The Unreasonable Effectiveness of Large Language-Vision Models for Source-free Video Domain Adaptation Supplementary material

The supplementary material is organized as follows: In Sec. A we provide additional implementation details of our proposed method. Sec. B reports the results of additional experiments and ablation studies. Finally, in Sec. C we provide UMAP visualizations.

A. DALL-V implementation details

In this section we describe additional implementation details of DALL-V. The pseudo-code of the ensemble distillation in DALL-V is provided in Algo. 1.

Network architecture. We employ the CLIP pre-trained ViT-B/32 [4] backbone as the vision encoder for the source pre-training and the target adaptation phase. For the student network in the ensemble distillation phase (Sec. 4.3.2 of the main) we employ the CLIP pre-trained ResNet50 backbone to be comparable with the best SFVUDA competitors. Note that in all the training phases, we keep the CLIP vision encoders frozen to avoid losing the rich representation power of CLIP.

Following the prior works on parameter efficient fine-tuning [2, 1] of pre-trained models, we append trainable adapters $A(\cdot)\colon \mathcal{R}^d \rightarrow \mathcal{R}^d$ on top of the vision encoder of CLIP in all the phases of our DALL-V, where d is the input feature dimension. As shown in Fig. 1, the adapter is composed of a down-projection linear layer, ReLU non-linearity and a second up-projection layer, followed by a last ReLU. The dimension of the hidden features after the first down-projection layer is $1/4^{\text{th}}$ of the input dimension d.

Unlike [1], we do not use adapter on top of the language encoder, and the language embeddings are directly used to compute the output probability following Eq. 1 of the main paper. Similar to the vision encoder, we do not update the language encoder.

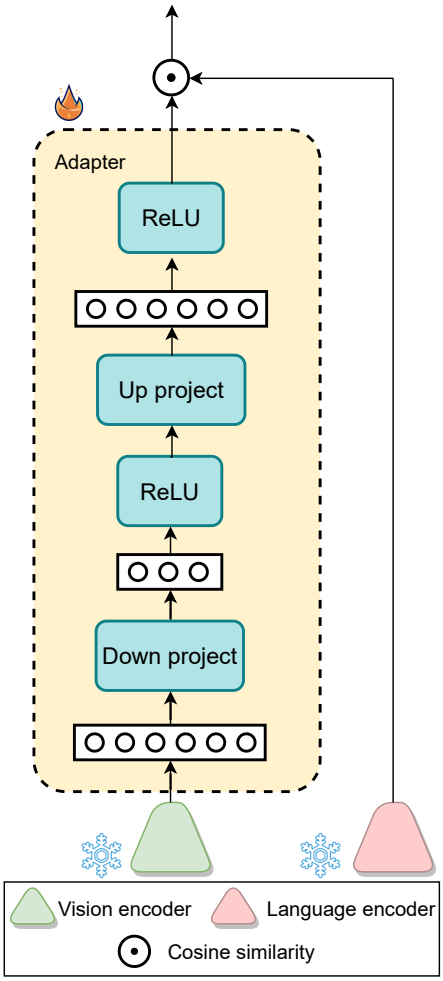


Figure 1: Architecture of the adapter and integration with the CLIP vision encoder. The 🔥 means the network is trainable, while the ❄ means the network is frozen.

Pseudo-labeling protocol. As mentioned in Sec. 4.3.1 of the main paper, we employ zero-shot CLIP (ViT-B/32) to obtain pseudo-labels of the target videos, which are then used to train the target adapter A^{T}. Given the pseudo-labels can be noisy, we opt for a pseudo-label filtering technique to reduce the impact of noisy pseudo-labels in the target adaptation phase. In detail, we follow [7] to obtain a set of class-wise thresholds to filter out the noisy pseudo-labels. We consider the distribution of the confidence values of all the

Algorithm 1 Pseudo-code of *ensemble distillation* in DALL-V in a PyTorch-like style

```
# class_names (list(str)): names of classes
# class_idxs (list(int)): index of classes
# prompts (list(str)): textual prompts
# ensemble (nn.Module): ensemble of CLIP('ViT-B/32'),
# source adapter, and target adapter.

# create the student backbone
st_backbone: nn.Module = CLIP('RN50')
st_backbone.eval() # freeze the student

# add the student adapter
st_backbone.adapter: nn.Module = Adapter()

# get the textual embeddings
prompts: list(str) = combine(prompts, class_names)
prompts_z: Tensor = st_backbone.language_encoder(
   prompts
)

# distill
for epoch in epochs:
   for x in target_loader: # use the whole target
      # pseudo-label with the ensemble of teachers
      ensemble_p: Tensor = ensemble(x)
      pseudo_y = ensemble_p.max(dim=-1)[1]

      # forward the images
      out: Tensor = st_backbone.vision_encoder(x)
      out: Tensor = st_backbone.adapter(out)

      # evaluate the zero-shot probabilities
      images_p: Tensor = cosine_sim(out, classes_z)
      images_p: Tensor = softmax(images_p)

      # calculate the loss
      discrepancy_loss: Tensor = KL(
         images_p, ensemble_p
      )
      ce_loss: Tensor = CrossEntropy(
         images_p, pseudo_y
      )

      # calculate the loss
      loss: Tensor = (
         alpha * discrepancy_loss +
         (1 - alpha) * ce_loss
      )

      # update the adapter parameters
      loss.backward()
      update(st_backbone.adapter.params)
```

target predictions associated with a class and set the threshold as the 80$^{\text{th}}$ percentile. All the predictions for that class having confidence lower than the chosen threshold are filtered out and not used in target adaptation.

B. Additional experiments

Parameter/Performance trade-off. As outlined in Sec. 4.3.2 of the main paper, in DALL-V we fine-tune only the adapter $\bar{A}$, appended to the *student* vision encoder $\bar{G}_{\mathtt{V}}(\cdot)$, during the ensemble distillation phase. While this design choice substantially reduces the number of trainable parameters, it can be sub-optimal in cases where the target domain differs greatly from the CLIP training distribution. Thus, it presents a trade-off between performance and parameter efficiency.

To better understand this trade-off, we compare the adapter fine-tuning with the *full* fine-tuning of the vision encoder, where the entire encoder is trainable. Note that the adapter is not used in the full fine-tuning experiment, as done in prior works [2]. We conducted this ablation study on the ***UCF-HMDB***$_{full}$ benchmark and report the results in Tab. 1. We observe that for the **HMDB** $\rightarrow$ **UCF** adaptation setting, fine-tuning all the weights of the encoder leads to an improvement of 2.5% when compared with training only the adapter weights. On the other hand, for the reverse adaptation setting of **UCF** $\rightarrow$ **HMDB**, fine-tuning all the weights is detrimental to the performance, with a drop of 0.9% points. Thus, overall the full fine-tuning baseline outperforms the adapter model by 0.8% on average, at the cost of increased training time due to the significantly higher magnitude of trainable weights in the network. To summarize, fine-tuning only the adapter, which is $\sim$ 0.25% of the full model size, is highly parameter-efficient and, at the same time, maintains comparable performance. This ablation study's findings align with the usage of adapters in NLP tasks [2].

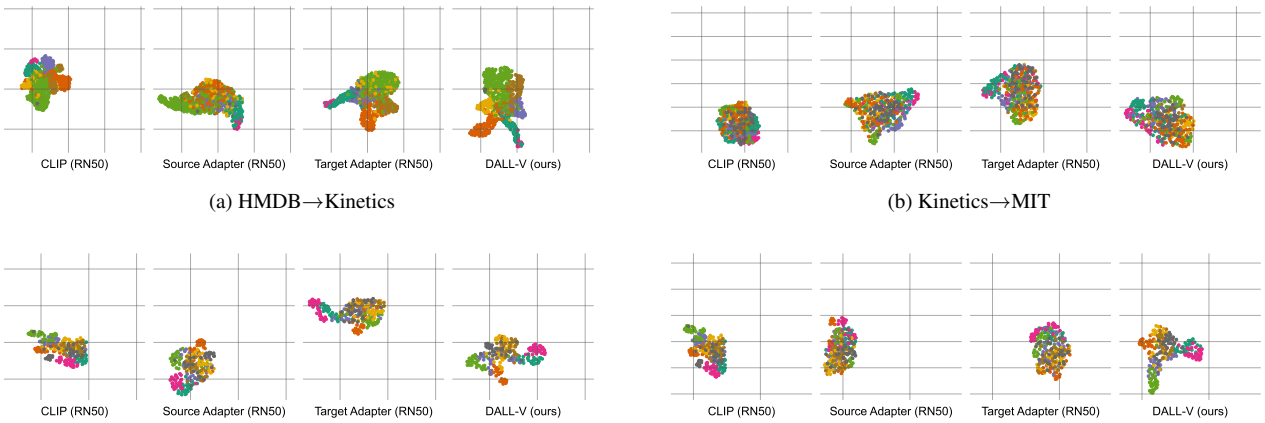
	Trainable # params	H→U	U→H	**Avg.**
Adapter	0.26M	93.1	**88.9**	91.0
Full fine-tune	102M	**95.6**	88.0	**91.8**

Table 1: Comparison of performance between adapter fine-tuning and full encoder fine-tuning on the ***UCF-HMDB***$_{full}$ benchmark. "M" stands for million.

Prompts
a photo of action [CLS]
a picture of action [CLS]
Human action of [CLS]
[CLS], an action
[CLS] this is an action
[CLS], a video of action
Playing action of [CLS]
[CLS]
Playing a kind of action, [CLS]
Doing a kind of action, [CLS]
Look, the human is [CLS]
Can you recognize the action of [CLS]?
Video classification of [CLS]
A video of [CLS]
The man is [CLS]
The woman is [CLS]

Table 2: The list of all 16 prompts used in DALL-V.

Effectiveness of prompts. It has been shown in the NLP [5, 3] and vision-based [9, 8, 6] tasks that *prompting* has a significant impact on the performance when re-



(a) HMDB→Kinetics (b) Kinetics→MIT

(c) MIT→HMDB (d) ARID→HMDB

Figure 2: UMAP visualisation of the feature space on ***Daily-DA*** for the zero-shot CLIP (RN50), source, target and final DALL-V models. The chart shows the ability of our proposed DALL-V to efficiently cluster the action categories in the features space by combining knowledge from the other three models.

Prompts	***UCF-HMDB***$_{full}$	***Daily-DA***
Mixed (Ours)	89.7	**48.1**
Only-image	**89.8** (+0.1%)	47.7 (-0.6%)
CLIP [4]	83.2 (-6.5%)	43.7 (-4.4%)

Table 3: Impact of prompts on the zero-shot performance using the ***UCF-HMDB***$_{full}$ and ***Daily-DA*** benchmarks.

purposing (or fine-tuning) large-scale pre-trained models on downstream tasks. Following ActionCLIP [6], we also choose a set of hand-crafted language prompts for obtaining pseudo-labels and training DALL-V. The complete set of prompts is reported in Tab. 2.

To further assess the impact of the prompts in SFVUDA, we design an ablation study where we vary the prompts provided to the language encoder. In detail, we create an alternate version of the prompts listed in Tab. 2, where we replace all the occurrences of the token "video" with the token "image". This is done with the motivation that we obtain predictions at the frame level, which are then fused in the output space of the network. We call this baseline an "only-image" since the prompts do not contain the token "video". We denote the prompts in our DALL-V as "mixed", given it uses a mixture of both kinds of tokens (*i.e.*, "image" and "video") in the prompts. Finally, we create another baseline that uses the hand-engineered prompts used in the original CLIP paper (we refer the reader to [4] for the full list). We report the results of the experiments on both benchmarks in Tab. 3. Note that we simply report the zero-shot validation performance in Tab. 3 and not the performance after the final distillation step.

From Tab. 3, we observe that the "only-image" baseline has comparable performance compared to our DALL-V (or "mixed"). This kind of behaviour is expected because the predictions from the CLIP backbone are obtained at frame level and the network is activated more or less similar when the "video" token in the prompt is replaced by "image".

On the contrary, usage of original prompts from the CLIP paper resulted in big drops of 6.5% and 4.4%, on ***UCF-HMDB***$_{full}$ and ***Daily-DA***, respectively. We can infer from these results that shorter and more action-oriented prompts (as in "mixed" or "only-image") are more beneficial for the SFVUDA task.

C. Additional visualizations

In Fig. 2 we plot the UMAP visualizations of the features produced by the zero-shot CLIP (RN50), source, target and final DALL-V models for the ***Daily-DA*** dataset, which were omitted for space issues from the main paper. The chart shows that on this benchmark, similarly to what is shown for ***UCF-HMDB***$_{full}$ in the main paper, our proposed DALL-V method is able to benefit from all intermediate complementary models in order to enforce a more class-discriminative modeling of the target domain.

References

[1] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv*, 2021. 1

[2] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer

learning for nlp. In *ICML*, pages 2790–2799. PMLR, 2019. 1, 2

[3] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv*, 2021. 2

[4] 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

[5] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv*, 2020. 2

[6] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *arXiv*, 2021. 2, 3

[7] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In *NeurIPS*, 2021. 1

[8] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022. 2

[9] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022. 2