Supplementary
Too Many Frames, Not All Useful: Efficient Strategies for Long-Form Video QA

Jongwoo Park [1] **Kanchana Ranasinghe** [1] **Kumara Kahatapitiya** [1]
Wonjeong Ryoo **Donghyun Kim** [2] **Michael S. Ryoo** [1]

[1]Stony Brook University [2]Korea University

jongwoopark@cs.stonybrook.edu
https://github.com/jongwoopark7978/LVNet

Appendix

A.1 Additional Ablations

In this section, we present additional experiments conducted to inform the LVNet's design. We have tested different LLMs and experimented with various scales of the visual feature map.

Table A.1: **Ablations on EgoSchema [2]:** We evaluate different design decisions of our framework on EgoSchema 500-video subset for zero-shot VQA.

(a) **Choice of LLM**: We consider different options for our LLM for video QA.

LLM	Acc. (%)
GPT-3.5	61.0
GPT-4	65.4
GPT-4o	68.2

(b) **Effect of Patch Size on Keyword Matching in CKD**: We consider different patch sizes to measure the similarity between the image and the keyword.

Patch Size	Acc. (%)
1x1	63.6
7x7	66.2
14x14	68.2

Choice of LLM: Our results show that GPT-4o outperforms both GPT-4 and GPT-3.5 by 2.8% and 7.2%, respectively. Given that GPT-4o is more cost-effective and lightweight compared to GPT-4, we have selected it as our default LLM.

Effect of Patch Size on Keyword Matching in CKD: Since keywords can represent activities spanning the entire image or confined to a small region, we adjust the resolution of the visual feature map output from the spatially aware contrastive image pre-training (CLIP) network [3] to match keywords. Our findings show that higher resolutions lead to better accuracy. In LVNet, we use a 14×14 feature map and determine the confidence level of the keyword by selecting the maximum value between the 14×14 patches and the keyword's text embedding.

Workshop on Video-Language Models at 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

A.2 Algorithms in Detail

Our algorithms are presented in full detail in Algorithm 1, Algorithm 2, and Algorithm 3. TSC in Algorithm 1 extracts per-frame visual features using ResNet-18, followed by an iterative clustering procedure to identify n non-overlapping frame sets. Within each of the n sets, we uniformly sample $\leq \tau$ frames, obtaining a total of $T_a \leq \tau \times n$ frames. For example, LVNet sets $\psi = 5, \lambda = 12, \tau = 18$, resulting in approximately $n \sim 25$ and $T_a \sim 390$ on the EgoSchema dataset. CKD in Algorithm 2 selects top L frames based on similarity/confidence scores, which are calculated using cosine similarity between frames and keywords with CLIP-B/16. LVNet employs $L = 32, len(K) \leq 25$ on the EgoSchema dataset. FKD in Algorithm 3 sorts frames and their corresponding keywords by confidence scores, and reorder the K frames with the lowest scores temporally. It groups frames sequentially into visual templates, each consisting of N frames. From each template, the M frames and keywords most relevant among the N pairs are selected using GPT-4o. We set $L = 32, K = 16, N = 8, M = 3$.

Algorithm 1: Temporal Scene Clustering

```
1: Require: ResNet-18 [1] pretrained on imagenet dataset $f$, frame list $\mathbf{List}_{frame}$, image index
   list $\mathbf{List}_{index} \in \{1, \ldots, N\}$, minimum number of list length $\psi$, temperature $\lambda$, number of
   sample $\tau$, function to find index of $x$ in list w index($x$, w), and function to sort list sort(List)
2: for all $img^i$ in $\mathbf{List}_{frame}$ do
3:     $\mathbf{F}^i \leftarrow f(img^i)$
4:     $\mathbf{List}_{feat}$.insert($\mathbf{F}^i$)
5: end for
6: for all $\mathbf{F}^i$ in $\mathbf{List}_{feat}$ do
7:     $\mathbf{List}_{dist} \leftarrow \frac{\sum_y \sum_x \sqrt{(\mathbf{F_i} - \mathbf{List}_{feat})^2}}{x \times y}$
8:     $\mathbf{M}_{dist}$.insert($\mathbf{List}_{dist}$)
9: end for
10: while length of $\mathbf{List}_{index} > \psi$ do
11:     $\mathbf{List}_{sample} \leftarrow \emptyset$
12:     $\mathbf{List}_{\delta} \leftarrow \emptyset$
13:     $i \leftarrow \mathbf{List}_{index}$.pop(0)
14:     $\mathbf{p}^i \leftarrow$ softmax($\mathbf{M}^i_{dist}$)
15:     $\mu_{\mathbf{p}^i}, \sigma_{\mathbf{p}^i} \leftarrow$ mean($\mathbf{p}^i$), std($\mathbf{p}^i$)
16:     $\beta \leftarrow \mu_{\mathbf{p}^i} - \sigma_{\mathbf{p}^i} \sum_{i=0} e^{1-i/\lambda}$
17:     for all prob in $\mathbf{p}^i$ do
18:         if prob < $\beta$ then
19:             $\mathbf{List}_{selected}$.insert(index(prob, $\mathbf{p}^i$))
20:         end if
21:     end for
22:     for all $\gamma$ in $\mathbf{List}_{selected}$ do
23:         $\delta \leftarrow \gamma$ $th$ value in $\mathbf{List}_{index}$
24:         $\mathbf{List}_{\delta}$.insert($\delta$)
25:         $\mathbf{List}_{index}$.pop($\gamma$)
26:     end for
27:     $\mathbf{List}_{\delta}$.insert($i$)
28:     $\mathbf{List}_{sample} \leftarrow$ sample $\tau$ items from $\mathbf{List}_{\delta}$
29:     sort($\mathbf{List}_{sample}$)
30:     for all $frame^j$ in $\mathbf{List}_{frame}$ do
31:         if $j$ in $\mathbf{List}_{sample}$ then
32:             Outputs.insert($frame^j$)
33:         end if
34:     end for
35: end while
```

Algorithm 2: Keyword-Image Matching Process in CKD

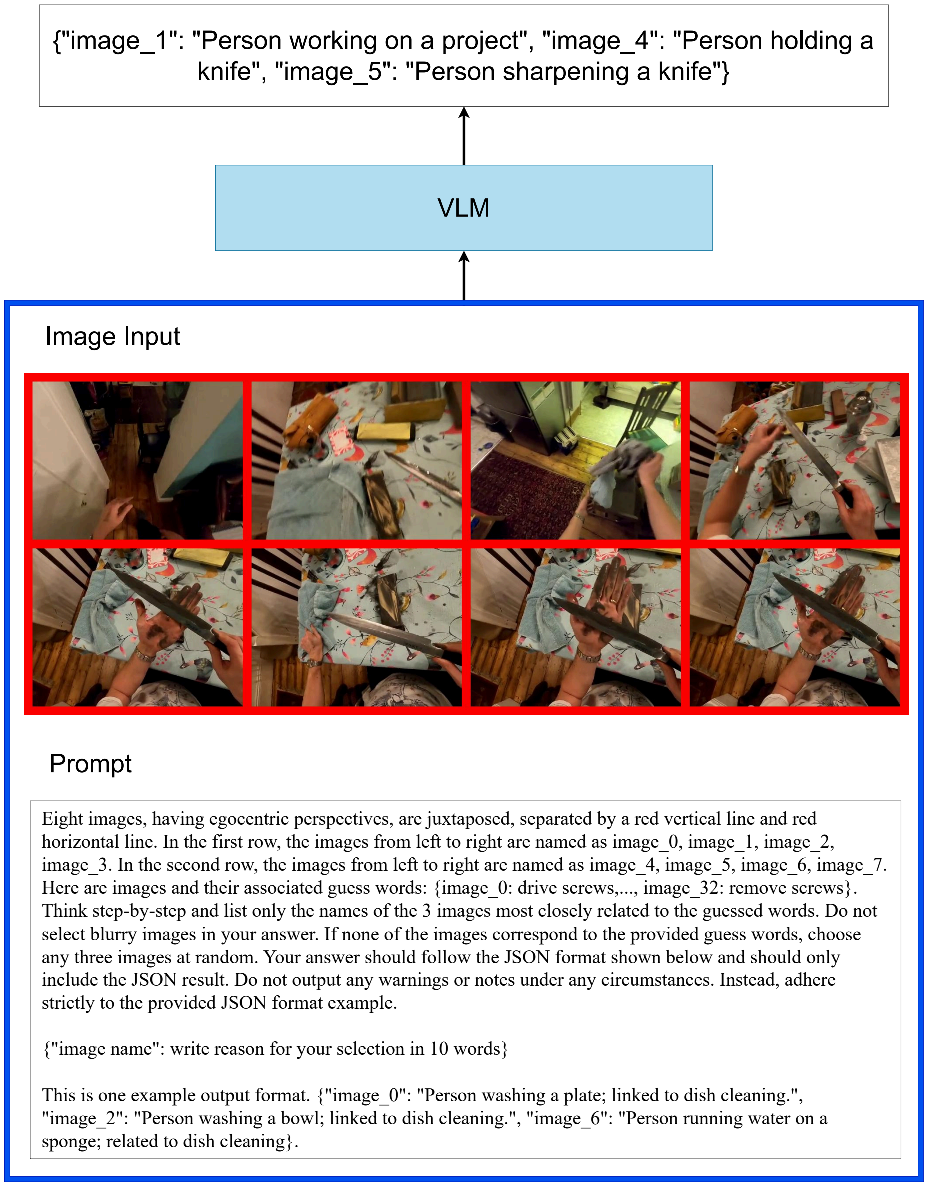
```
1: Require: keyword set $\mathbf{K}$, image set $\mathbf{I}$, total length of selected image set $L$, function to calculate similarity matrix $\mathbf{sim}(\mathbf{K}, \mathbf{I})$, function to sort similarity matrix and return indices $\mathbf{sort}(\mathbf{S})$
2: $\mathbf{S} \leftarrow \mathbf{sim}(\mathbf{K}, \mathbf{I})$
3: $\mathbf{S}_{sorted}, \mathbf{idx}_{sorted} \leftarrow \mathbf{sort}(\mathbf{S})$
4: Initialize $\mathbf{P}_{best}$ as an empty list
5: Initialize $\mathbf{I}_{selected}$ as an empty set
6: while length of $\mathbf{I}_{selected} < L$ do
7:   for $k \in \mathbf{K}$ do
8:     for $i \in \mathbf{I}$ do
9:       $i_{index} \leftarrow \mathbf{idx}_{sorted}[k][i]$
10:      if $i_{index}$ not in $\mathbf{I}_{selected}$ then
11:        $\mathbf{P}_{best}$.insert($k, i_{index}$)
12:        $\mathbf{I}_{selected}$.insert($i_{index}$)
13:        break
14:      end if
15:    end for
16:    if length of $\mathbf{I}_{selected} \geq L$ then
17:      break
18:    end if
19:  end for
20: end while
21: return $\mathbf{P}_{best}$
```

Algorithm 3: Fine Keyframe Detection Process (FKD)

```
1: Require: keyword set $\mathbf{K}$, image set $\mathbf{I}$, similarity score list $\mathbf{S}$, total length $L$, number of low similarity indices $K$, number of frames per visual template $N$, number of keyframes selected per visual template $M$, function to sort by similarity $\mathbf{sort}(\mathbf{S})$, function to order indices temporally $\mathbf{temporal\_order}()$
2: $\mathbf{idx}_{sorted} \leftarrow \mathbf{sort}(\mathbf{S})$
3: $\mathbf{idx}_{low\_sim} \leftarrow \mathbf{idx}_{sorted}[-K:]$
4: $\mathbf{idx}_{temporal} \leftarrow \mathbf{temporal\_order}(\mathbf{idx}_{low\_sim})$
5: $\mathbf{idx}_{final} \leftarrow \texttt{concatenate}(\mathbf{idx}_{sorted}[: -K], \mathbf{idx}_{temporal})$
6: $\mathbf{I}_{ordered}, \mathbf{K}_{ordered} \leftarrow \mathbf{I}[\mathbf{idx}_{final}], \mathbf{K}[\mathbf{idx}_{final}]$
7: sets $\leftarrow \texttt{create\_sets}(\mathbf{I}_{ordered}, \mathbf{K}_{ordered}, L//N)$
8: for each set $\in$ sets do
9:   $\mathbf{I}_{selected} \leftarrow \texttt{select\_top\_M}(\text{set}, M)$
10: end for
11: return $\mathbf{I}_{selected}$
```

A.3 Prompting: Fine Keyframe Detector

We prompt the VLM to select frames that are most compatible with the list of given keywords. Each template image contains 8 images, and their order is described in language (e.g. top left to right, bottom left to right) and the VLM outputs the selected images according to our prompting as described in Figure A.1.



{"image_1": "Person working on a project", "image_4": "Person holding a knife", "image_5": "Person sharpening a knife"}

Eight images, having egocentric perspectives, are juxtaposed, separated by a red vertical line and red horizontal line. In the first row, the images from left to right are named as image_0, image_1, image_2, image_3. In the second row, the images from left to right are named as image_4, image_5, image_6, image_7. Here are images and their associated guess words: {image_0: drive screws,..., image_32: remove screws}. Think step-by-step and list only the names of the 3 images most closely related to the guessed words. Do not select blurry images in your answer. If none of the images correspond to the provided guess words, choose any three images at random. Your answer should follow the JSON format shown below and should only include the JSON result. Do not output any warnings or notes under any circumstances. Instead, adhere strictly to the provided JSON format example.

{"image name": write reason for your selection in 10 words}

This is one example output format. {"image_0": "Person washing a plate; linked to dish cleaning.", "image_2": "Person washing a bowl; linked to dish cleaning.", "image_6": "Person running water on a sponge; related to dish cleaning}.

Figure A.1: **Prompt for Fine Keyframe Detection**: The figure illustrates the input image, the prompt provided to the VLM, and the output. The input image represents a visual template composed of eight frames, and the prompt requests the three best frames along with their corresponding keywords. The output displays the top three selected frames and their associated keywords.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *ArXiv*, abs/2308.09126, 2023.

[3] Kanchana Ranasinghe, Brandon McKinzie, Sachin Ravi, Yinfei Yang, Alexander Toshev, and Jonathon Shlens. Perceptual grouping in contrastive vision-language models. In *ICCV*, 2023.