A. Algorithm Details

We introduce the details of the training process for our ST-Prompt in Algorithm A1.

Algorithm A1 Training Process of ST-Prompt

Initialization: Training dataset of task k, $D^k = \{(V_i^k, y_i^k), y_i^k \in L^k\}$; Pre-trained model, $F(;\Theta) = \{f_v(;\theta_v), f_t(;\theta_t)\}$; Task-specific prompt pools of task k, $\mathbf{P}_s^k$, $\mathbf{P}_t^k$ and $\mathbf{P}_c^k$; Temporal-modeling prompt pool, $\mathbf{P}_a$;

for $k = 1, 2, ..., K$ **do**
  Extract multi-grained features of videos in the training dataset D^k;
  Run K-Means algorithm on multi-grained features respectively to get the centroids of each kind of feature;
  Calculate and store the centroids of comprehensive, spatial and temporal features for task-specific prompt pools;
  Calculate and store the centroids of frame features for temporal-modeling prompt pool;
  for iter=1,2,...Iter$_{max}$ **do**
    Draw a video V_i^k and its label y_i^k in the training dataset D^k
    Extract the query feature $F_v(V_i^k)$, $F_s(V_i^k)$, $F_t(V_i^k)$ and $F_v(I_{ij}^k)$ by the pre-trained visual encoder $f_v(;\theta_v)$;
    Lookup task-specific prompts from k-th prompt pools by feature matching;
    Organize the visual and text task-specific prompts together, respectively, via Eq. 2;
    Prepend task-specific prompts to the video and text embedding feature: $V_{ip}^k = [P_i^{vk}; V_{ie}^k], W_{wp}^k = [P_i^{tk}; W_{we}^k]$
    Calculate the task-agnostic prompts by frame-level attention;
    Prepend the task-agnostic prompts to the video embedding feature as $V_{ip}^k = [P_{i1}^k; P_{i2}^k; ...; P_{iT}^k; P_i^{vk}; V_{ie}^k]$;
    Calculate per sample loss as $\mathcal{L}_i = \mathcal{L}(f_v(V_{ip}^k), f_t(W_{wp}^k), y_i^k)$
    for $P \in$ zip$(\mathbf{P}_s^k, \mathbf{P}_t^k, \mathbf{P}_c^k, \mathbf{P}_a)$ **do**
      $P \leftarrow P - \eta \nabla_P \mathcal{L}_i$
    end for
  end for
end for

The training loss $\mathcal{L}$ follows the contrastive similarity loss in CLIP [5]. The testing process of ST-Prompt is similar as the training process, where the most significant difference is that the selection of task-specific prompts is not constrained by the task-id. The prediction probability is calculated as $p = \frac{e^{<f_v(V_{ip}^k), f_t(W_{wp}^k)>}}{\sum_{w=1}^{W} e^{<f_v(V_{ip}^k), f_t(W_{wp}^k)>}}$ for classification.

B. Different Pooling Operations for Multi-grained Features

During prompts initialization and matching, we utilize average pooling operation along the temporal and spatial dimension to gain the spatial and temporal features from a fused spatio-temporal video feature, respectively. Here we discuss how other kinds of pooling operations influence the final results of matching in Table A1.

Operation	Stochastic	Max	Average
HMDB51	58.36	59.29	**60.14**
UCF101	83.68	84.15	**85.54**

Table A1. Ablation for the pooling operation of multi-grained features.

From the results, we can find that: first, no matter which kind of pooling operation is used, multi-grained prompts can bring performance improvement through diversified feature matching; second, the different types of pooling truly influence the final results of ST-Prompt, and average pooling achieves the better performance than stochastic and max pooling. The reason may be that average pooling can better obtain the overall information, while the others only focus on certain points.

C. Additional Results and Discussion of Hyperparameter Effects

In the main body, we have discussed the effects of hyperparameters in ST-Prompt by illustrating the results on HMDB51 with 5 steps. Here we implement some results on UCF101 with 10 steps in Fig. A1 to verify the generality of our conclusions of hyperparameter selection.

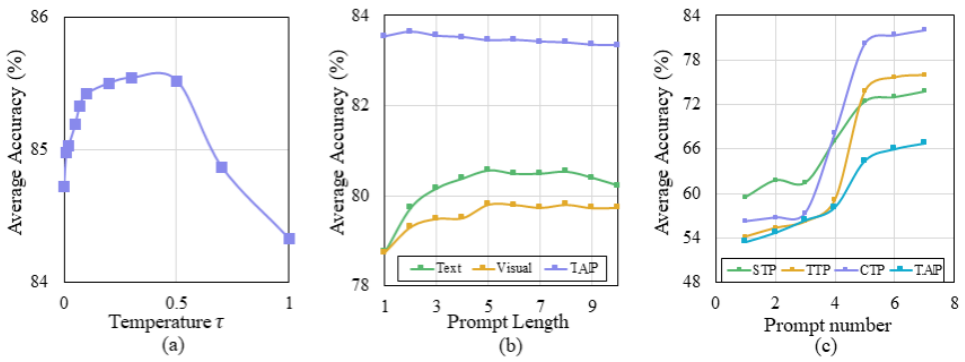

Figure A1. The effects of some hyperparameters in ST-Prompt on UCF101 with 10 steps.

Further, especially for the number of prompts, *i.e*. cluster numbers of K-Means, all results prove that more clusters in a task lead to higher accuracy. This is contrary to the conclusion in [7], which claims that its method is stably for different cluster numbers. We think the different conclusion is due to different tasks. For domain-incremental learning, the feature distribution is closed to each other in one task, but the situation is different in class-incremental learning. The classes are organized in a sequence with random order, so that the features of different classes in a task may not be close to each other, but scattered in different locations in the feature space. Hence, the final average accuracy is tightly corresponding to the value of cluster numbers.

D. Experiments on vCLIMB benchmark.

We present experiments on the vCLIMB benchmark in Table A2 for comparison with other video class-incremental learning methods. Our method still shows significant improvement compared to our main baseline method, *i.e*. S-liPrompts.

Method	K400		UCF101	
	10 Tasks	20 Tasks	10 Tasks	20 Tasks
Zero-shot	46.57	46.69	68.39	68.44
S-liPrompts	50.17	50.22	80.42	80.39
ST-Prompt	**54.61**	**54.49**	**85.14**	**85.29**
Upper-Bound	72.15	72.15	91.95	91.95

Table A2. Experiments on VCLIMB benchmark.

E. Dataset Details

In this paper, we follow the video frequently-used class-incremental benchmarks presented in TCD [3], which contains three standard action recognition datasets:

UCF101. UCF101 [6] is an action recognition dataset of realistic action videos, collected from YouTube, having 13.3K videos from 101 action categories, which can be found in https://www.crcv.ucf.edu/data/UCF101.php. In our experiments, we train 51 classes from UCF101 in the initial stage, and divide the remaining classes into groups of 10, 5, and 2 classes for class-incremental learning.

HMDB51. The HMDB51 dataset [2] is a large collection of realistic videos from various sources which consists of 6.8K videos and can be found in https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/. When optimizing model for HMDB51, the base model is trained using 26 classes and the rest classes are equally split into 5 and 25 groups.

Something-Something v2. Something-Something v2 [1] is a crowd-sourced dataset that shows humans performing predefined basic actions with everyday objects, which requires better temporal modeling ability than UCF101 and HMDB51. It can be found in https://developer.qualcomm.com/software/ai-datasets/something-something. In this paper, the model for Ssv2 is trained with 84 classes first and the remaining classes are grouped into 10 and 5 classes.

F. Experimental Details

Model and Hyperparameter Setup. CLIP ViT-B/16 [5] is employed as our backbone, and we follow the data preprocessing procedure and the basic training setting of TCD [3]. Each incremental training procedure takes 50 epochs. We use SGD optimizer without weight decay with an initial learning rate of 0.001 for all datasets, which is reduced by cosine learning rate schedule. For the hyperparameters of ST-Prompt, the length of task-specific prompts is set as 5 and the length of temporal-modeling prompts is 1. The total number of prompts in each prompt pool is equal to the number of classes in all settings.

Hardware Information. We train all models on 8 NVIDIA V100 GPUs and use PyTorch [4] for all our experiments.

References

[1] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 3

[2] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 2

[3] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *ICCV*, 2021. 2, 3

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 2019. 3

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 3

[6] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2

[7] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *NeurIPS*, 2022. 2