# A PICTURE IS WORTH A THOUSAND WORDS: LANGUAGE MODELS PLAN FROM PIXELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Planning is an important capability of artificial agents that perform long-horizon tasks in real-world environments. In this work, we explore the use of pre-trained language models (PLMs) to reason about plan sequences from text instructions in embodied visual environments. Prior PLM based approaches for planning either assume observations are available in the form of text by a captioning model, reason about plans from the instruction alone, or incorporate information about the visual environment in limited ways (such as a pre-trained affordance function). In contrast, we show that the PLM can accurately plan even when observations are directly encoded as input prompts for the PLM. We show this simple approach outperforms prior approaches in experiments on the ALFWorld and VirtualHome benchmarks.

## 1 INTRODUCTION

The ability to reason about plans is critical for performing long-horizon tasks (Erol, 1996; Sohn et al., 2018; Sharma et al., 2022), compositional generalization (Corona et al., 2021) and generalization to unseen tasks and environments (Shridhar et al., 2020). Consider a simple scenario where a robot is tasked with preparing a meal and serving it on the table. This presents a non-trivial planning problem since the agent needs to understand the sequence of operations required to perform the task and search for the relevant objects in the unfamiliar environment by interacting with various objects.

Large language models have been recently shown to possess commonsense knowledge about the world such as object affordances and physical dynamics (Ouyang et al., 2022; Chowdhery et al., 2022). Early approaches considered text based environments and fine-tuned PLMs to predict actions given the history of past observations and actions (Jansen, 2020; Micheli & Fleuret, 2021; Yao et al., 2020). Recent work has used this ability to reason about plans from text instructions in simulated household environments by making simplifying assumptions such as text-only environment observations or feedback (Huang et al., 2022; Ahn et al., 2022; Li et al., 2022; Logeswaran et al., 2022).

We focus on *visually grounded planning* with PLMs — the ability to adapt plans based on interaction and visual feedback from the environment. While PLMs have strong planning commonsense priors, predictions from a PLM may not be directly realizable in the environment since the observation and action spaces are unknown. Adapting the PLM to predict executable plans requires *grounding* it in the environment, which is difficult as it is highly non-trivial to adapt PLMs to directly observe visual feedback. Some prior works assume the availability of a pre-trained affordance function (Ahn et al., 2022) or a success detector (Mirchandani et al., 2021). Notably, SayCan (Ahn et al., 2022) completely decouples the PLM from observation information by selecting actions that are both likely by the PLM and admissible (through a pre-trained affordance model). Although this helps grounding, the use of visual feedback for action admissibility alone is limited. Often an agent must choose one of many admissible actions using observations. For example, a driving agent should stop when encountering a stop sign, but both stopping and driving forward are admissible and indistinguishable to SayCan.

Another workaround explored in prior work is translating the information in the visual observations to text using a pre-trained captioning system (Shridhar et al., 2021; Huang et al., 2022). However, it can be difficult to faithfully describe an image in words and information is lost in this inherently noisy process, which limits the information available to the planner.

Recent work shows that PLMs can be adapted for various natural language tasks by inserting tunable embeddings or soft prompts at the input of the PLM (also called prompt tuning or prefix tuning) (Li

& Liang, 2021; Lester et al., 2021). This approach also extends to multi-modal understanding tasks such as image captioning (Mokady et al., 2021) and VQA (Tsimpoukelli et al., 2021) where visual embeddings computed are provided as input and trained for the PLM. On the other hand, transformer based architectures have been successfully applied to offline Reinforcement Learning in recent work (Chen et al., 2021; Janner et al., 2021; Li et al., 2022; Reid et al., 2022).

Taking inspiration from these works, we propose the simple approach of embedding visual observations ('visual prompts') and *directly inserting them as input* for the PLM. The visual encoder and PLM are jointly trained for the target task, an approach we call **Visual Prompt Planning**-(VP$^2$). This approach solves the issues discussed in prior workarounds by letting the PLM train to observations in an end to end manner without using external data (captions and admissibility). We show that this simple approach performs better than prior PLM-based planning approaches on two embodied planning benchmarks based on ALFWorld (Shridhar et al., 2021) and Virtualhome Puig et al. (2018).

## 2 APPROACH

**Problem Setting** We assume a goal-based MDP setting, parameterized by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, P, R_{\mathcal{G}})$: a state space $\mathcal{S}$, action space $\mathcal{A}$, a goal space $\mathcal{G}$, transition probabilities $P$, and reward $R_{\mathcal{G}}$. The planner is given $N$ expert demonstrations $\mathcal{D} = \{(g^{(i)}, o_0^{(i)}, a_0^{(i)}, o_1^{(i)}, a_1^{(i)}, \ldots, o_T^{(i)}, a_T^{(i)})\}_{i=1}^N$ where goals $g^{(i)} \in \mathcal{G}$ and actions $a^{(i)} \in \mathcal{A}$ are available as text and observations $o^{(i)} \in \mathbb{R}^{H \times W \times C}$ are images of size $H \times W \times C$. Further, we do not assume the list of possible actions available to the agent is known, or any pretrained admissibility or affordance function is known. Given goal description $g$, past actions $a_1, \ldots, a_{t-1}$ and observations $o_1, \ldots, o_t$, we seek to build a policy $\pi$ which models the next action probability $\pi(a_t | g, a_{1 \cdots t-1}, o_{1 \cdots t})$.

**Visual Prompt Planning** If goal description, actions and observations are available in the form of discrete token sequences, predicting the next action is similar to a language modeling task and a PLM can be fine-tuned for next action prediction: maximize $\log p_{\text{LM}}(a_t \mid \text{cxt}_t)$, where $\text{cxt}_t = \text{concat}\,(g, o_1, a_1, o_2, a_2, \ldots, a_{t-1}, o_t)$. However, observations may not be available in the form of text or discrete tokens in practice and we attempt to tackle this scenario.

As we discuss in Appendix A.1, PLMs are capable of processing a sequence of embeddings (which may not necessarily correspond to actual text tokens). The context can be re-written in terms of embedding sequences as $\text{cxt}_t = \text{concat}\left(g^e, o_1^e, a_1^e, o_2^e, a_2^e, \ldots, a_{t-1}^e, o_t^e\right)$ where $g^e, o_i^e, a_i^e$ respectively represent *embedding sequences* corresponding to the goal description, observations and actions. We assume that the goal description and actions are available in the form of text and $g^e, a_i^e$ can be obtained as the corresponding token embedding sequences.

**Observation Encoder** To obtain observation embeddings $o_i^e$, we propose to learn an observation encoder $f : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{m \times E}$. $f$ maps visual observations $o$ to a sequence of $m$ embeddings each of size $E$, where $m$ is a hyperparameter and $E$ is the PLM's embedding dimensionality. In our experiments we consider an observation encoder of the form $f(o) = f_{\text{FFN}}(f_{\text{pretrained}}(o))$ where $f_{\text{pretrained}}$ is a pre-trained visual encoder and $f_{\text{FFN}}$ is a feedforward network.

**Training Objective** We learn to model the next action given previous actions, observations, and the goal. Similar to prior approaches Micheli & Fleuret (2021); Huang et al. (2022), we define the loss as $\mathcal{L}_{\mathcal{D}} = -\frac{1}{N} \sum_{i,t} \log p_{\text{LM}}(a_t^{(i)} \mid \text{cxt}_t^{(i)})$ where $\text{cxt}_t^{(i)}$ is a concatenation of goal, action and observation embeddings as previously described.

## 3 EXPERIMENTS

### 3.1 ENVIRONMENTS

We experiment with embodied agent tasks that involve navigating and manipulating objects in a simulated household environment. The agent acts by feeding text commands to a low-level controller that executes various pretrained skills (such as *go to cabinet* or *take apple from cabinet*).

| Approach | VizALF | | VH | |
|---|---|---|---|---|
| | ID | OD | ID | OD |
| Ignore (Jansen, 2020) | 35.4 | 22.4 | 6.0 | 1.5 |
| Captions (Shridhar et al., 2021) | 53.2 | 20.3 | 7.0 | 2.5 |
| $\text{SayCan}_{\text{trained}}$ (Ahn et al., 2022) | 36.6 | 19.3 | 7.0 | 1.0 |
| $\text{SayCan}_{\text{oracle}}$ (Ahn et al., 2022) | 51.2 | 27.8 | 13.6 | 5.0 |
| $\text{VP}^2$ (ours) | **55.3** | **27.8** | **20.6** | **7.5** |

Table 1: Success rates of approaches on VizALF and VirtualHome (VH) benchmarks. We present the average success rate for in distribution (ID) and out of distribution (OD) tasks.

**VizALF** This environment is based on ALFWorld (Shridhar et al., 2021) and contains 6 types of tasks that are compositional and contain multiple subgoals that must be completed. In contrast to ALFWorld which is a purely text based environment, we consider the same set of tasks but with **only** visual observations from the AI2-Thor simulator (Kolve et al., 2017). We used the training and evaluation task split provided in ALFWorld which consists of 4620 training tasks, 187 in distribution evaluation tasks, and 192 out of distribution evaluation tasks. However, we found 64/187 and 52/190 of the ID and OD evaluation tasks respectively were impossible to complete, even for an oracle agent. So, we normalized the success rate of all agents by the oracle's success rate.

**VirtualHome** We experiment with tasks from LID (Li et al., 2022) which are based on the VirtualHome simulator (Puig et al., 2018). Each task is specified using a set of goal conditions that must hold at the end of the episode (e.g., There must be two pancakes in the fridge and the stove must be turned on). We use the in distibution and novel scene splits from LID. 2000 in distribution tasks were used for training and 200 novel scene tasks were used for evaluation.

### 3.2 Models

We use the $\text{GPT2}_{\text{med}}$ language model (355M parameters) in all our experiments. We consider the following baselines for comparison.

- **Ignore** A simple baseline inspired by Jansen (2020) that ignores the visual observations and predicts the entire text action sequence only from the goal text description. This baseline is finetuned with the same objective as $\text{VP}^2$, $\mathcal{L}_{\mathcal{D}}$, but with observations removed from the context.
- **Captions** Instead of feeding visual observations to the planner language model, we use text captions predicted by a captioning model as a proxy (Shridhar et al., 2021). We train a ClipCap (Mokady et al., 2021) model on ground-truth captions from the respective environment's training demonstrations and use them for captioning. The captioning model is trained on 70k and 60k captions on VizALF and VirtualHome respectively.
- **SayCan** We implement two versions of the SayCan (Ahn et al., 2022) architecture. 1) An oracle version which uses a perfect affordance function. 2) We train an affordance function using the training demonstrations with annotated affordances, where a PLM is given a visual prompt and an action and must predict affordance. In both versions, we finetune a PLM to predict action sequences from the goal text, then select an action based on actions that have the highest SayCan score: affordance score times PLM score. To adapt to our setting, our implementation of SayCan deviates from the original implementation which uses a pre-trained LM without any fine-tuning (details in the appendix).

### 3.3 Results

Table 1 compares the performance of our method against baselines. Our simple approach ($\text{VP}^2$) performs better than all baselines, despite not using external data (caption and affordance information). $\text{VP}^2$ benefits from direct coupling between the planner language model and environment observations.

The Ignore baseline performs worse compared to other methods that make use of observations. However, on out of distribution tasks it suffers less from domain-shift compared to some of the other grounding baselines such as Captions. The Captions baseline performs better than Ignore, but suffers from information loss in the captioning process.

| Ablation | | VizALF | |
|---|---|---|---|
| | | ID | OD |
| | $VP^2$ | 55.3 | 27.8 |
| visual enc. | resnet50 | 30.1 | 12.9 |
| prompt | CLIPCap | 50.5 | 15.9 |
| base LM | $GPT2_{med}$ Frozen | 48.0 | 16.5 |
| | $GPT2_{XL}$ Frozen | 48.0 | 16.7 |
| | GPTJ Frozen | 50.0 | 15.8 |

Table 2: Select ablations on the VizALF benchmark, with average success rate for in distribution (ID) and out of distribution (OD) tasks. A full ablation table is included in the appendix (Table 4).

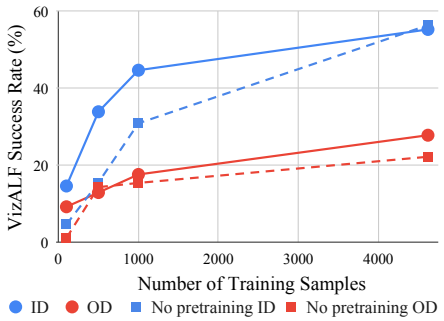


Figure 1: Ablation on VizALF when changing the number of training demonstrations and whether the base language model $GPT2_{med}$ is pretrained.

In comparison, SayCan with oracle affordance performs better than Ignore and Captions in spite of incorporating observation information only through the affordance function. SayCan using the trained affordance only performs slightly better than Ignore on in distribution, and similarly suffers from domain-shift on out of distribution tasks.

### 3.4 Ablations

Table 2 presents ablations we perform to identify the importance of each component of our approach.

**Visual Encoder** Replacing the CLIP visual encoder with a Resnet50 (He et al., 2016) significantly degrades the performance. This suggests that the image-text alignment pre-training of CLIP helps produce observation features that are more easily interpreted by the language model. In contrast to prior methods that consider auxiliary alignment objectives to match the distribution of inputs (Reid et al., 2022), it could be more beneficial to use powerful encoders such as CLIP pre-trained for image-text alignment.

**Pretrained Prompt Model** We also test how using pre-trained visual prompts can affect performance. We used visual prompts pre-trained with the CLIPCap captioning objective on the Conceptual Captions dataset (Sharma et al., 2018). However, using this pre-trained visual prompt decreased the success rate. We hypothesize that this decrease is due to the LM over-fitting to predict captions, and loses capacity to predict actions.

**Prompt Tuning on frozen language models** Finally, we consider an ablation where the language model backbone is held fixed and only visual prompts and task prompt tokens are tuned (similar to Frozen (Tsimpoukelli et al., 2021)). We found that freezing the language model generally performs worse than fine-tuning it. The drop in performance can be attributed to the limited influence of prompt tokens alone in controlling the language model's behavior.

**Effect of LM pre-training** We examine the benefit of LM pre-training by training a model from scratch on varying amounts of training data in Figure 1. We find that the pre-trained model converges faster and is more sample efficient compared to the model trained from scratch. This confirms findings from prior work about how language model pre-training benefits learning sequential decision making tasks (Reid et al., 2022).

## 4 Conclusion

We present a simple approach for planning from pixels building on the planning commonsense knowledge acquired by large language models. Compared to prior work, which indirectly incorporates observation information by captions or affordance, our approach is simpler, does not use external data, and benefits from directly coupling the planner language model and environment observations. Experimentally, we showed our approach performs better than these prior methods on two embodied agent benchmarks.

## REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1, 3, 7

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 2

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022. 1

Rodolfo Corona, Daniel Fried, Coline Devin, Dan Klein, and Trevor Darrell. Modular networks for compositional instruction following. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1033–1040, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.81. URL https://aclanthology.org/2021.naacl-main.81. 1

Kutluhan Erol. *Hierarchical task network planning: formalization, analysis, and implementation*. PhD thesis, 1996. 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*, pp. 770–778, 2016. 4

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022. 1, 2

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021. 2

Peter Jansen. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4412–4417, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.395. URL https://aclanthology.org/2020.findings-emnlp.395. 1, 3

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 3

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243. 2

Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022. 1, 2, 3, 8

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353. 1

Lajanugen Logeswaran, Yao Fu, Moontae Lee, and Honglak Lee. Few-shot subgoal planning with language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5493–5506, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.402. URL https://aclanthology.org/2022.naacl-main.402. 1

Vincent Micheli and Francois Fleuret. Language models are few-shot butlers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9312–9318, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.734. URL https://aclanthology.org/2021.emnlp-main.734. 1, 2

Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. Ella: Exploration through learned language abstraction. *Advances in Neural Information Processing Systems*, 34:29529–29540, 2021. 1

Ron Mokady, Amir Hertz, and Amit H Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021. 2, 3, 9

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. 1

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018. 2, 3

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022. 2, 4

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2556–2565, 2018. 4

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1713–1726, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.120. URL https://aclanthology.org/2022.acl-long.120. 1

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020. 1

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *ICLR*, 2021. 1, 2, 3, 7, 8

Sungryull Sohn, Junhyuk Oh, and Honglak Lee. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. *Advances in Neural Information Processing Systems*, 31, 2018. 1

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021. 2, 4

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 8

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8736–8754, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.704. URL https://aclanthology.org/2020.emnlp-main.704. 1

# A APPENDIX

## A.1 PRELIMINARY: PROMPT TUNING

Given a sequence of tokens $x_1, \ldots, x_t$, an auto-regressive language model predicts a probability distribution over the next token $p_{\text{LM}}(\cdot|x_1, \ldots, x_t)$. While a PLM expects to see natural language tokens in its context, the model can be extended to process a sequence of input embeddings instead. The input layer of a PLM converts tokens $x_1, \ldots, x_t$ into token embeddings $e_1, \ldots, e_t$ which are passed on to subsequent layers. Soft prompt tuning introduces additional tunable embeddings $p_1, \ldots, p_k$ in the input layer $p_{\text{LM}}(\cdot|e_1, \ldots, e_t, p_1, \ldots, p_k)$[1] which can be optimized with respect to a target training objective using gradient descent. In addition, for cross modal reasoning tasks the soft embeddings $p_i$ can be a function of data-modalities other than text such as images (e.g., $p_i = f_i(v; \theta)$ where $v$ is an image and $\theta$ are trainable parameters). We use this approach to augment the language model context with visual observations.

## A.2 EXPERIMENTAL SETUP DETAILS

**Visual Observations, Text Actions** VizALF uses the same action set defined in ALFWorld with the following difference. ALFWorld is a text-based environment and it references multiple instances of an object type with object identifiers (cabinet 1, cabinet 2, etc.). These object identifiers are no longer meaningful in the visual setting as the grounding of cabinet 1 to the physical cabinet is unknown to the agent. We thus removed these numeric identifiers in the action and ground-truth text observation (*e.g.*, "cabinet 1" becomes "cabinet"). The set of valid actions and objects is defined in (Shridhar et al., 2021).

We use the same procedure for removing object numeric identifiers from VirtualHome.

**Ground Truth Captions** Ground truth captions are needed to train the captions baseline and needed for the captioning auxiliary task. In VizALF, we use the ground truth captions provided in ALFWorld (Shridhar et al., 2021), which are generated from visible objects and a defined template. We define ground truth captions for VirtualHome using a similar template on the environment's list of interactable objects for each observation.

**Ground Truth Admissibility** Ground truth labels for admissibility are needed to train the SayCan trained affordance model. As the trained affordance is a binary classifier, we create a training set of observation-action-admissible pairs. To do this, for each observation, we sampled all admissible actions, and a subset of un-admissible actions (as there are vastly more un-admissible actions in each state). We tested two methods for sampling un-admissible actions: 1) We used a hand-coded heuristic that selects un-admissible actions based on actions that are likely to be predicted by the LM but are not admissible. 2) We use actions that are predicted with high likelihood from the Ignore baseline but are not admissible. We did not find a significant difference in un-admissible examples on performance.

## A.3 SAYCAN BASELINE DETAILS

In the original SayCan (Ahn et al., 2022), the PLM gives a score for every possible action at each timestep. However, in our setting, as each step has *thousands* of possible actions (actions are a combinatorial product on action type and objects), evaluating the LM on each action is infeasible. Instead, we use a beam search on the PLM to generate the top-$k$ most likely actions, rank these likely actions using the SayCan score. Specifically, for a given observation $o$, predict the next action $a$ by:

1. Sample top-$k$ actions $a_1, \ldots, a_k$ from PLM.
2. Compute
   $\text{Score}_{\text{SayCan}}(a_i) = p_{\text{LM}}(a_i) p_{\text{affordance}}(a_i, o)$
3. Return $\text{argmax}(\text{Score}_{\text{SayCan}}(a))$

[1] In an abuse of notation, we will use $p_{\text{LM}}$ with token inputs or embedding inputs interchangeably.

In addition, the original SayCan does not perform any finetuning on the PLM. However, we found that SayCan was not able to succeed in any of our environment tasks without finetuning.

### A.4 Code Implementation Details

Code is attached in the anonymous submission form, and will be released publicly at a later date.

Our approaches and baselines were implemented using the Huggingface Transformers Python library Wolf et al. (2019). Code for the VizALF environment was modified from the ALFWorld codebase Shridhar et al. (2021). Code for the VirtualHome environment was modified from the LID codebase Li et al. (2022).

### A.5 Computational Budget

Experiments were conducted on a100 GPUs. Most approaches used a $\text{GPT2}_{\text{med}}$ as a base LM, totalling 400M parameters. Ablations with larger frozen models had 1.5B and 6B parameters in total.

### A.6 Hyperparameters

In general we used similar hyperparameters for the PLM and visual prompts across approaches, but adjusted the epochs to account for learning modalities and number of training samples.

| Hyperparameter | Value |
|---|---|
| Epochs | 50 |
| Batch Size | 8 |
| Seed | {0, 1} |
| Grad Accum. Steps | 1 |
| LM | $\text{GPT2}_{\text{med}}$ |
| LM learning rate | 5e-5 |
| LM weight decay | 1e-3 |
| Gradient clipping | None |
| Weight decay | 0.01 |
| VP learning rate | 1e-2 |
| VP size | 10 |
| VP visual encoder | CLIP ViT-B/32 |
| VP arch. | 2-hidden-layer MLP |
| Max context tokens | 1000 |

Table 3: Hyperparameters for $\text{VP}^2$ in VizALF and VirtualHome.

**$\text{VP}^2$** We show relevant hyperparameters for $\text{VP}^2$ in Table 3.

**Ignore Baseline** Use the same hyperparameters as Table 3 but remove visual prompt parameters.

**Caption Baseline** To train a captioning model, we use the same hyperparameters as Table 3 but set {epochs = 10}. We train a separate PLM for the action prediction model (which uses predicted captions as input) using the same hyperparameters as Table 3 but set {epochs = 20}. For the context of the captions, we add goal text to encourage the PLM to produce captions: "Your task is to: caption the following observation". We also tested finetuning a CLIPCap captioning model for this baseline, but found this decreased captioning performance.

**SayCan Baseline** For the action prediction model, we use the same hyperparameters as Table 3 but set {epochs = 20}. For the trained affordance model, we use the same hyperparameters as Table 3 but set {epochs = 2}. For the context of the trained affordance, we add goal text to encourage the PLM to predict affordance: "Your task is to: predict whether the following action is valid." The PLM must either output the token "valid" or "invalid", given the observations and action contexts.

### A.7 FULL ABLATIONS

| Ablation | | VizALF | |
|---|---|---|---|
| | | ID | OD |
| | $VP^2$ | 55.3 | 27.8 |
| visual enc. | resnet50 | 30.1 | 12.9 |
| prompt | CLIPCap | 50.5 | 15.9 |
| base LM | $GPT2_{med}$ Frozen | 48.0 | 16.5 |
| | $GPT2_{XL}$ Frozen | 48.0 | 16.7 |
| | GPTJ Frozen | 50.0 | 15.8 |
| prompt size | 1 | 48.8 | 14.9 |
| | 5 | 54.4 | 17.1 |
| | 20 | 54.8 | 20.0 |
| aux. task | inv-dyn. | 58.8 | 20.7 |
| | captions | 58.5 | 21.4 |
| | goal-pred. | 52.9 | 17.8 |
| samples | 100 | 14.6 | 9.2 |
| | 500 | 33.9 | 12.9 |
| | 1000 | 44.7 | 17.6 |
| samples, no pre-train | 100 | 4.6 | 1.1 |
| | 500 | 15.5 | 14.3 |
| | 1000 | 30.9 | 15.4 |
| | 4620 (all) | 56.4 | 22.2 |

Table 4: All ablations on the VizALF benchmark, with average success rate for in distribution (ID) and out of distribution (OD) tasks. We tested our approach $VP^2$ with various components added or removed.

We show the full ablations in Table 4. This is the same as Table 2 but with ablations for prompt size, training samples, and LM pretraining.

**Prompt Size** We tested visual prompt sizes $\{1, 5, 10, 20\}$, where 10 is the prompt size used in $VP^2$. Lowering the prompt size to 1 and 5 lowers the success rate in VizALF, as less visual information can be contained in each prompt. However, raising the prompt size to 20 may allow more information in a visual prompt, but also causes the LM's context to increase. This can harm the LM as the context needs to be trimmed to fit within the limited context window. From Table 4, this seems to harm the success rate.

### A.8 AUXILIARY TASKS

In $VP^2$, the LM is only trained on the action prediction loss function $\mathcal{L}_{\mathcal{D}}$. We hypothesized that auxiliary tasks that train visual prompts in additional ways can help ground and improve the visual prompts for planning. To do this, we trained the LM concurrently on $\mathcal{L}_{\mathcal{D}}$ and a loss derived for each auxiliary task. 1) Inverse Dynamics (inv-dyn. ). The LM must predict the action that is executed between two observations: $\mathcal{L}_{\text{inv-dyn}} = -\frac{1}{N}\sum_{i,t} \log p_{\text{LM}}(a_t^{(i)}|o_t^{(i)}, o_{t+1}^{(i)})$. 2) Captions. The LM must predict the ground truth caption text for each observation. This is the same training objective used in CLIPCap (Mokady et al., 2021): $\mathcal{L}_{\text{cap}} = -\frac{1}{N}\sum_{i,t} \log p_{\text{LM}}(\text{caption}_t^{(i)}|o_t^{(i)})$. 3) Goal Prediction (goal-pred). The LM must predict the goal text given action-observation context. $\mathcal{L}_{\text{goal-pred}} = -\frac{1}{N}\sum_{i} \log p_{\text{LM}}(g^{(i)}|o_1^{(i)}, a_1^{(i)}, \ldots, o_T^{(i)}, a_T^{(i)})$.

**Results** We show an example of the captioning auxiliary task in Figure 2. We find auxiliary tasks (inv-dyn. and captions) can help ground visual prompts and improve success rate for in distribution

tasks. However they also cause VP$^2$ to overfit and perform worse on out of distribution tasks. The last auxiliary task goal pred. seems to decrease performance in both ID and OD.

**Hyperparameters** For auxiliary tasks, we add a tunable task embedding to each context, where each task (action prediction vs. auxiliary task) has a separate task embedding. This embedding helps the LM to learn multiple tasks. We used a task embedding length = 10. To compute the loss, we add a weight parameter to control the loss between action prediction and auxiliary task: $\mathcal{L} = \mathcal{L}_{\mathcal{D}} + \alpha \mathcal{L}_{\text{aux}}$. In our experiments, we tested $\alpha = \{0.1, 1.0\}$ and found $\alpha = 0.1$ works the best. Otherwise, we used the same hyperparameters as Table 3.

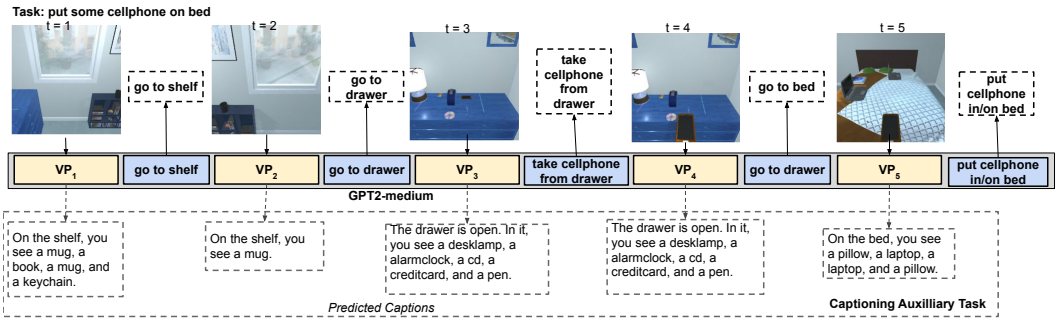


Figure 2: Example of VP$^2$ with the captioning auxiliary task in VizALF. Each observation is encoded into a visual prompt VP$_t$ and used to predict the next action. The LM is also trained to predict a caption for each VP$_t$.