Enhancing Visual Planning with Auxiliary Tasks and Multi-token Prediction

Supplementary Material

This supplementary material is organized as follows. First we provide additional experiments in Section 1. Then we provide more details about the training and evaluation process of our method in Section 2. Finally we provide qualitative analysis of our method for both VPA and LTA tasks in Section 3.

1. Additional Experiments

1.1. Ablations on Visual Encoder

Table 1 shows the performance of different visual encoders in our model. We use Llama2-7B as the LLM for all experiments. The results show that MetaCLIP as the visual encoder outperforms VideoCLIP across all metrics.

Visual	T=3			T=4		
Encoder	SR	mAcc	mIoU	SR	mAcc	mIoU
VideoCLIP	25.6	45.3	67.7	18.2	43.3	71.9
MetaCLIP	**29.1**	**50.1**	**69.4**	**20.5**	**47.5**	**73.9**

Table 1. **Ablations on Visual Encoder on COIN Dataset.** We use Llama2-7B as the LLM. MetaCLIP as the visual encoder outperforms VideoCLIP across all metrics.

1.2. State Prediction as An Auxiliary Task

In addition to Goal Modality Augmentation and Goal Prediction, we also explore State Prediction as an auxiliary task. Specifically, given current observation $\mathcal{O}$ and a sequence of future actions $\mathcal{A} = \{a_1, a_2, \ldots, a_H\}$, the model needs to predict a sequence of future object states $\mathcal{S} = \{s_1, s_2, \ldots, s_H\}$. Each object states s_i is a short text (*e.g. 'the sofa cover is stretched out and fitted onto the sofa'*) describing the object states after the user performs the action a_i. However, we do not have the object states annotation. To generate the object states, a straightforward approach is to extract captions from the input video using pre-trained large vision-language models (VLMs). However, while state-of-the-art VLMs can reliably perform object recognition, recent works show that they consistently struggle to capture the objects' physical states [5]. Motivated by prior works [6, 8, 9], we leverage LLMs to generate language descriptions of object states based on their commonsense knowledge. Specifically, we feed the action label and the high-level task goal to the LLMs and prompt for descriptions about possible object states before that action. Following prior works [6], we adopt Chain-of-thought Prompting to first describe the details of action steps and then describe the object states according to the details of the steps. The prompt is designed as:

```
  First, describe the details of
[action] for [goal] with one verb.
Second, use 3 sentences to describe the
object states before [action], avoiding
using [verb].
```

In this prompt, `[verb]` refers to the verb from the action name (*e.g.*, *'install'*) to increase the description diversity. To generate the object states after one action, we simply replace "before" with "after" in the above prompt.

Table 2 shows the results of adding State Prediction as an auxiliary task. We find that using State Prediction does not yield the optimal results. Specifically, when planning for the future 3 actions, the model variant without State Prediction (w.o. SP) is only 0.1% lower in SR compared with the model variant with State Prediction (w. SP). When planning for the future 4 future actions, the w.o. SP model variant achieves the best results across all metrics. We leave the exploration in this direction for future work.

Aux. Tasks	T=3			T=4		
	SR	mAcc	mIoU	SR	mAcc	mIoU
w. SP	**29.2**	**50.7**	**69.5**	19.5	46.7	73.1
w.o. SP	29.1	50.1	69.4	**20.5**	**47.5**	**73.9**

Table 2. **Effects of State Prediction.** SP: State Prediction. When planning for the next 3 future actions, our model with all auxiliary tasks performs best across all metrics. When planning for the next 4 actions, our model without SP leads to the best results.

2. Additional Implementation Details

2.1. Training

We train our model for 1 epoch with a batch size of 1024. We set gradient accumulation step to 16 to reduce GPU memory usage. For all experiments, we use LoRA [1] for efficient fine-tuning. The LoRA parameters are set to $r = 64$ and $alpha = 128$. In the auxiliary task pre-training stage, we set learning rate to $3e$-4, batch size to 1024 and train the model for 1 epoch. When finetuning the model on the VPA task, we set learning rate to $6e$-4, batch size to 512 and optimize for 4 epochs.

2.2. Evaluation

The output space of the MLLM is unconstrained. To evaluate our model, we need to map the free-form text to the discrete action indices. To achieve this, we use Sentence-

Task Type	Input	Output	Psuedo-Prompt
Goal Modality Aug.	Goal (text)	Action (text)	*`<obs>` The person is trying to achieve `<goal text>`. What are the next steps?*
	Goal (image)	Action (text)	*`<obs>` The person is trying to achieve the goal `<goal image>`. What are the next steps?*
	-	Action (text)	*`<obs>` What are the next steps of the person?*
Goal Prediction	-	Goal (text)	*`<obs>` What is the person trying to achieve?*
State Prediction	Action (text)	State (text)	*`<obs>` The person will take these `<actions>`. What are the states before and after these actions?*

Table 3. **Example Instructions and Responses for All Tasks.** The model takes in the instruction and generates the response. `<obs>` denotes the observation representation. During training, we replace `<obs>` with a video or an image, or current object states in the form of text. `<goal image>` denotes the embedding of the goal image.

BERT [7] to compute the text embeddings for the MLLM's free-form output and all candidate actions in the datasets following prior works [2–4]. Then we compare the text embedding of the free-form output with the text embeddings of all action candidates and choose the one with the highest cosine similarity as the target action.

2.3. Prompt Design

Our framework consists of multiple different tasks. We design task-specific prompts to handle all types of tasks. Table 3 shows the template for designing the instructions and responses. The model takes the instructions as the input and generates the responses.

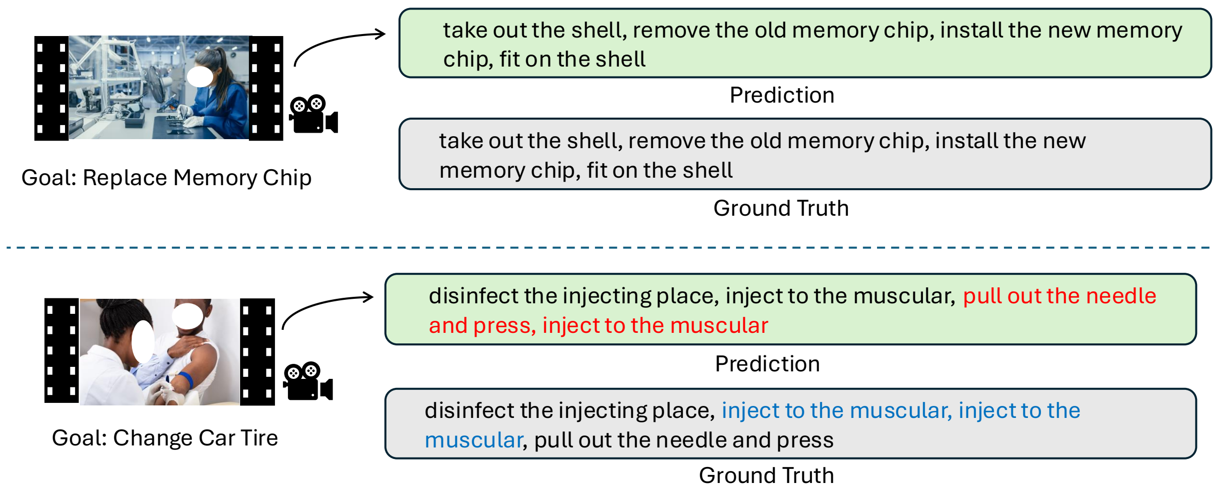
The instructions are constructed purely from the ground truth annotations from the datasets. The annotations include the start time, end time, and labels for each action. Specifically, when replacing `<obs>` with a video, we use a 50s video before the first action to predict. When replacing `<obs>` with an image, we use the frame right before the first action to predict. When replacing `<obs>` with text, we use the object states generated using the method described in Section 1.2. `<goal image>` is replaced with the last frame of the last future action segment to predict as the goal image.

For VPA and Goal Modality Augmentation, the responses are the concatenation of future actions to predict. For Goal Prediction, we leverage the task type label (e.g. Assemble Sofa) from the dataset annotations as the responses. For State Prediction, we generate the responses using the method described in Section 1.2.

3. Qualitative Analysis

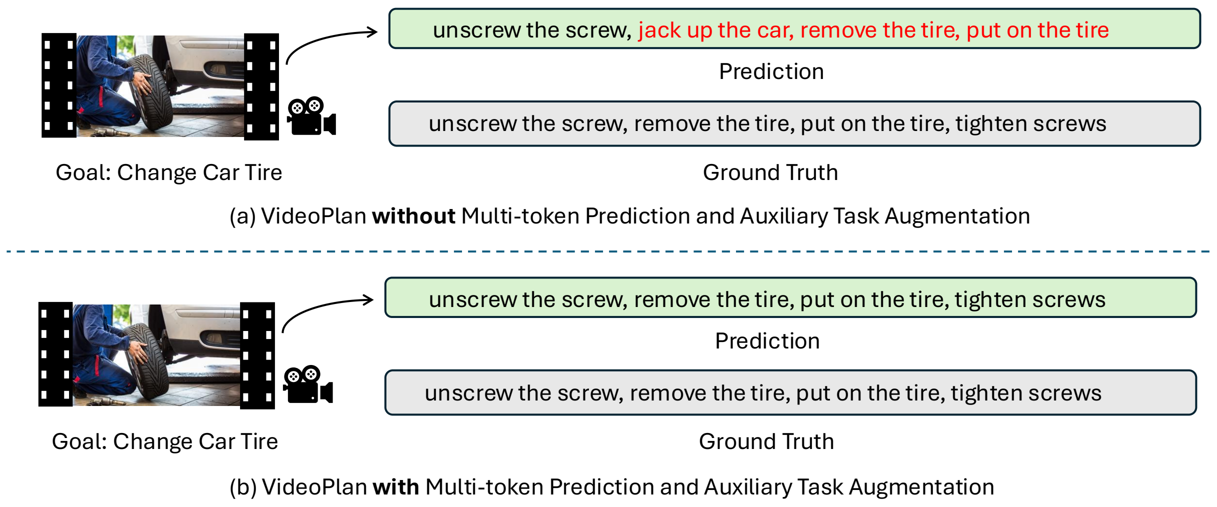
VPA. We visualize success cases and failure cases of our method in Figure 1. The predictions in the figure are raw outputs from our method with little post-processing. Although our method generates free-from text as outputs, the raw outputs still make valid action names. From the figure we can also observe that the dataset annotations contain repetitive actions. In most cases, the repetitive actions are hard to predict. Even though our model predicts plausible actions without repetitive actions, it is still treated as failed. In Figure 2 we explore the effects of Auxiliary Task Augmentation (ATA) and Multi-token Prediction (MTP). From the figure we can observe that our method without ATA and MTP predicts the second action incorrectly while our method with ATA and MTP predict all steps correctly.

LTA. Figure 3 shows two examples from the Ego4D LTA task. From the figure we can see that our model is able to produce reasonable action sequences. Additionally, the model predicts "dough", "container" while the person is not doing cooking-related tasks. This indicates that our model's perception ability still has room for improvements. Finally, we can observe that verb prediction is more accurate than noun prediction. This shows the strong planning ability of our method.



take out the shell, remove the old memory chip, install the new memory chip, fit on the shell
Prediction
take out the shell, remove the old memory chip, install the new memory chip, fit on the shell
Ground Truth
Goal: Replace Memory Chip
disinfect the injecting place, inject to the muscular, pull out the needle and press, inject to the muscular
Prediction
disinfect the injecting place, inject to the muscular, inject to the muscular, pull out the needle and press
Ground Truth
Goal: Change Car Tire

Figure 1. **Success Cases and Failure Cases from COIN Dataset.** The red text denotes wrong predictions. The blue text denotes repetitive action annotations in the dataset. **Top**: One success case of our method. Our model correctly predicts all future actions. **Bottom**: One failure case of our method. The ground truth annotations contain repetitive actions "inject to the muscular". Our method only predicts one "inject to the muscular". Therefore, it begins to be incorrect from the third future action.



unscrew the screw, jack up the car, remove the tire, put on the tire
Prediction
unscrew the screw, remove the tire, put on the tire, tighten screws
Ground Truth
Goal: Change Car Tire
(a) VideoPlan **without** Multi-token Prediction and Auxiliary Task Augmentation
unscrew the screw, remove the tire, put on the tire, tighten screws
Prediction
unscrew the screw, remove the tire, put on the tire, tighten screws
Ground Truth
Goal: Change Car Tire
(b) VideoPlan **with** Multi-token Prediction and Auxiliary Task Augmentation

Figure 2. **Effects of Auxiliary Task Augmentation (ATA) and Multi-token Prediction (MTP).** The red text denotes wrong predictions. **Top**: Our method **without** ATA and MTP. The model mistakenly outputs the action "jack up the car" when predicting the second action. Even though the third and the fourth predicted actions (i.e. "remove the tire", "put on the tire") match the second and the third ground truth actions, the task is still treated as failed. **Bottom**: Our method **with** ATA and MTP. Our model correctly predicts all steps.

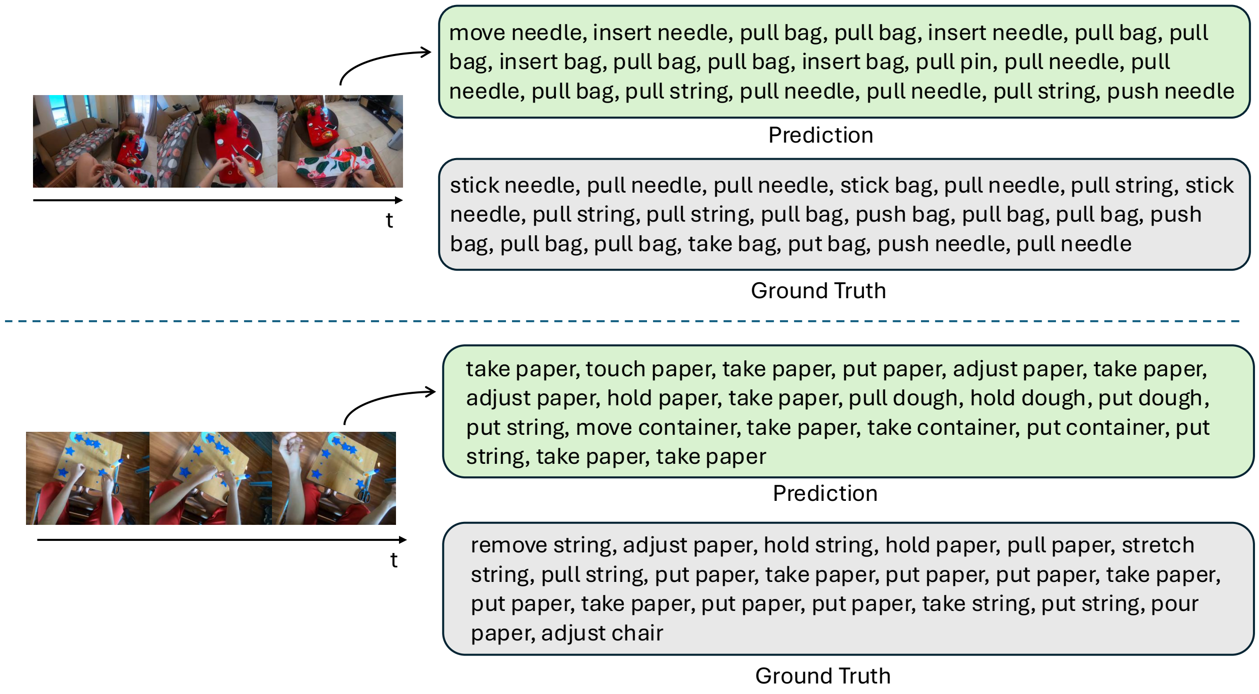
References

[1] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1

[2] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022. 2

[3] Md Mohaiminul Islam, Tushar Nagarajan, Huiyu Wang, Fu-Jen Chu, Kris Kitani, Gedas Bertasius, and Xitong Yang. Propose, assess, search: Harnessing llms for goal-oriented planning in instructional videos. *arXiv preprint arXiv:2409.20557*, 2024.

[4] Jiateng Liu, Sha Li, Zhenhailong Wang, Manling Li, and Heng Ji. A language-first approach for procedure planning. In *Find-*



move needle, insert needle, pull bag, pull bag, insert needle, pull bag, pull bag, insert bag, pull bag, pull bag, insert bag, pull pin, pull needle, pull needle, pull bag, pull string, pull needle, pull needle, pull string, push needle

stick needle, pull needle, pull needle, stick bag, pull needle, pull string, stick needle, pull string, pull string, pull bag, push bag, pull bag, pull bag, push bag, pull bag, pull bag, take bag, put bag, push needle, pull needle

take paper, touch paper, take paper, put paper, adjust paper, take paper, adjust paper, hold paper, take paper, pull dough, hold dough, put dough, put string, move container, take paper, take container, put container, put string, take paper, take paper

remove string, adjust paper, hold string, hold paper, pull paper, stretch string, pull string, put paper, take paper, put paper, put paper, take paper, put paper, take paper, put paper, put paper, take string, put string, pour paper, adjust chair

Figure 3. **Qualitative Results for Ego4D LTA.** Predicting long-term future actions are extremely challenging because the future is uncertain and there are multiple possible future action sequences. The action sequences produced by our method generally matches the person's goal and behavior. In the bottom subfigure, the model predicts "dough", "container" while the person is not in a cooking scenario. This suggests that perception ability of our model still has room for improvements.

ings of the Association for Computational Linguistics: ACL 2023, pages 1941–1954, 2023. 2

[5] Kaleb Newman, Shijie Wang, Yuan Zang, David Heffren, and Chen Sun. Do pre-trained vision-language models encode object states? *arXiv preprint arXiv:2409.10488*, 2024. 1

[6] Yulei Niu, Wenliang Guo, Long Chen, Xudong Lin, and Shih-Fu Chang. Schema: State changes matter for procedure planning in instructional videos. *arXiv preprint arXiv:2403.01599*, 2024. 1

[7] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. 2

[8] Masatoshi Tateno, Takuma Yagi, Ryosuke Furuta, and Yoichi Sato. Learning object states from actions via large language models. *arXiv preprint arXiv:2405.01090*, 2024. 1

[9] Zihui Xue, Kumar Ashutosh, and Kristen Grauman. Learning object state changes in videos: An open-world perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18493–18503, 2024. 1