15Mb</td>
<td>58.75 73.97</td>
<td>49.37 70.78</td>
<td>48.31 65.86</td>
</tr>
<tr>
<td>1F × 2V=0.3Mb</td>
<td>67.59 76.18</td>
<td>62.06 74.18</td>
<td>55.84 72.88</td>
</tr>
<tr>
<td>1F × 5V=0.75Mb</td>
<td>73.05 76.70</td>
<td>72.93 76.64</td>
<td>68.79 75.36</td>
</tr>
<tr>
<td>1F × 8V=1.2Mb</td>
<td>76.09 77.88</td>
<td>73.64 76.98</td>
<td>73.57 76.76</td>
</tr>
<tr>
<td>1F × 16V=2.4Mb</td>
<td>77.24 78.06</td>
<td>75.19 77.43</td>
<td>74.92 76.92</td>
</tr>
<tr>
<td>1F × 40V=6.0Mb</td>
<td><strong>78.13 78.64</strong></td>
<td><strong>76.38 78.14</strong></td>
<td><strong>75.77 77.49</strong></td>
</tr>
</tbody>
</table>

As can be seen from Table A1, under other settings, the results of our method are similar to that on UCF101 with 10 steps. When using about 20% of the storage space, our method can achieve better performance than TCD, and the accuracy is further improved when the number of saved exemplars is increased, which also proves the robustness of our approach.

*equal contribution
B Additional Analysis of the Balance Weights

This is supplementary to Section 4.3 "Ablation Study". We further discuss the sensitivity to the balance weight for each term in $L_{fc}$ on UCF101 with 10 steps.

We first discuss the sensitivity of $\alpha$ and $\beta$, the balance weights for $L_{fc}$ and $L_{ce}$ which are used to achieve stronger constraint on $W^k_i$ for its effective training. Table A2 shows the performance of frame condensing (without instance-specific prompt) under various combinations of $\alpha$ and $\beta$. We find the performance under the combination of $\{\alpha = 1, \beta = 1\}$ consistently exceeds others. Although a slightly higher performance is obtained on NME under $\{\alpha = 2, \beta = 1\}$, it is difficult to achieve the same gain on the two different metrics after continuous adjustment.

Table A2: Sensitivity of the performance of Frame Condensing to $\alpha$ and $\beta$ on UCF101 with 10 steps. Default settings are marked in gray.

<table>
<thead>
<tr>
<th>$\alpha (L_{fc})$</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>0.5</th>
<th>0.1</th>
<th>0.01</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta (L_{ce})$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0.5</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td>CNN</td>
<td>72.29</td>
<td>72.27</td>
<td>71.83</td>
<td>72.22</td>
<td>72.07</td>
<td>71.94</td>
<td>71.93</td>
<td>71.44</td>
<td>72.14</td>
<td>71.96</td>
<td>71.92</td>
</tr>
<tr>
<td>NME</td>
<td>76.42</td>
<td>76.43</td>
<td>76.36</td>
<td>76.25</td>
<td>76.21</td>
<td>76.13</td>
<td>76.37</td>
<td>76.23</td>
<td>76.22</td>
<td>75.91</td>
<td>75.84</td>
</tr>
</tbody>
</table>

Then we further analyze sensitivity of $\gamma$ and $\eta$, the balance weights for $L_{pf}$ and $L_{pce}$, when setting $\alpha$ and $\beta$ as the optimal value, 1. Table A3 shows the performance of FrameMaker (frame condensing with instance-specific prompt) under various combinations of $\gamma$ and $\eta$.

Table A3: Sensitivity of the performance of FrameMaker to $\gamma$ and $\eta$ ($\alpha = 1, \beta = 1$) on UCF101 with 10 steps. Default settings are marked in gray.

<table>
<thead>
<tr>
<th>$\gamma (L_{pf})$</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>0.5</th>
<th>0.1</th>
<th>0.01</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta (L_{pce})$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0.5</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td>CNN</td>
<td>72.93</td>
<td>72.94</td>
<td>72.92</td>
<td>72.88</td>
<td>72.64</td>
<td>72.67</td>
<td>72.90</td>
<td>72.85</td>
<td>72.92</td>
<td>72.80</td>
<td>72.73</td>
</tr>
<tr>
<td>NME</td>
<td>76.64</td>
<td>76.62</td>
<td>76.58</td>
<td>76.61</td>
<td>76.59</td>
<td>76.61</td>
<td>76.65</td>
<td>76.58</td>
<td>76.61</td>
<td>76.55</td>
<td>76.51</td>
</tr>
</tbody>
</table>

We find the performance under the combination of $\{\gamma = 1, \eta = 1\}$ always exceeds others. Some groups of weights make one of the test metrics higher but only $\{\gamma = 1, \eta = 1\}$ obtain the best balance of them. Therefore, we set all the balance weights as 1 finally in our experiments.

C Implementation Details

This section provides some additional details about the experiments in the main body.

Training Details. This is supplementary to Section 3.3 "Training". We further supplement the elaborate training process and knowledge distillation loss in this section. Figure A1 shows the overall framework of training process.

In the training process of the incremental step $k$, we use the dataset $D^k = D^k \cup M^{k-1}$ to update the model from $F(\cdot; \Theta_{k-1})$, where $D^k$ is the dataset of the task $k$ which consists of videos belonging to current classes, and $M^{k-1}$ is memory bank which contains condensed frames generated by FrameMaker after the step $k - 1$. The model classifies the samples from the two datasets respectively and use exemplars in old classes for knowledge distillation.

For the classification task, we adopt cross entropy loss to supervise. $L_{ce}^d$ and $L_{ce}^n$ are the cross entropy losses for video example $V^k_i$ in $D^k$ and condensed frame exemplar $I^{k-1}_i$ in $M^{k-1}$.

\[
L_{ce}^d = \text{CrossEntropy}(F(V^k_i; \Theta_k), y^k_i), \tag{A1}
\]

\[
L_{ce}^n = \text{CrossEntropy}(F(I^{k-1}_i; \Theta_k), y^{k-1}_i), \tag{A2}
\]
where $y^k_i$ and $y^{k-1}_i$ is the ground truth label of $V^k_i$ and $I^{k-1}_i$, respectively.

In order to save the information of old classes better, the knowledge distillation function of PODNet [1] is employed on the old classes data $M^{k-1}$ and it consists of two components, the spatial distillation function $L_{\text{spatial}}$ and the final embedding distillation function $L_{\text{flat}}$.

$$
L_{\text{spatial}}(f_i(I_i^{k-1}; \Theta_k), f_l(I_l^{k-1}; \Theta_{k-1})) = \sum_{c=1}^C \sum_{h=1}^H \left[ \sum_{w=1}^W f_{l,c,w,h}(I_l^{k-1}; \Theta_k) - \sum_{u=1}^W f_{l,c,w,h}(I_l^{k-1}; \Theta_{k-1}) \right]^2 \tag{A3}
$$

$$
+ \sum_{c=1}^C \sum_{w=1}^W \left[ \sum_{h=1}^H f_{l,c,w,h}(I_l^{k-1}; \Theta_k) - \sum_{h=1}^H f_{l,c,w,h}(I_l^{k-1}; \Theta_{k-1}) \right]^2,
$$

where $f_i(\cdot; \Theta_k)$ is the output of intermediate convolution layer $l$ of model $F(\cdot; \Theta_k)$. $C$, $H$, $W$ are the size of channel, height and width of the output feature of layer $l$.

$$
L_{\text{flat}}(f(I_i^{k-1}; \Theta_k), f(I_l^{k-1}; \Theta_{k-1})) = \|f(I_i^{k-1}; \Theta_k) - f(I_l^{k-1}; \Theta_{k-1})\|^2, \tag{A4}
$$

where $f(\cdot; \Theta_k)$ is the feature extractor of model $F(\cdot; \Theta_k)$.

The final distillation loss of the condensed frame exemplars is given by:

$$
L_{\text{dist}} = L_{\text{spatial}}(f_i(I_i^{k-1}; \Theta_k), f_l(I_l^{k-1}; \Theta_{k-1})) + L_{\text{flat}}(f(I_i^{k-1}; \Theta_k), f(I_l^{k-1}; \Theta_{k-1})), \tag{A5}
$$

To simplify, the hyperparameters used in PODNet [1] final distillation loss is ignored.

The total objective function for task $k$ is given by:

$$
L_{\text{cil}} = L_{\text{ce}}^d + L_{\text{ce}}^m + L_{\text{dist}}^m, \tag{A6}
$$

which is the same as Eq. 8 in the manuscript.

**Dataset Details.** Following TCD [5], we use three action recognition datasets, UCF101 [8], HMDB51 [3] and Something-Something V2 [2]. The HMDB51 dataset is a large collection of realistic videos from various sources which can be found in https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/. UCF101 dataset is an extension of UCF50 [7] and consists of 13,320 video clips, which can be found in https://wwwcrcv.ucf.edu/data/UCF101.php. Something-SomethingV2 dataset is a large-scale motion-sensitive dataset that shows humans performing pre-defined basic actions with everyday objects, which can be found in https://developer.qualcomm.com/software/ai-datasets/something-something. For UCF101, we have three settings, 51 + 10 × 5 stages, 51 + 5 × 10 stages and 51 + 1 × 25 stages. For HMDB51, the number of initial task classes is 26, and the remaining 25 classes are separated into 5 or 25 groups. For Something-Something V2, the benchmarks are 84 + 10 × 9 stages and 84 + 5 × 18 stages. The random seed we use is also the same as TCD [5].
Figure A2: Channel comparisons between different inputs on HMDB51. Each line indicates the feature channel activation value obtained with the specified frame as input. FC and ISP are abbreviations of our proposed Frame Condensing and Instance-Specific Prompt, respectively.

Figure A3: Visualizations of condensed frames and GradCAM maps on HMDB51. FC and ISP refer to proposed Frame Condensing and Instance-Specific Prompt, respectively.

**Model and Hyperparameter Setup.** TSM [4] is employed as our backbone, and we follow the data preprocessing procedure and the basic training setting of TSM. We train a ResNet-34 TSM for UCF101 and a ResNet-50 TSM for HMDB51 and Something-Something V2. Each incremental training procedure takes 50 epochs. We use SGD optimizer with an initial learning rate of 0.04 for UCF101 and Something-Something V2, and 0.002 for HMDB51, which is reduced by half after 25 epochs. For incremental steps, we set the learning rate for all datasets as 0.001 and make the same reduction. The learning rate is set as 0.01 and 0.001 for condensing weights and the instance-specific prompt, respectively, and the total iteration for their optimization is set to be 8k.

**Hardware Information.** We train all models on eight NVIDIA V100 GPUs and use PyTorch [6] for all our experiments.

**D Visualizations**

**Visual comparison of features.** Figure A2 plots visual comparisons of the extracted features with different frames as model inputs. We can summarize that the average frame shows the largest feature difference compared to the features extracted from the original clips. Our proposed Frame Condensing and Instance-Specific Prompt can effectively reduce the feature differences with the original clips and thus effectively improve the feature quality of condensed frames.
E Training Prompt without Condensed Frame

We attempt to train Instance-Specific Prompt without condensed frames, and the learned prompting parameters are visualized in Figure A4(c) and (d). We observe that the learned prompts have no intuitive semantics. To further understand this phenomenon, we plot the per-task accuracy and the loss curve of the prompts in Figure A4(a) and (b), respectively. We can summarize two points: (i) the learned prompts without condensed frames cannot fight against catastrophic forgetting, which indicates that the semantic information of the learned prompts is weak; (ii) the loss curve of the prompts without condensed frames is much higher than that with condensed frames, which implies that it is difficult to train a single prompts from scratch. Thus, it is crucial that the condensed frames provide a good initialization for the prompts to avoid the local minima of prompts.

References


