DPO-Finetuned Large Multi-Modal Planner with Retrieval-Augmented Generation @ EgoPlan Challenge ICML 2024

Kwanghyeon Lee [1] **Mina Kang** [1] **Hyungho Na** [1] **Heesun Bae** [1] **Byeonghu Na** [1] **Doyun Kwon** [1] **Seungjae Shin** [1] **Yeongmin Kim** [1] **Taewoo Kim** [1] **Seungmin Yun** [1] **Il-Chul Moon** [1 2]

Abstract

This paper presents technical details for solving a multi-modal task, EgoPlan-Bench. Our model adopts Direct Preference Optimization (DPO), which is originally developed for a single-modal task, to be utilized in a multi-modal setting. This DPO adaptation improves prediction accuracy by highlighting positive answers over negative choices. Additionally, we apply Retrieval-Augmented Generation (RAG) to further enhance generation performance in Multi-modal Large Language Models (MLLMs). However, in our settings, the RAG method does not result in a performance improvement due to the limited retrieval of similar tasks. Our model utilizing DPO shows performance improvements and achieves 53.98% test accuracy compared to baseline methods of 41.35%. Our code is available at https://github.com/aailabkaist/EgoPlan_Challenge_Team_AAILab.

1. Introduction

From the wide utilization of Multi-modal Large Language Models (MLLMs), recent innovations focus on using MLLMs in Embodied AI (EAI), i.e. generating action plans (Mu et al., 2024). EAI requires a comprehensive understanding of its designated tasks with fine-grained information via visual and textual inputs. For instance, EgoPlan-Bench (Chen et al., 2023) aims to predict the most probable action among multiple choices given information of previous actions, current state images, and task descriptions.

This paper reports our innovations in developing a task planner for EgoPlan-Bench. Specifically, we adopt Direct Preference Optimization (DPO) (Rafailov et al., 2024) and Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) to improve the performance of MLLMs. In our experiments, there is an evident performance improvement with DPO but no further improvement with RAG.

[1]KAIST, Daejeon, Republic of Korea [2]summary.ai, Daejeon, Republic of Korea. Correspondence to: Il-Chul Moon <icmoon@kaist.ac.kr>, Kwanghyeon Lee <rhkdgus0414@kaist.ac.kr>.

Multi-modal Foundation Model meets Embodied AI Workshop in the 41^{st} International Conference on Machine Learning, Vienna, Austria, 2024.

2. Preliminaries

RLHF Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017) often aligns Large Language Models (LLMs) to follow human preference. Typically, RLHF with Supervised Fine-Tuning (SFT) requires 1) training by a reward model to reflect human preferences and 2) fine-tuning LLMs using reinforcement learning to maximize the estimated reward. This procedure suggests that there can be higher variance if the human preference changes by periods or by individuals. With a pre-trained SFT policy function, π_{SFT}; we construct static preference dataset $\mathcal{D}_P = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$ where y_w and y_l denote preferred and dis-preferred responses corresponding to given prompts x, respectively. Then, we train a reward function, $r_\phi(x, y)$, by the guidance from $\mathcal{D}_P$.

$$\mathcal{L}_R(\phi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_P}[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))] \tag{1}$$

where σ is the logistic function. According to the Bradley-Terry model (Bradley & Terry, 1952), we enforce the policy model to further align with human preferences via reinforcement learning with the below loss function.

$$\begin{aligned}\mathcal{L}_{RL}(\theta) = &-\mathbb{E}_{x\sim\mathcal{D}_P, y\sim\pi_\theta(y|x)}[r_\phi(x, y)] \\ &+ \beta\mathbb{D}_{\text{KL}}[\pi_\theta(y|x)||\pi_{ref}(y|x)]\end{aligned} \tag{2}$$

Here, π_{ref} is a reference model used for updating policy model π_θ; β is a scale factor controlling the deviation from the reference model; $y \in \mathcal{Y}$ is the entire domain dataset of y_w and y_l. In the case of EgoPlan-Bench, multi-modal input x includes both textual input $t = (t_g, t_s)$ and visual input $v = (v_p, v_o)$[1]. The textual input t_g and t_s[2] correspond to

[1]This paper will distinguish the bold symbol as a sequence of such symbol elements. For instance, t will be a text input at a certain timestep, and $\mathbf{t}$ is a sequence of such text inputs.

[2]Here, t will be a token if it is used in the embedding procedure by transformers, and it will be a text sentence if used otherwise.

tokenized task goal g and supplemental text s. Visual input v_p and v_o correspond to visual tokens of given previous video frames and current observation, respectively.

RAG Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) is a technique in natural language processing that enhances the generation performance of language models by incorporating a retrieval mechanism. RAG dynamically retrieves relevant information from external databases or corpus over the answer generation process. This approach improves the accuracy of generated responses, particularly when the retrieved information reveals new information that can validate a generated answer. We assume that RAG is applicable because there are similar situations in evaluation datasets; and because pseudo-textual directions can be generated on those evaluation instances. This technique is applicable in both training and testing stages, so the previously mentioned fine-tuning can benefit from RAG, as well.

3. Methodology

Figure 1 illustrates the overview of the proposed model adopting both DPO and RAG.

3.1. DPO for MLLMs

Let's say that a given Supervised Fine-Tuning (SFT) model has a reference policy π_{ref} and preference data $\mathcal{D}_P$. Direct Preference Optimization (DPO) (Rafailov et al., 2024) presents a new parameterization of π_{ref} to solve the RLHF problem with only a simple classification loss, without explicit reward modeling. Following the prior works (Peters & Schaal, 2007; Korbak et al., 2022; Go et al., 2023), DPO derives the closed-form optimization solution, Eq. (3), to minimize the KL-constrained objective in Eq. (2).

$$\pi_r(y|x) = \frac{1}{Z(x)}\pi_{ref}(y|x)\exp(\frac{1}{\beta}r(x,y)) \tag{3}$$

Here, $Z(x) = \sum_y \pi_{ref}(y|x)\exp(\frac{1}{\beta}r(x,y))$ is the partition function. Afterward, we hypothesize an optimal policy function π_r, so we derive a closed-form of reward function $r(x,y)$, Eq. (4), given implicitly from $\mathcal{D}_P$.

$$r(x,y) = \beta\log\frac{\pi_r(y|x)}{\pi_{ref}(y|x)} + \beta\log Z(x) \tag{4}$$

By substituting r_ϕ in Eq. (2) with Eq. (4), the alternative loss-minimization objective is derived without an explicit definition of the reward model as follows:

$$\mathcal{L}_{DPO}(\pi_\theta;\pi_{ref},\mathcal{D}_P) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_P} \\ [\log\sigma(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)})] \tag{5}$$

To train MLLMs with Eq. (5), the static preference dataset $\mathcal{D}_P$ is required. To this end, we utilize EgoPlan-Bench (Chen et al., 2023) as a training dataset, which includes a golden answer y_w and numbers of negative answers y_l to a given input x described in Section 2. In this technical report, a policy π_θ refers to a classifier from Video-LLaMA (Zhang et al., 2023) which makes a prediction on y by computing the product of the predicted probabilities of the tokens composing response y. To adopt DPO, Eq. (5) requires a reference policy function, π_{ref}. For this, we duplicate additional π_θ and set it as π_{ref}. Note that π_{ref} is frozen throughout the training. The details of training and related loss functions will be discussed in Appendix C.

Iterative DPO Fine-tuning In the original DPO, π_{SFT} is used as π_{ref}, so there is a clear supervision on preferences. In this challenge, π_{SFT} would be the finetuned model of EgoPlan-Bench. However, we choose to iteratively finetune π_{ref} from original Video-LLaMA (Zhang et al., 2023), not from EgoPlan-Bench. The rationale is the consistency over the MLLM fine-tuning process since the fine-tuning loss of EgoPlan-Bench is not consistent with DPO Loss of Eq. (5). In detail, we start from π_{ref} and π_{ref}, which comes from the original Video-LLaMA. Then, we finetune π_θ with $\mathcal{L}_{DPO}$ under $\mathcal{D}_P$. After the loss satuation, we update π_{ref} with π_θ. This procedure would be a self-training concept applied to DPO.

3.2. Retrieval-Augmented Generation

Action Database Construction The action database is a collection of possible action sequences. Given a query asking current situational contexts, RAG defines how to find such similar instances relevant to the query. Hence, we establish an action database, denoted as $\mathcal{B}$, constructed by EgoPlan-IT (Chen et al., 2023) and an additional generated dataset from Ego4D videos.

Each instance in the action database consists of a pair $(q, \mathbf{t}_a) \in \mathcal{B}$, where q is query, $\mathbf{t}_a := \{t_{a,h}\}_{h=1}^H$ is the action sequence, $t_{a,h}$ is the h-th action, and H is the number of actions. We construct two types of databases depending on the type of query: $\mathcal{B}^{\text{goal}}$ where q is task goal g and $\mathcal{B}^{\text{ans}}$ where q is a pair of answer and negative answer (y_w, y_l).

We first include EgoPlan-IT in the action database. This dataset comprises task goal g, answer y_w, negative answer y_l, and action sequence $\mathbf{t}_a$. Therefore, we construct the Egoplan-IT action database $\mathcal{B}^{\text{goal}}_{\text{egoplan}} = \{(g^{(i)}, \mathbf{t}_a^{(i)})\}_{i=1}^M$ and $\mathcal{B}^{\text{ans}}_{\text{egoplan}} = \{((y_w^{(i)}, y_l^{(i)}), \mathbf{t}_a^{(i)})\}_{i=1}^M$ where M is the number of instances in EgoPlan-IT.

Additionally, we generate new instances from the Ego4D videos, whose scenarios are cooking without video IDs that appear in validation and test data. We follow the dataset generation process of EgoPlan-Bench (Chen et al., 2023). Specifically, we utilize the GPT-4o (Achiam et al.,

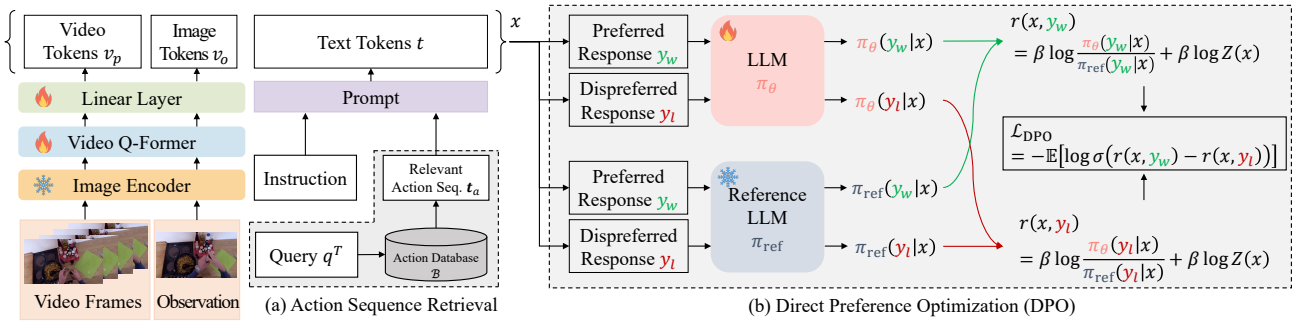

Figure 1: Overview of our framework. We primarily adopt the same network structure that is used to finetune Video-LLaMA (Zhang et al., 2023) in EgoPlan-Bench (Chen et al., 2023). Our framework consists of two distinctive components: (a) action sequence retrieval and (b) direct preference optimization (DPO). In (a), a query from the data instance is sent to the action database to extract the relevant action sequence, which is then added to the prompt. In (b), the preferred response y_w and the dispreferred response y_l are propagated with multimodal tokens x through both the target LLM and the reference LLM. Then, we evaluate the rewards $r(x, y_w)$ and $r(x, y_l)$ for each response and compute the loss $\mathcal{L}_{DPO}$ where σ is the logistic function.

2023) to generate the task goals and their corresponding time interval. For video j, we utilize the set of actions in the video and its corresponding timestamp, $d^{(j)}$, and task goal generation prompt, P, from EgoPlan-Bench. We generate task goals, $g_k^{(j)}$, and an associated time interval, $T_k^{(j)}$, using GPT-4o. This process is expressed as $\{(g_k^{(j)}, T_k^{(j)})\}_{k=1}^{N^{(j)}} = f_{\text{gpt}}(d^{(j)}, P)$ where f_{gpt} represents the GPT-4o; and $N^{(j)}$ is the number of task goals for video j.

Using the GPT outputs, we construct the generated Ego4D dataset, which is detailed in Appendix A. Then, similar to the EgoPlan-IT action database, we construct the generated Ego4D action database: $\mathcal{B}^{\text{goal}}_{\text{ego4d}} = \{(\tilde{g}^{(i)}, \tilde{\mathbf{t}}_a^{(i)})\}_{i=1}^{\tilde{M}}$ and $\mathcal{B}^{\text{ans}}_{\text{ego4d}} = \{((\tilde{y}_w^{(i)}, \tilde{y}_l^{(i)}), \tilde{\mathbf{t}}_a^{(i)})\}_{i=1}^{\tilde{M}}$ where $\tilde{M}$ is the number of instances in generated Ego4D dataset.

Finally, we combine the Egoplan-IT database and generated Ego4D database to form the final database: $\mathcal{B}^{\text{goal}} = \mathcal{B}^{\text{goal}}_{\text{egoplan}} \cup \mathcal{B}^{\text{goal}}_{\text{ego4d}}$ and $\mathcal{B}^{\text{ans}} = \mathcal{B}^{\text{ans}}_{\text{egoplan}} \cup \mathcal{B}^{\text{ans}}_{\text{ego4d}}$.

Retrieval Methods We apply different retrieval methods depending on the type of database. For goal action database $\mathcal{B}^{\text{goal}}$, we focus on the task goal g, and we measure similarity based on the co-occurrence of object and verb within task goal. We leverage the pos-tagging function provided by the Natural Language Toolkit (NLTK) (Bird et al., 2009) to isolate object and action entities from task goal texts. Subsequently, these entities are vectorized using NLTK CountVectorizer to facilitate the generation of co-occurrence vectors for each instance. We define the resulting vector as query embedding $\mathbf{q}$.

For $\mathcal{B}^{\text{ans}}$, the approach involves assessing the similarity between each answer, denoted as y, using BERT embeddings (Devlin et al., 2018). For including only action-relevant information, all prepositional phrases, except for the verb-object structure, are removed. The result embedding vectors are extracted using BERT and we define it as query embedding $\mathbf{q}$.

We measure the cosine similarity as $\cos(\mathbf{q}^T, \mathbf{q}_i)$ for all $i \in \{1, ..., M + \tilde{M}\}$ to identify the retrieval instance, where $\mathbf{q}^T$ is the target query and $\mathbf{q}_i$ represents the query from the action database. We utilize the retrieved action sequence $\mathbf{t}_a$ as the input prompt.

To filter out potentially irrelevant retrieved action sequences, we set a specific threshold, τ, on the cosine similarity. By including only the retrieval instances with a similarity greater than τ, we assume that this filtering will help in obtaining only informative sequences while ignoring noisy information. The resulting $\mathbf{t}_a$ is included in t_s.

4. Experiment

4.1. Experiment Setting

We use the following settings for model training. The model training and evaluation are implemented based on the Video-LLaMA (Zhang et al., 2023) and EgoPlan-Bench (Chen et al., 2023) code provided by the organizers. For training, we use 8 A100 GPUs and a batch size of 16.

For RAG training, we use $\mathcal{B}^{\text{goal}}$ with a threshold of 0.96 to retrieve the longest action sequence, covering 24,148 instances. In the validation dataset, from $\mathcal{B}^{\text{ans}}$ with a threshold of 0.95, we retrieve 111 out of 923 instances.

Furthermore, as the test dataset and a portion of the validation dataset consist of the Ego4D dataset, we evaluate the model performance based on the Ego4D validation dataset, which has 923 instances, to ensure the accuracy of our test

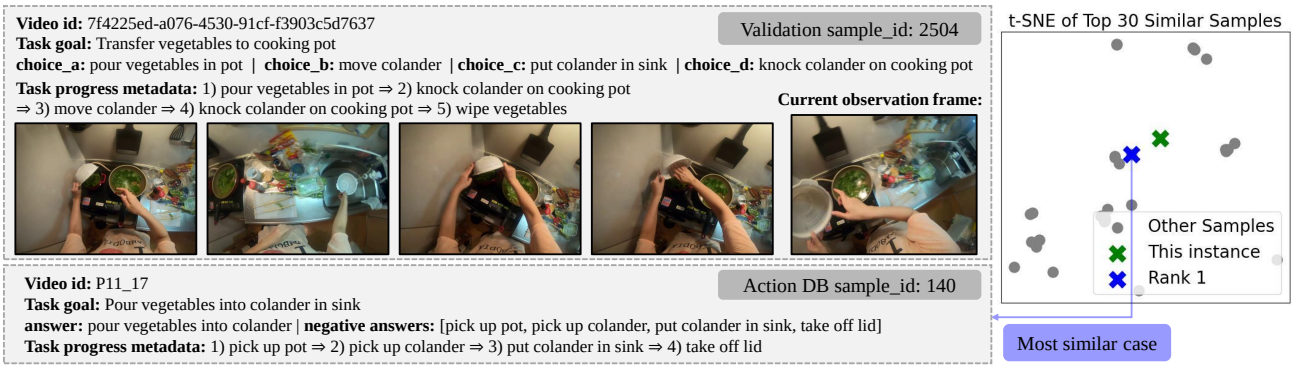

Figure 2: Qualitative analysis on Retrieval-Augmented Generation.

Table 1: Test accuracy with regard to our method component DPO loss presented in Eq. (5).

Model	Base	Loss Type	Test Acc.(%)
Baseline	Original	Contrastive	41.35
Ours	Original	DPO	**53.98**

Table 2: Validation accuracies for various combinations of our method components. **Base** indicates the initial checkpoint from which the model is fine-tuned. Results marked with † are from Chen et al. (2023).

Model	Base	Loss Type	RAG	Valid Acc.(%) / Approx. Training Time
Baseline	Original	–	–	30.44† / Given Pre-trained Model
		Contrastive	✗	44.42† / Given Pre-trained Model
Ours	Original	DPO	✗	60.24 / 0.5 days
	DPO-Finetuned	Contrastive (Iterative)	✓	46.05 / 0.5 days
		DPO (Iterative)	✗	**61.11** / 0.5 days
		DPO (Iterative)	✓	60.24 / 0.5 days

results.

4.2. Quantitative Analysis

Our best model utilizes DPO, starting from a base model that is finetuned by the DPO framework prior. Table 1 indicates that the best case achieves a 12.6% increment in test accuracy compared to the official finetuned model.

Table 2 identifies that DPO fine-tuning achieved higher performance than the official finetuned model trained on the same training dataset via the original contrastive loss. This observation suggests that DPO loss of Eq. (5) improves performance by training MLLMs to choose preferred answer y_w more, compared to the reference model of π_{SFT} in EgoPlan-Bench (Chen et al., 2023). In addition, Table 2 shows the advantage of iterative DPO fine-tuning, which has been observed in previous works (Gorbatovski et al., 2024).

The current version of RAG results in a performance decrease compared to the DPO-finetuned case. Thus, we conduct qualitative analysis to understand the limitations of the current RAG method.

4.3. Qualitative Analysis

The right side of Figure 2 shows the t-SNE (Van der Maaten & Hinton, 2008) of the BERT embeddings from 1) candidate choices of validation instance no. 2504 and those from 2) concatenation of answer and negative answers of the top 30 instances from $\mathcal{B}^{\text{ans}}$ whose BERT embeddings show the highest cosine similarity to validation instance.

From the left side of Figure 2, we observe that answer candidates of 1) and 2) are actually similar when they are close to each other in the BERT embedding space. However, since the task goals of the two instances are different, the action sequence of the retrieved instance may not be helpful for predicting the progress of the current validation instance.

As previously mentioned, the task goal is not considered in the RAG process during testing. From here, we notice that it is necessary to supplement our retrieval method by reflecting factors that help explain the instance, such as task goal. An additional failure case and its potential solution are discussed in Appendix D.

5. Conclusions

We propose the method to adopt Direct Preference Optimization (DPO) and Retrieval-Augmented Generation (RAG) for fine-tuning MLLMs on the scenarios from EgoPlan-Bench. We utilize RAG to search for action narrations of similar situations for a given question instance. However, RAG in our settings does not show any performance improvement, which we attribute to the limited retrieval of similar tasks from the action database. On the other hand, utilizing DPO achieves test performance surpassing the existing baseline.

Acknowledgements

This research was supported by AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data (IITP) funded by the Ministry of Science and ICT (2022-0-00077).

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Bird, S., Klein, E., and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Chen, Y., Ge, Y., Ge, Y., Ding, M., Li, B., Wang, R., Xu, R., Shan, Y., and Liu, X. Egoplan-bench: Benchmarking egocentric embodied planning with multimodal large language models. *arXiv preprint arXiv:2312.06722*, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Go, D., Korbak, T., Kruszewski, G., Rozen, J., Ryu, N., and Dymetman, M. Aligning language models with preferences through f-divergence minimization. *arXiv preprint arXiv:2302.08215*, 2023.

Gorbatovski, A., Shaposhnikov, B., Malakhov, A., Surnachev, N., Aksenov, Y., Maksimov, I., Balagansky, N., and Gavrilov, D. Learn your reference model for real good alignment. *arXiv preprint arXiv:2404.09656*, 2024.

Korbak, T., Elsahar, H., Kruszewski, G., and Dymetman, M. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *Advances in Neural Information Processing Systems*, 35:16203–16220, 2022.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Li, K., He, Y., Wang, Y., Li, Y., Wang, W., Luo, P., Wang, Y., Wang, L., and Qiao, Y. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

Mu, Y., Zhang, Q., Hu, M., Wang, W., Ding, M., Jin, J., Wang, B., Dai, J., Qiao, Y., and Luo, P. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36, 2024.

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Zhang, H., Li, X., and Bing, L. Video-llama: An instruction-tuned audio-visual language model for video understanding, 2023. URL https://arxiv.org/abs/2306.02858.

Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

A. Construction of Ego4D-Generated Dataset for Action Database

A.1. Video ID Selection

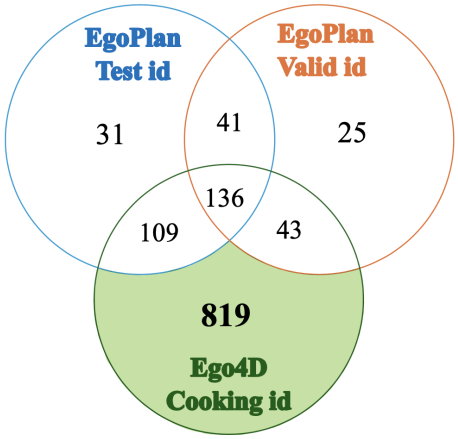

Figure 3: The EgoPlan Test id and Valid id circles show video IDs of the EgoPlan-IT bench's test and validation datasets. The Ego4D Cooking id circle includes cooking scenario IDs from Ego4D. Intersections represent overlapping video IDs between datasets, while non-overlapping parts are unique IDs. The green area shows the 819 cooking video IDs used to make the Ego4D-generated dataset, chosen to avoid overlap with test or validation datasets.

We identify 1107 video IDs from Ego4D where the scenario is exclusively cooking to generate new instances. As shown in Figure 3, we select 819 cooking IDs, represented by the green-shaded area, that do not overlap with video IDs appearing in the test or validation datasets. For constructing the Ego4D-generated dataset, we use 605 video IDs out of the 819, excluding 214 video IDs that lacked narration text from Ego4D. These 605 video IDs are then utilized for the construction of the action database.

A.2. Data Generation Using GPT-4o Task Goals

We provide an exhaustive explanation of the methodologies employed to extract, generate, and construct the dataset, primarily derived from the Ego4D videos.

We extract the corresponding action during $\tilde{T}_k^{(j)}$ as $\tilde{A}_k^{(j)} = \{\tilde{t}_{a,k,h}^{(j)}\}_{h=1}^{l_{a,k}^{(j)}}$, where $\tilde{t}_{a,k,h}^{(j)}$ is the h-th action of k-th task goal in the j-th video. Finally, we form pairs, $\{(\tilde{g}_k^{(j)}, \tilde{A}_k^{(j)})\}_{k=1}^{N^{(j)}}$, incorporating the identified task goals and their associated action text sequences. To augment the data pair for the different frames, we randomly sample $t \in (1, l_{a,k}^{(j)})$ for each goal-action pair, and set $\tilde{t}_{a,k,t}^{(j)}$ as answer $\tilde{y}_w$; and $\tilde{t}_{a,k,h\neq t}^{(j)}$ as negative answers $\tilde{y}_l$. Finally, the set of action texts $\tilde{\mathbf{t}}_a$ becomes $\tilde{\mathbf{t}}_{a,k}^{(j)} = \{\tilde{t}_{a,k,h}^{(j)}\}_{h=1}^{t}$.

Now, we construct the database for each video, and by combining these for all videos, we build the database for the generated Ego4D dataset:

$$\mathcal{B}_{\text{ego4d}}^{\text{goal}} = \cup_{j=1}^{\tilde{M}} \{(\tilde{g}_k^{(j)}, \tilde{\mathbf{t}}_{a,k}^{(j)})\}_{k=1}^{N^{(j)}}, \tag{6}$$

$$\mathcal{B}_{\text{ego4d}}^{\text{ans}} = \cup_{j=1}^{\tilde{M}} \{((\tilde{t}_{a,k,t}^{(j)}, \tilde{t}_{a,k,h\neq t}^{(j)}), \tilde{\mathbf{t}}_{a,k}^{(j)})\}_{k=1}^{N^{(j)}}. \tag{7}$$

B. Sensitivity Analysis

B.1. DPO hyper-parameter

Figure 4 shows the performance depends on DPO hyperparameter β. 0.05 and 0.1 show high performance, while 0.5 yields lower performance than other values. Although the peak performance of 0.05 and 0.1 were similar, 0.1 shows a slightly higher performance than 0.05. Therefore, we set the β for DPO training to 0.1.

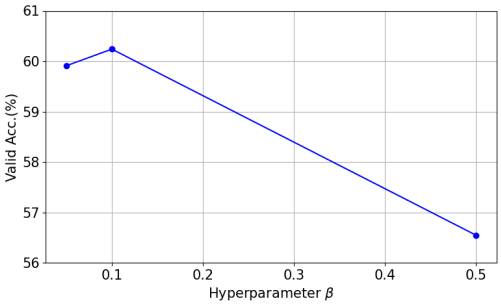

Figure 4: Graph of peak performance for each β. 0.1 shows the best peak performance among 0.05 and 0.5.

C. Training Details

We present the details of the training procedure for π_θ. To this end, we first define the dataset used during training. First, the EgoPlan-IT dataset denoted as $\mathcal{D}_P$ is used to fine-tune for EgoPlan-Bench (Chen et al., 2023). EgoPlan-IT-action dataset[3] is adopted for action recognition task, predicting previous actions from video data. We also use other datasets, such as MiniGPT-4 (3K) (Zhu et al., 2023), LLaVA (150K) (Liu et al., 2024), and VideoChat (11K) (Li et al., 2023), for further instruction tuning as in the original approach (Chen et al., 2023). For conciseness, we denote a dataset other than EgoPlan-IT as $\mathcal{D}_E$, which includes answers y for given inputs x. We utilize the different loss functions according to the dataset as follows.

$$\mathcal{L}_\theta(\mathcal{D}) = \begin{cases} \mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}, \mathcal{D}), & \text{if } \mathcal{D} = \mathcal{D}_P \\ -\mathbb{E}_{(x,y)\sim\mathcal{D}}[\log \pi_\theta(y|x)], & \text{if } \mathcal{D} = \mathcal{D}_E \end{cases} \tag{8}$$

D. Failure Case Analysis

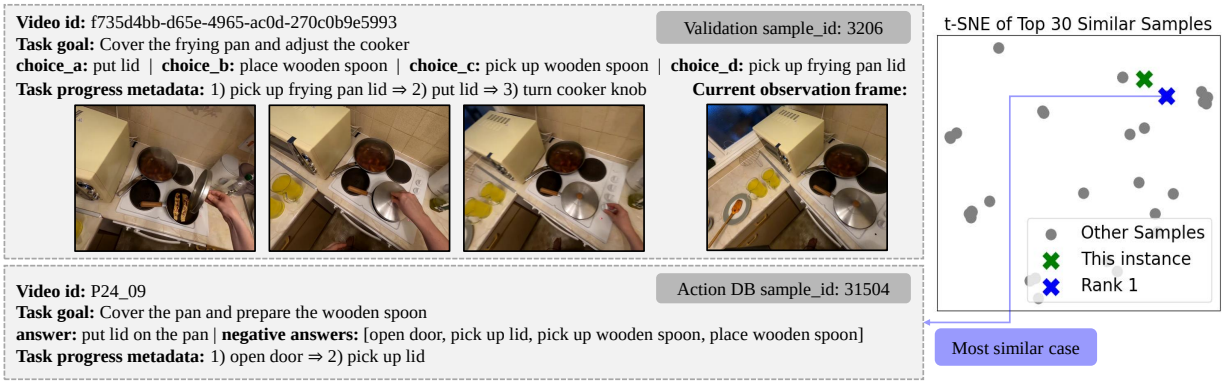

Figure 5: Qualitative analysis on Retrieval-Augmented Generation.

We show another failure retrieval case where answer candidates of 1) and 2) are actually similar and the task goals of the two instances are also similar. However, the retrieved action sequence has a shorter length than that of the current validation instance. In this case, since all tasks specified in the retrieved action sequence are already done in the current validation instance, the action sequence of the retrieved instance would not be helpful for predicting the next task of the current instance.

Because we construct an instance-based action database at this time, a fully specified action sequence for each task goal is not guaranteed. To handle this problem, considering the purpose of the RAG method, there need to be generalized expressions of task goals and a fully specified action sequence for each task goal.

[3]In this technical report, we separately define that EgoPlan-IT-action from EgoPlan-IT since its question prompt is changed.