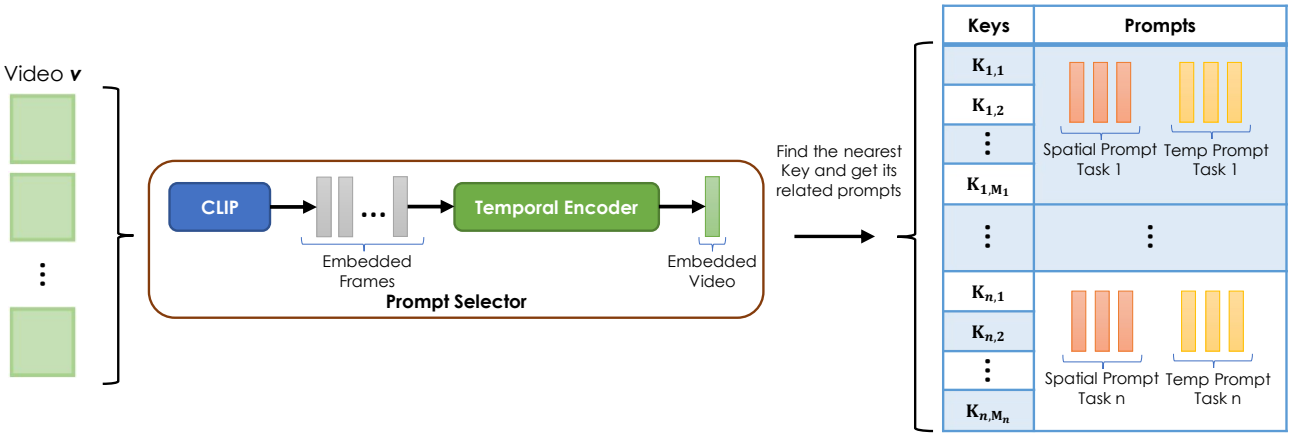

Figure 4. **Outline of Prompt Selector.** Our prompt selector leverages the CLIP f_{sp} and Temporal Encoder f_{tp} as the query function $q = f_{sp} \circ f_{tp}$ to select the suitable task prompts based on the similarity between the encoded video $q(\boldsymbol{v})$ and the keys K of all task prompts.

A. Prompt Selector

In Section 3.3, we outlined our key components in PIVOT. The Prompt Selector module in PIVOT selects the most suitable prompts for classifying a given input. Here we delve into the details of our Prompt Selector module.

Given a class incremental sequence of n tasks, PIVOT learns a set of n task-specific prompts $P_i = \{(P_i^{sp}, P_i^{tp})\}_{i=1}^{n}$, where each prompt P_i is a key-value pair (K_i, P_i). Note that K_i consists of the encoded representations of the M_i labels found in task i. These representations are obtained using the CLIP Text encoder. As can be seen in Figure 4, our prompt selector passes the video $\boldsymbol{v}$ through the CLIP visual encoder f_{sp} and our temporal encoder f_{tp} to compute a query $f_{tp}(f_{sp}(\boldsymbol{v}))$. Following the tokenization procedure of CLIP, each input representation includes a [class] token. The computation of our query takes into account the [class] tokens encoded in the representations of both the visual and temporal encoders. PIVOT selects the corresponding task prompts for $\boldsymbol{v}$ based on the similarity between the query and all keys.

We present Algorithm 2, which summarizes the prompt selection process of PIVOT and how it uses the selected prompts. Likewise, Algorithm 1 clarifies the forward pass of our base PIVOT w/o prompts, which does not require the task-specific prompts.

Algorithm 1: PIVOT w/o prompts Forward Pass

Data:

$\boldsymbol{Y} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_M)$; `/* The representations of all learned classes computed with the CLIP Text Encoder. */`

Components:

f_{sp}, f_{tp} ; `/* CLIP Model, Temporal Encoder */`

$\gamma(.,.), f_{cls}$; `/* Cosine distance function, Select the class label whose representation is most similar to the video */`

Forward Pass:

$\boldsymbol{v}_{tp} = f_{tp}(f_{sp}(\boldsymbol{v}))$ where $\boldsymbol{v} \in D_t$; `/* Compute the video embedding */`

$y = f_{cls}(\boldsymbol{v}_{tp}, \boldsymbol{Y})$; `/* Classify the video */`

B. Number of Trainable Parameters in PIVOT

Considering the implementation details presented in section 4, we analyze and compare the trainable parameters of our PIVOT model against the baseline models. Note that the vCLIMB baselines utilize TSN as a backbone and train it at every task. On the other hand in PIVOT w/o prompts, we leverage the extensive knowledge of CLIP by freezing its visual and text encoders, so we only train our temporal encoder. In PIVOT, we further learn task-specific prompts, which does not result in a significance increase in the number of parameters. As a result of leveraging the knowledge in CLIP without fine-tuning it,

Table 4. **The number of trainable parameters.** All the vCLIMB Baselines use the TSN with ResNet50 as a backbone, in addition to a linear layer to perform the classification. These models have approximately the same number of parameters. We consider ActivityNet, which have 200 classes in total, to compute the number of parameters of the linear layers. We note that PIVOT w/o prompts and PIVOT results in an order of magnitude reduction in the number of parameters to train in video class incremental learning.

Model	Num. Trainable Parameters
vCLIMB Baselines	23.610632×10^6
PIVOT (10-task)	9.496064×10^6
PIVOT (20-task)	9.534464×10^6
PIVOT w/o prompts	9.457664×10^6

Table 5. **Prompt Hyper-parameters.** We vary the prompt length (L) and number of prompts per task (N_p) and report PIVOT performance. To assess L, we fix $N_p = 1$ and use the same L for both spatial and temporal prompts $(L = L_{sp} = L_{tp})$. Likewise, for N_p, we set L to its optimal value $(L = 3)$.

Length of prompts	Acc	Num. Prompts	Acc
PIVOT $(L = 1)$	73.2%	PIVOT $(N_p = 1)$	**73.8%**
PIVOT $(L = 3)$	**73.8%**	PIVOT $(N_p = 2)$	72.60%
PIVOT $(L = 5)$	73.1%	PIVOT $(N_p = 4)$	70.59%
PIVOT $(L = 7)$	72.7%	PIVOT $(N_p = 6)$	69.11%

we substantially reduce the number of total trainable parameters. Table 4 shows that PIVOT w/o prompts and PIVOT train at most 40.56% of the parameters that the vCLIMB baselines train. It is worth highlighting that the task-specific prompts correspond to an increase of 0.40% and 0.80% of PIVOT w/o prompts parameters in the 10-task and 20-task scenarios, respectively. Thus, the resulting PIVOT is comparable in the number of parameters to PIVOT w/o prompts.

C. Prompt Hyper-parameter Analysis

As observed in Table 5, we explored different configurations for the prompt length L and number of prompts N_p per task on the validation set of the most challenging dataset we evaluated, ActivityNet. We considered the same L for both spatial and temporal prompts $(L = L_{sp} = L_{tp})$. PIVOT is more sensitive to the number of prompts per task than their length. It is important to note that $N_p = 1$ and $L = 3$ for both spatial and temporal prompts work better for ActivityNet. For simplicity, we use the same setup for the other datasets we evaluated.

Algorithm 2: PIVOT Forward Pass

Data:

$\boldsymbol{Y} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_M)$; /* The representations of all learned classes computed with the CLIP Text Encoder */

Components:

f_{sp}, f_{tp} ; /* CLIP Model, Temporal Encoder */

$f_{sp} = f_{sp}^{e} \circ f_{sp}^{s}$' ; /* Where f_{sp}^{e} is the input layer and f_{sp}^{s} the self-attention layers */

$f_{sp}^{avg}, f_{tp}^{avg}$; /* Compute an average pooling through the added spatial and temporal prompts */

$(K_i, P_i)_{i=1}^{n}$; /* Set of prompts for the n tasks */

$K_i \in \mathbb{R}^{M_i \times D_m}$; /* Keys of task i, where M_i is the number classes of task i */

$P_i = (P_i^{sp}, P_i^{tp})$; /* Spatial and Temporal Prompts of task i */

$\gamma(.,.), f_{cls}$; /* Cosine distance function, Select the class label whose representation is most similar to the video */

Forward Pass:

$\boldsymbol{v}_{tp} = f_{tp}(f_{sp}(\boldsymbol{v}))$ where $\boldsymbol{v} \in D_t$; /* Compute the Query */

$P_k = f_{min}(\gamma(\boldsymbol{v}_{tp}, K))$; /* Select the task prompt that is closet the Query */

$\boldsymbol{v}_{sp}^{e} = [P_k^{sp}; f_{sp}^{e}(\boldsymbol{v})]$; /* Add the spatial prompt */

$\boldsymbol{v}_{sp}^{avg} = f_{sp}^{avg}(f_{sp}^{s}(\boldsymbol{v}_{sp}^{e}))$; /* Compute the representation of the frames per video */

$\boldsymbol{v}_{tp} = f_{tp}^{avg}(f_{tp}([P_k^{tp}, \boldsymbol{v}_{sp}^{avg}]))$; /* Add the temporal prompt and Compute the final video embedding */

$y = f_{cls}(\boldsymbol{v}_{tp}, \boldsymbol{Y})$; /* Classify the video */