

# Pretrained Language Models as Visual Planners for Human Assistance

Dhruvesh Patel\*  
UMass, Amherst

dhruveshpate@cs.umass.edu

Hamid Eghbalzadeh  
Meta Reality Labs Research

heghbalz@meta.com

Nitin Kamra  
Meta Reality Labs Research

nitinkamra@meta.com

Michael Louis Iuzzolino  
Meta Reality Labs Research

mliuzzolino@meta.com

Unnat Jain  
Meta AI Research

unnatjain@meta.com

Ruta Desai†  
Meta Reality Labs Research

rutadesai@meta.com

## Abstract

To make progress towards multi-modal AI assistants which can guide users to achieve complex multi-step goals, we propose the task of ‘Visual Planning for Assistance (VPA)’. Given a goal briefly described in natural language, e.g., “make a shelf”, and a video of the user’s progress so far, the aim of VPA is to obtain a plan, i.e. a sequence of actions such as “sand shelf”, “paint shelf”, etc. to achieve the goal. This requires assessing the user’s progress from the untrimmed video, and relating it to the requirements of underlying goal, i.e. relevance of actions and ordering dependencies amongst them. Consequently, this requires handling long video history, and arbitrarily complex action dependencies. To address these challenges, we decompose VPA into video action segmentation and forecasting. We formulate the forecasting step as a multi-modal sequence modeling problem and present **Visual Language Model based Planner (VLaMP)**, which leverages pre-trained LMs as the sequence model. We demonstrate that VLaMP performs significantly better than baselines w.r.t all metrics that evaluate the generated plan. Moreover, through extensive ablations, we also isolate the value of language pre-training, visual observations, and goal information on the performance. We will release our data, model, and code to enable future research on visual planning for assistance.

## 1. Introduction

Imagine assembling a new piece of furniture or making a new recipe for a dinner party. To achieve such a goal,

\*Work done while interning at Meta Reality Labs Research.

†Corresponding author: rutadesai@meta.com

<sup>1</sup>The images in the Fig. 1 and Fig. 2 are from Ego4D[19], and are for the purpose illustration only.

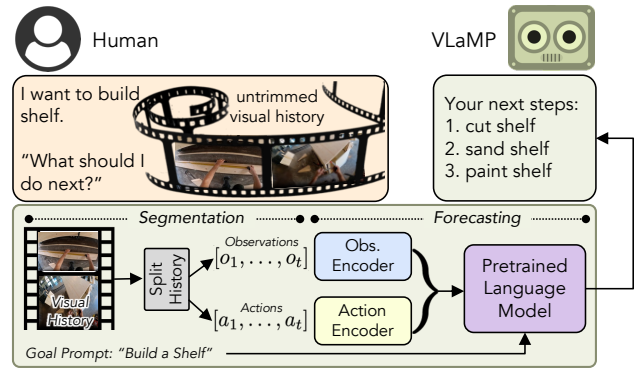


Figure 1: Given a user’s goal described in natural language and corresponding visual history depicting the user’s progress till time  $t$ , the aim of Visual Planning for Assistance is to plan a sequence of actions aimed at achieving the goal. Our approach entails multi-modal sequence modeling using pre-trained video segmentation and language models. <sup>1</sup>

you might follow a manual or a video tutorial, going back and forth as you perform the steps. Instead of fumbling through a manual, imagine an assistive agent capable of being invoked through natural language, having the ability to understand human actions, and providing actionable multi-step guidance for achieving your desired goal.

Towards this next generation of assistive agents, multi-modal neural models should be able to reason human activities from visual observations, contextualize it to the goal at hand, and provide multi-step guidance. To develop such multi-modal neural models, we propose a new and intuitive learning task – *Visual Planning for Assistance (VPA)*.

As illustrated in Fig. 1, given a user-specified goal in natural language and corresponding video observations depict-

ing the user’s progress towards this goal, the VPA objective is to generate the ordered sequence of next actions towards achieving the goal. Complementing extensive research in on forecasting or anticipation in videos, we take inspiration from active or embodied vision [14]. Particularly, we formulate VPA as *goal-conditioned* task planning where the goal is specified in natural language allowing to easily assist humans. Further, we focus on performing complex human activities and on real-world human videos. A ubiquitous category of such multi-step and complex stream of actions is procedural human activities, which include cooking, assembly, repair, *etc.* Focusing on this, we based VPA off instructional videos of such procedural activities from YouTube from large, open-sourced, human datasets.

Given a procedural activity, e.g., building a shelf, the assistant must estimate the user’s progress towards their goal from a long video with potentially many irrelevant background frames, which is challenging. Then, conditioned on the user’s progress, the assistant must generate and recommend a valid plan of actions such that the actions respect the constraints of the activity. For instance, the shelf shouldn’t be painted before sanding it. Reasoning about such constraints present in human activities while generating plans is another challenge for VPA.

To overcome these challenges, we formulate an approach for VPA that is based on video action segmentation and a transformer-based neural sequence modeling, where the former allows us to deal with long video history, while the latter can handle arbitrary sequential constraints [64, 29]. Furthermore, our formulation enables leveraging state-of-the-art pre-trained transformer based language models (PTLMs), which may contain useful priors about action-action similarity, action-goal association, and action ordering [6, 23, 42]. Our model, which we call Visual Language Model Planner (VLaMP), conditions the generated plan onto the visual history by using a transformer-based mapper network that projects embeddings corresponding to visual history into the input space of the LM. We show that VLaMP performs significantly better than the baselines. We also perform ablations to understand the effect of language pre-training of the LM, the role of visual history, and goal description on the model’s performance on VPA.<sup>2</sup>

In summary, our main contributions are: (1) We propose a new learning task VPA that considers the unique needs of real-world vision-powered assistive agents. (2) We introduce a formulation with integrated segmentation and forecasting for VPA, which allows leveraging pre-trained multi-modal encoders and sequence models. (3) Using our formulation, we propose a novel PTLM model called VLaMP, and investigate its efficacy for VPA using two existing

<sup>2</sup>To enable use of our formulation as a benchmark for Visual Planning for Assistance, we release the data processing code, as well as metrics and model implementation.

	Action Prediction	Goal Modality	Visual Reasoning
Action anticipation [11]	Single	None	Video-based
Action forecasting [19]	Multiple, ordered	None	Video-based
Procedural planning [8]	Multiple, ordered	Vision	Image-based
<b>VPA (ours)</b>	Multiple, ordered	Natural Language	Video-based

Table 1: **VPA task vs. prior tasks related to forecasting of real-world human activities.** We predict an ordered sequence of actions based on videos with a focus on natural language goal-conditioning for human-assistive applications.

video datasets of procedural activities: CrossTask [66] and COIN [54].

## 2. Related Work

**Forecasting in Videos.** VPA entails future action sequence prediction for a video and is closely related to anticipation in video understanding, including future localization [17, 30, 59], future frame prediction [58, 34], next active object estimation [16, 4, 19], as well as short- and long-term action anticipation [49, 12, 15, 18, 19, 48, 47, 41, 33, 1, 38, 36]. While short-term forecasting approaches, such as [49], are limited to predicting the single next action occurring only a few seconds into the future, our work focuses on predicting sequences of actions over longer time horizons (on the order of minutes) into the future. The long-term action (LTA) forecasting benchmarking was recently established on a small subset of Ego4D [19], and although VPA is similarly devised to predict a temporally ordered sequence of actions conditioned on video-based visual observations, our approach differs in that the LTA task does no goal-conditioning. Our task incorporates goal-conditioning on long term sequence prediction as we argue this will drive the development of critical system aspects, such as virtual assistants that afford human interaction via the natural language goals. Additionally, in contrast to LTA, we focus on a wide range of goal-oriented activities available within the CrossTask and COIN datasets. Tab. 1 highlights the differences between VPA and other forecasting tasks.

**Procedural Planning Approaches.** VPA is similar to the task of procedure planning [8], wherein given a starting and terminating visual observation, the aim is to predict the actions that would transform the state from starting to terminating. However, we argue that due to the unavailability of the terminating visual state, the procedure planning task is not useful in a real-world assistance setting. While several recent works introduce novel models for procedure planning, these models cannot be deployed for VPA either because they rely heavily on the availability of visual goal [52, 65, 5], or assume access to true action history [35].

**Transformers for Decision Making.** VLaMP autoregres-

sively predicts future states and/or observations and actions and is similar in spirit to sequence models for decision making such as GATO [44], Decision Transformer [9], and Trajectory Transformer [25]. VLaMP extends such models to work with visual states.

**Planning in Embodied AI using LMs.** Past research has leveraged PTLMs for task planning in real world for embodied agents [23, 50, 31, 6, 32, 24, 13, 26]. Many of these works focus on converting a high-level task or instruction into sequence of low-level steps and then ground them in the environment using either affordance functions [6], visual feedback [24], or multimodal prompts [26, 13]. Owing to their focus on robotic agents, these works focus on predominately pick-place tasks. Instead VPA is focused on complex tasks, in which humans might require assistance in the form of recommended future actions. Consequently, VLaMP requires grounding and reasoning of much more complex states and actions, which it accomplishes by finetuning a pretrained LM on multimodal sequences of (visual) observations and (text-based) actions.

**Multi-modal LMs.** Recent works have successfully used the transformer architecture for modeling multi-modal sequences that have visual tokens. For instance, [2, 22, 13] train large transformer based multi-modal sequence prediction models. While [3, 55, 40] focus on adapting pretrained LMs to work with visual tokens by aligning the representation spaces of the two modalities. VLaMP’s approach of modeling sequence of visual and textual representations using PTLMs is similar in spirit to these, *i.e.* VLaMP also uses a mapper network to align video representations to LM’s token space and jointly learns the mapper with LM finetuning for VPA. However, in contrast to previous works, VLaMP predicts the visual tokens autoregressively at inference time to enable forecasting of the state for planning. Consequently, VLaMP’s token prediction loss is multi-modal, unlike most multi-modal LMs that only use token prediction loss for text tokens.

### 3. Visual Planning for Assistance

Here, we introduce the task of Visual Planning for Assistance (VPA), towards enabling multi-step guidance to humans in their real-world activities. We instantiate VPA for procedural activities, where humans routinely seek assistance. In this section, we include the definition of VPA, and describe the evaluation protocol.

#### 3.1. Task Definition

The following two intuitive inputs are given to any model performing VPA.

**Goal Prompt ( $G$ ).** The natural language description (in short phrase) of the user’s goal, emulating a typical user’s request for assistance for a day-to-day task. Examples include, “build a shelf” and “change a tire”.

**Visual History ( $V_t$ ).** An untrimmed video that provides context about the user’s progress towards a goal from the start till time, say  $t$ . We assume that  $V_t$  contains  $k$  actions or steps  $\{a_1, \dots, a_k\}$  pertaining to the goal. However, VPA doesn’t have access to  $\{a_1, \dots, a_k\}$  or  $k$  and must work with  $V_t$ .

Given these two inputs,  $V_t$  and  $G$ , the objective of VPA is to generate a *plan*  $\mathcal{T}$ . The plan is a sequence of actions that should be executed (in the next steps) to assist the user in achieving the goal  $G$ . Concretely, the prediction is denoted by  $\mathcal{T} = (a_{k+1}, \dots, a_{k+l})$ , where  $a_i$  are represented in natural language but come from a closed set  $\mathcal{A}$ . Here,  $l \leq K$  denotes the number of future actions that should be predicted, out of the  $K$  number of remaining actions required to accomplish the goal. For our shelf-building example, the correct  $\mathcal{T} = (\text{sand shelf, paint shelf, attach shelf})$ , capturing the remaining 3 future actions (here,  $l = N = 3$ ).

In day-to-day activities, we request assistance for goals in natural language. However, prior works *procedural planning* [8, 5] assume access to visual goal state. This is not a realistic assumption for an AI agent assisting humans. Hence, we purposefully relax this assumption, making our formulation of the task significantly more practical. Moreover, an agent with access to the visual modality, should be able to improve its plan by filtering out relevant information from the raw video stream of the progress. These are the two central assumptions around which we formulate the task of VPA.

#### 3.2. Evaluation

**Open-Sourced Video Data.** We leverage existing datasets CrossTask [66] and COIN [54], originally developed to enable video action understanding for VPA, based on the following three requirements:

- *Rich diversity of activities from multiple domains:* The data from different domains such as cooking, assembly etc., enables testing of VPA models in a more generalized manner.
- *Goal-oriented activities consisting of long sequences of actions:* Since the objective is to generate  $l$  future actions, activities in these datasets that require diverse sequences of actions e.g., “making a pancake”, instead of “running” are more suitable for VPA.
- *Action annotations from a fixed closed set of actions:* Action labels described using verb-noun from a finite set [11], which are temporally aligned with the videos. This makes evaluating the accuracy of  $\mathcal{T}$  prediction straightforward. Specifically, free-form, narration-style descriptions of actions that are available in recent video datasets aid in efficient multi-modal representation learning for video understanding [19, 39]. However, evaluating the quality of action *sequences* towards goal achievement, where each action is described in free-form natural language, is non-trivial. We

Dataset	# train videos	# test videos	# test samples	actions per video	# goals	# domains
CrossTask [66]	1756	752	4123	7.6 ± 4.3	18	3
COIN [54]	9428	1047	2011	3.9 ± 2.4	180	12

Table 2: **VPA datasets.** We evaluate VPA on two existing video datasets containing multiple goal-oriented procedural activities from varied domains. Such activities contain sequences of multiple actions making them ideal for VPA.

leave the instantiation of VPA with free-form natural language actions as future work.

Table 2 summarizes the features from CrossTask and COIN, aligned with the above requirements.

**Metrics.** The planning performance of a VPA model is measured by comparing the generated plan  $\hat{\mathcal{T}} = (\hat{a}_{k+1}, \dots, \hat{a}_{k+l})$  to the ground truth plan  $\mathcal{T}$  for  $l$  actions in the future, given  $V_t$  and  $G$ . Here  $\hat{a}_{k+i}$  denotes the prediction for the  $k+i$ -th step given history till  $k$ -th step. Consistent with community practices [8, 5], we use the following metrics, listed in decreasing order of strictness: *success rate* (SR), *mean accuracy* (mAcc), *mean intersection over union* (mIOU). Success rate requires an exact match between all actions and their sequence between  $\hat{\mathcal{T}}$  and  $\mathcal{T}$ . Mean accuracy is the accuracy of the actions at each step. Unlike success rate, mean accuracy does not require a 100% match to ground truth. Instead it considers matching at each individual step. Lastly, mean intersection over union captures the cases where the model predicts the steps correctly, but fails to identify the correct order. Concretely,

$$\text{mIOU}_l = \frac{|\hat{a}_{\{k+1:k+l\}} \cap a_{\{k+1:k+l\}}|}{|\hat{a}_{\{k+1:k+l\}} \cup a_{\{k+1:k+l\}}|}, \quad (1)$$

$$\text{mAcc}_l = \frac{1}{l} \sum_{i=1}^l \mathbb{1}[\hat{a}_{k+i} = a_{k+i}], \quad (2)$$

$$\text{SR}_l = \prod_{i=1}^l \mathbb{1}[\hat{a}_{k+i} = a_{k+i}], \quad (3)$$

where  $\mathbb{1}[\cdot]$  is the identity function, which is 1 when the condition in its input is true, and 0, and  $\hat{a}_{\{k+1:k+l\}}$  denotes the set of  $l$  future actions in  $\hat{\mathcal{T}}$ , i.e., a sequence but disregarding the order. To complement the above metrics, we also measure the accuracy of predicting the next action i.e. nAcc, as defined in (4), where ‘n’ stands for next. Note that all metrics are averaged over the test set details of which are included in Sec. 5.

$$\text{nAcc} = \mathbb{1}[\hat{a}_{k+1} = a_{k+1}] \quad (4)$$

## 4. Visual LM Planner

VPA can be viewed as a sequential decision making problem, where the model (say,  $\pi$ ) predicting the sequence of next actions is a policy conditioned on the visual history  $V_t$ , serving as (partially-observed) state, and goal prompt  $G$ . Inspired by the offline learning formulation closest to ‘learning from offline demonstrations’ [20, 57, 27, 60], in VLAMP, we formulate VPA as a goal-conditioned, multi-modal sequence prediction problem.<sup>3</sup> This formulation allows us to leverage high-capacity sequence models like Transformers [56], which have been successfully applied to sequential decision making [25, 9]. Furthermore, inspired by the recent success of pre-trained transformer language models (PTLMs) on such tasks, [45, 23, 9], we propose to leverage PTLMs as the sequence model in our formulation. In the remainder of this section we describe our approach for VPA, called **Visual Language Model based Planner** (VLAMP), consisting of a segmentation module and PTLM based sequence prediction module.

### 4.1. Planning with Segmentation and Forecasting

We define  $\pi$  as a goal-conditioned, multi-modal sequence prediction problem:

$$\pi = P(a_{k+1}, a_{k+2}, \dots | V_t, G). \quad (5)$$

where  $\pi$  models the probability of goal-relevant and valid future actions sequences conditioned on  $V_t$  and  $G$ .

Modeling the multi-modal sequence in Eq. 5 is computationally expensive and difficult to scale because of the high-dimensional state space of raw untrimmed video  $V_t$ .<sup>4</sup> Also, there is limited data to learn the distribution over valid action sequences for goals in real-world applications. Tackling these challenges, we argue that the latent space of factors influencing the future plan can indeed be expressed in lesser dimensions. Particularly, to ensure scalability for handling raw videos and sample efficiency in learning valid action sequence distributions, we *decompose* our policy  $\pi$  into two modules. The first module is *video segmentation*, which converts the untrimmed video history  $V_t$  into a sequence of video segments i.e. a segment history  $S_k = (s_1, \dots, s_k)$ , where each segment corresponds to an action  $a_i$  that occurred in the video. The second module enables *forecasting* i.e. it transforms the output of the segmentation module and generates the plan. While a probabilistic formulation of this decomposition can be expressed as:

$$\pi = \sum_{S_k} \underbrace{P(a_{k+1}, a_{k+2}, \dots | S_k, G)}_{\text{Forecasting}} \underbrace{P(S_k | V_t)}_{\text{Segmentation}}, \quad (6)$$

<sup>3</sup>Future works may explore alternate policy optimization formulations based on reward shaping, inverse RL, or by employing additional online interactions through photorealistic simulation [28, 46, 53, 51].

<sup>4</sup>A typical untrimmed video of 100p video of 5 minutes at 8fps will have  $2^7$  raw pixel values.

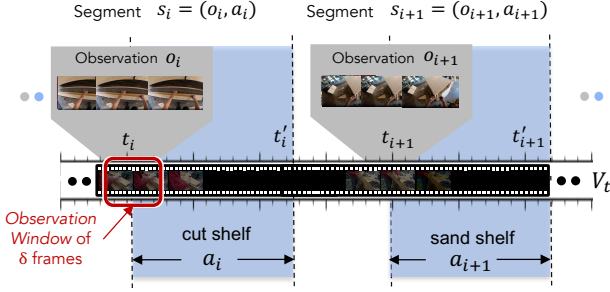


Figure 2: **VLaMP – Segmentation Module.** The untrimmed visual history  $V_t$  is converted into segments, each consisting of observation  $o_i$  and the action  $a_i$ . The observation  $o_i$  is the collection of video frames of  $\delta$  seconds around the start time stamp  $t_i$  of the corresponding action  $a_i$ . Two such segments are shown here.

where the segment history  $S_k$  is a latent variable, the summation over all possible segment histories is intractable. We, however, use Eq. 6 as the guiding expression to formulate the input and output of both modules. Next, we include the technical details of both of these modules.

## 4.2. Segmentation Module

This module splits the untrimmed video history  $V_t$  into segment history  $S_k$  of multiple segments, each segment corresponding to an action. The segmentation is done using a video-action segmentation model in three steps: pre-processing, classification, and consolidation. In the pre-processing step, the raw video frames from  $V_t$  are bundled into fixed-length window clips  $c_i$ , each of length 1 second, to obtain  $\tilde{V}_t = (c_1, \dots, c_t)$ . In the classification step, a video-action segmentation model is used to output the most probable action for each clip  $c_i$ , which can be denoted as  $\tilde{A}_t = (\tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_t)$ . Finally, in the consolidation steps, we convert  $\tilde{A}_t$  into a form that can be used by the forecasting module; this form consists of two sequences: **action history**  $A_k$  and **observation history**  $O_k$ . To this end, same actions in consecutive seconds in  $\tilde{A}_t$  are consolidated to form the action history  $A_k = (a_1, \dots, a_k)$ . As illustrated in Fig. 2, assuming  $t_i$  denotes the starting timestamp for  $a_i$ , we also extract video frames from  $t_i - \delta/2$  to  $t_i + \delta/2$  to obtain a *observation window*  $o_i$  corresponding to  $a_i$ , and consequently the full observation history  $O_k = (o_1, \dots, o_k)$ . The resultant segment history is termed  $S_k = ((o_1, a_1), \dots, (o_k, a_k))$ , summarized in Fig. 2.

## 4.3. Forecasting Module

The usefulness of  $\pi$ 's decomposition expressed in Eq. (6) becomes apparent now. Modeling the segmentation module's output as segment history  $S_k$ , where each segment consisting of action and observations, allows writing the

output of the forecasting module in an autoregressive manner:

$$P(a_{k+1}, a_{k+2}, \dots \mid o_1, a_1, \dots, o_k, a_k, G) = \prod_{i>0} \sum_{o_{k+i}} P(o_{k+i}, a_{k+i} \mid o_1, a_1, \dots, o_k, a_k, G). \quad (7)$$

Illustrated in Fig. 3, this autoregressive expression allows the possibility of using any sequence-to-sequence neural network as the forecasting module in combination with pretrained text and video encoders for representing action and observation history  $A_k$  and  $O_k$ . This general-purpose framework allows the use of any neural sequence model – LSTM [21], GRU [10], Transformers [56], *etc.* We choose to instantiate the forecasting module for  $\pi$  using a pretrained transformer-based LM. Next we present the details of the encoders for the two modalities and the LM based sequence model.

**Action encoder ( $f_{\text{act}}$ ).** Each action  $a_i$  in  $A_k$  is encoded by  $f_{\text{act}}$  and the output is denoted by  $\alpha_i$ . Concretely, token embeddings are expressed as:

$$(\alpha_1, \dots, \alpha_k) = (f_{\text{act}}(a_1), \dots, f_{\text{act}}(a_k)), \text{ where} \\ \alpha_i = (\alpha_i^1, \dots, \alpha_i^{r_i}) \in \mathbb{R}^{r_i \times d} \quad (8)$$

As we illustrate in Fig. 3 (left), each action  $a_i$  is tokenized into  $r_i$  tokens using appropriate tokenizer for the LM, the tokens are indexed using the vocabulary of the LM, and are represented using an embedding lookup from the token embeddings of the LM to produce  $\alpha_i$ . Here,  $r_i$  is the number of tokens and  $d$  is the dimensionality of token embeddings.

**Observation encoder ( $f_{\text{obs}}$ ).** Visual cues play an integral role in knowing what actions lie ahead, towards achieving the goal prompt  $G$ . To this end, the visual observations  $O_k$  are encoded and play a critical role in the planner. Recall, the visual observation history  $O_k$  comprises of  $o_i$  corresponding to action  $a_i$ , each of  $\delta$  frames. As illustrated in Fig. 3 (middle), we transform each  $o_i$  employing the widely-adopted S3D backbone [62]  $f_{\text{S3D}^*}$  (\* denotes backbone is frozen). We must project visual encodings to a shared latent space of action (language) embeddings described before ( $\alpha_i$ ). To this end, we map S3D features via a trainable transformer mapper  $f_{\text{map}}$ . Concretely,

$$(\beta_1, \dots, \beta_k) = (f_{\text{obs}}(o_1), \dots, f_{\text{obs}}(o_k)), \text{ where} \\ \beta_i = (\beta_i^1, \dots, \beta_i^\delta) \in \mathbb{R}^{\delta \times d} \text{ and } f_{\text{obs}} = f_{\text{S3D}^*} \circ f_{\text{map}} \quad (9)$$

Overall, as we show in Fig. 3 (right), the resultant encoded sequence of representation for  $S_k$  is thus

$$f_{\text{enc}}(S_k) = (\beta_1, \alpha_1, \dots, \beta_k, \alpha_k) = H_k,$$

**Sequence model ( $f_{\text{seq}}$ ).** Given the above encoding for the segment history  $S_k$ , the role of the sequence model is to

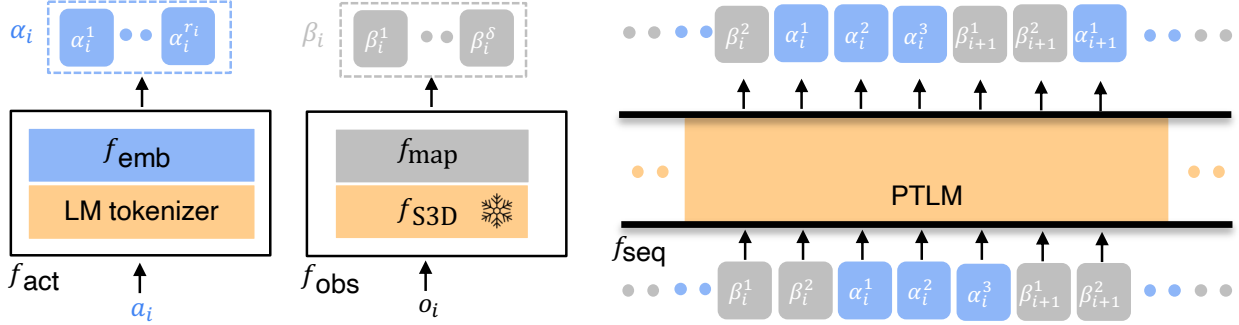


Figure 3: **VLaMP – Forecasting Module.** As shown in left and middle, actions and observations obtained from the segmentation module are encoded using appropriate modality encoders. The observation encoder leverages pretrained video encoder for observations, while also learning a mapper that aligns the representations from observations with actions. As shown on the right, VLaMP uses a joint sequence model on top of interleaved action (blue) and observation (gray) representations to forecast autoregressively the next representation.

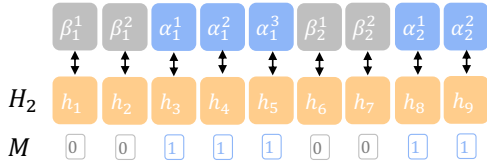


Figure 4: **Tokenized Sequence with Masks.** The encoded sequence of representations for  $k = 2$  segments, denoted alternatively using modality agnostic notation of  $H_2$  and mask  $M$  for next token prediction training.

predict a representation of the next token, that would in return enable VLaMP plan generation capabilities for VPA. Importantly, in the process of generating sequence of future actions autoregressively, we would also need to generate the representations of ‘future observations’. Therefore, as we shown in Fig. 3, our sequence model, which consists of the transformer layers of a PTLM, also produces representations for vision (in addition to the necessary action tokens). Before proceeding further, we pause and introduce additional notation for the sequence model, which will make the subsequent explanation for training and inference easier to follow. As shown in Figure 4, we alternatively denote the sequence of representations  $(\beta_1, \alpha_1, \dots, \beta_k, \alpha_k)$  by  $H_k = (h_1, \dots, h_n)$ , with  $n = k\delta + \sum_{i=1}^k r_i$ . A binary mask  $M = (m_1, \dots, m_n)$ , where  $m_i$  is 1 if the corresponding representation is for an action and 0 otherwise, can help obtain necessary action or visual observations. With this notation, given first  $j$  representations denoted as  $h_{1:j}$ , one step of the sequence model produces the representation for  $j + 1$ , i.e.,  $f_{\text{seq}}(h_{1:j}) = \hat{h}_{j+1}$ .

#### 4.4. Training

The joint training of the segmentation and forecasting modules following Eq. (6) is intractable.<sup>5</sup> But, by exploiting the availability of unpaired training data, we approximate Eq. (6) by feeding in the output of the segmentation module to the forecasting module and training them separately, each on their respective labeled data. The video-action segmentation model is trained utilizing the Video-CLIP setup [63], where in the segmentation model performs classification to predict the action for each second of the video. The forecasting model is trained by adopting the next representation prediction objective. Unlike vanilla LM pretraining, however, we also need to train for predicting visual representations in addition to text (action). Therefore, we use two different losses  $L_{\text{act}}$  and  $L_{\text{obs}}$  for text and visual representations respectively. Specifically,  $L_{\text{act}}$  is the conventional cross-entropy loss over the LM’s vocabulary  $V_{\text{LM}}$  for the action representations while  $L_{\text{obs}}$  is the mean-squared error between the predicted and the ground truth observation representations. The total loss is the sum of both the loss terms as shown in Eq. (10). In order to have a stable training, we use ground truth action history to construct  $S_k$  (and subsequently  $H_k$ ) instead of the output of the segmentation module. Appendix provides further details on loss and optimizers for training.

$$L = - \sum_{j=1}^n m_j L_{\text{act}}(\hat{h}_j) + (1 - m_j) L_{\text{obs}}(\hat{h}_j), \text{ where}$$

$$L_{\text{act}} = h_j \cdot \hat{h}_j - \sum_{p=1}^{|V_{\text{LM}}|} h_p \cdot \hat{h}_j; \quad L_{\text{obs}} = \frac{\|h_j - \hat{h}_j\|_2^2}{d} \quad (10)$$

<sup>5</sup>Despite tuning attempts, we found joint optimization to be intractable and inefficient on resources.

**Inference.** We next detail the inference procedure for VLaMP. Recall that we use  $1 : n$  to denote a sequence of  $n$  representations (i.e.,  $h_{1:n} = (h_1, \dots, h_n)$ ). Additionally, we denote the concatenation operator over two representation sequences by “ $\diamond$ ”. With this notation at hand, we define the *score* of an action  $a \in \mathcal{A}$  for following history  $h_{1:n}$  as

$$\phi(h_{1:n} \diamond f_{\text{act}}(a)) = \sum_{j=1}^{r_a} a^j \cdot f_{\text{seq}}(h_{1:n} \diamond a^{1:j}), \quad (11)$$

where  $\cdot$  is the vector dot product, and  $f_{\text{act}}(a) = \alpha^{1:r_a} = (\alpha^1, \dots, \alpha^{r_a})$  is the sequence of encoded representations for action  $a$ . In other words, this *score* is the sum of unnormalized log-probability under the sequence model using the standard softmax distribution. We use this scoring function with to perform beam search.<sup>6</sup>

## 5. Experiments

We instantiate VLaMP’s segmentation module utilizing VideoCLIP [63] fine-tuned on COIN and CrossTask; and the sequence model  $f_{\text{seq}}$  (in the forecasting module) by GPT2 [43]. We utilize the open-sourced parameters of GPT2, obtained from HuggingFace [61]. For inference, we use Algo. 1 presented in Appendix C, with beam size  $B = 10$  for CrossTask and a  $B = 3$  for COIN.

### 5.1. Data and Baselines

**Data.** For a video  $V$  with goal  $G$ , both CrossTask [66] and COIN [54] provide annotations of the form  $\{a_k, (t_k, t'_k)\}_{k=1}^K$ , where  $a_k$  are the actions in the video, and  $t_k$  (resp.  $t'_k$ ) are the start (resp. end) timestamps  $a_k$ . Given an annotated video consisting of  $K$  steps, we generate  $K - l$  examples, each with input  $x_k = (G, V_{t_k})$  and output  $y_k = (a_{k+1}, \dots, a_{k+l})$ , for  $k = 1, \dots, K - l$  (leaving at least  $l$  steps to predict in each example).<sup>7</sup> Therefore from  $M$  videos, we generated  $N = \sum_{m=1}^M (K_m - l)$  examples, where  $K_m$  is the number of steps in the  $m$ -th video, forming a dataset  $\mathcal{D} = \{x^{(j)}, y^{(j)}\}_{j=1}^N$  suitable for VPA (total number of samples in shown in Table 2).

**Baselines.** As a first step towards benchmarking, we utilize a random baseline, and additionally adopt a variety of strong goal-conditioned models. The procedure planning task is most relevant task from the literature to VPA, therefore, we adapt<sup>8</sup> the widely used DDN model introduced by Chang *et al.* [8], since it is an established model in many benchmarks in prior works [5, 65, 52], hence, is chosen for our experiments. The key details regarding our baselines are as follows<sup>9</sup>:

<sup>6</sup>The inference algorithm is presented in Algo. 1 Appendix C

<sup>7</sup>Since we evaluate for three and four next steps on our datasets, we use the maximum required length and set  $l = 4$ .

<sup>8</sup>We detail these adaptations in Appendix B.1.

<sup>9</sup>Additional details are deferred to Appendix B.

- *Random*: Predicts the plan by picking all  $l$  actions uniformly randomly from the set of all actions  $\mathcal{A}$ .
- *Random w/ goal*: A stronger baseline; for each goal  $G$ , we allow privilege access to a set of applicable actions to that goal  $\mathcal{A}_G \subseteq \mathcal{A}$ , and predicts the plan by randomly picking actions from the restricted set.
- *Dual Dynamics Network (DDN)* [8]: Accounting for the difference in task definition, i.e. lack of visual goal, a direct application of DDN inference algorithm was not possible. So we keep DDN’s network structure but use Algo. 1 for inference on VPA.

### 5.2. Quantitative Results

In the following, we include quantitative findings of benchmarking methods on two video data sources ( Tab. 3) and head-on ablations ( Tab. 4).

**Improved performance across video datasets.** As we show in Tab. 3, VLaMP significantly outperforms the baselines. DDN that is customized for procedural tasks performs significantly better than heuristics leading to a mAcc boost from 12.7  $\rightarrow$  24.1% (row 2 and 3,  $l = 4$ , Tab. 3). With our novel decomposition and pretraining objective, VLaMP outperforms these baselines with a further bump up from 24.1  $\rightarrow$  31.7% (row 3 and 4).

**Steady gains in short & long horizon predictions.** Comparing columns corresponding to  $l = 1$  and  $l = 4$ , the performance (as one might expect) decreases, across all baselines and tasks. However, zooming in on COIN results, we observe consistent gains of VLaMP over DDN. Particularly, a relative improvement of 54% (29.3  $\rightarrow$  45.2) for  $l = 1$  (row 7 and 8, Tab. 3) and a relative improvement of 68% (21.0  $\rightarrow$  35.2) in mAcc.

**Privileged random baseline comes close to DDN on easy metrics.** Interestingly, we observe the performance of *Random w/ goal* comes close to DDN, when evaluated for lenient metrics like mAcc and mIOU (in COIN section of Tab. 3). Note, *Random w/ goal* enjoys the privileged access to (a much smaller) ‘relevant actions set’ for a given goal. On an average, the relevant or feasible action set is smaller for COIN than CrossTask videos. This a priori access to feasible actions makes *Random w/ goal* a competitive baseline.

**Goal-conditioning is crucial.** A key difference between procedure planning and VPA is goal conditioning is provided to the neural policies. In Tab. 4 (rows 1 and 2), we measure the effect of providing the goal as a *textual description* for last-observation-only models. The only difference is goal prompt  $G$ , which increases mAcc performance from 44.5  $\rightarrow$  53.1% for  $l = 1$  and 28.3  $\rightarrow$  34.7% for  $l = 3$ .

### 5.3. Ablations and Error Analysis

Utilizing head-on ablations, we evaluate how the action history and the visual history affect the plan generation for

Data	Model	$l = 1$		$l = 3$		$l = 4$		
		{n/m}Acc	SR	mAcc	mIOU	SR	mAcc	mIOU
Cross-task	Random	0.9	0.0	0.9	1.5	0.0	0.9	1.9
	Rand.(w/ G)	13.2	0.3	13.4	23.6	0.0	12.7	27.8
	DDN [8]	33.4	6.8	25.8	35.2	3.6	24.1	37.0
	VLaMP (ours)	<b>50.6</b>	<b>10.3</b>	<b>35.3</b>	<b>44.0</b>	<b>4.4</b>	<b>31.7</b>	<b>43.4</b>
COIN	Random	0.1	0.0	0.1	0.2	0.0	0.1	0.2
	Rand.(w/ G)	24.5	1.7	21.4	42.7	0.3	20.1	47.7
	DDN [8]	29.3	10.1	22.3	32.2	7.0	21.0	37.3
	VLaMP (ours)	<b>45.2</b>	<b>18.3</b>	<b>39.2</b>	<b>56.6</b>	<b>9.0</b>	<b>35.2</b>	<b>54.2</b>

Table 3: **Performance on different datasets and horizons.** The mean of various metrics (Sec. 3.1) obtained using 5 runs with different random seeds (std. errors are provide in Appendix I). Note that the action and observation history are the output of the separately finetuned video-action segmentation model and hence are noisy compared to the ground truth history.

	$G$	$A_k$	$O_k$	$l = 3$		$l = 1$	
				SR	mAcc	mIOU	nAcc
1	$\times$	$\times$	$o_k$	$6.8 \pm 0.3$	$28.3 \pm 1.9$	$34.8 \pm 2.0$	$44.5 \pm 3.8$
2	$\checkmark$	$\times$	$o_k$	$8.9 \pm 0.2$	$34.7 \pm 0.7$	$41.6 \pm 0.8$	$53.1 \pm 2.0$
3	$\checkmark$	$\checkmark$	$\times$	$14.9 \pm 0.3$	$37.8 \pm 0.4$	$50.8 \pm 0.6$	$48.0 \pm 0.2$
4	$\checkmark$	$\checkmark$	$O_k$	$15.2 \pm 0.3$	$43.5 \pm 0.8$	$51.4 \pm 0.9$	$64.8 \pm 0.9$
<b>R</b>	$\checkmark$	$\checkmark$	$O_k$	$10.7 \pm 0.2$	$36.5 \pm 0.7$	$41.7 \pm 0.6$	$61.4 \pm 1.8$

Table 4: **Role of different inputs and LM pre-training in VLaMP.** Here  $G$ ,  $A_k$ ,  $O_k$  specify the inputs provided to VLaMP during training and inference, in terms of goal, action history and observation history respectively.  $o_k$  refers to the use of most recent observation from the history instead of the full observation history. The mean  $\pm$  std. error over 5 runs with different random seeds on CrossTask are shown. The row **R** corresponds to VLaMP trained with random initialization as compared to LM pretraining.

VPA. To remove confounding factors, in Tab. 4 we use the ground truth output for the segmentation module.

**Action and observation history improve complementary planning metrics.** As seen in all the rows with action history Tab. 4, the provision of action history increases difficult metrics such as SR. In comparing rows 2 and 3, we see that SR increases by about 67% ( $8.9 \rightarrow 14.9$ ), simply by using action history even without access to any past observation. However, the lack of observation history affects nAcc, which drops by 10% ( $53.1 \rightarrow 48.0$ ) when observation history is swapped by action history between rows 2 and 3.

**Priors from the pre-trained LM improve performance.** The row **R** of Tab. 4 shows the performance of VLaMP when its forecasting module is trained with random initialization, i.e., the transformer of the same architecture trained

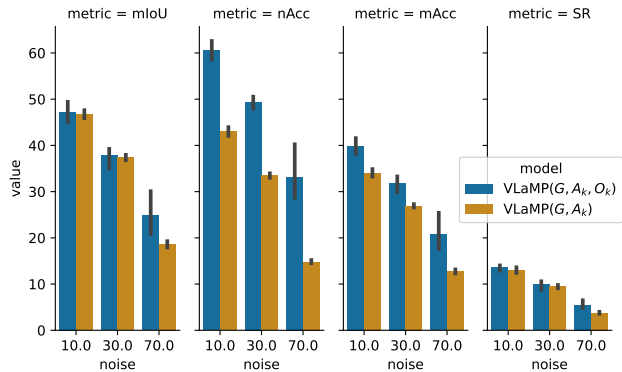


Figure 5: **Effect of segmentation errors.** Performance of VLaMP on CrossTask with classification noise (%) in the segmentation.

from scratch instead of LM pre-training. The performance in row **R** is thus much lower than that of VLaMP with pre-trained LM shown in Tab. 3, which highlights the importance of LM pre-training.

**Segmentation errors are detrimental.** The accuracy of finetuned VideoCLIP model for video-action segmentation is 80.2 and 68.7 when segmenting a video per second for CrossTask and COIN datasets respectively. The effect of such segmentation error in VLaMP’s performance can be observed by comparing row 4 of Table 4, which uses *ground-truth* segments, with row 4 of Table 3 that uses a finetuned segmentation model – all metrics significantly drop due to the introduced segmentation error.

**Visual history aids in planning when segmentation has errors.** The error of the segmentation module may lead to mis-classification of actions leading to erroneous action history. In order to systematically study the effect of erroneous action history on VLaMP’s performance in visual planning, decoupled from VideoCLIP’s segmentation accuracy, we perform controlled experiments wherein we add noise in the ground truth segmentation. Specifically, we replace varying amounts of actions in the ground truth action history by random actions. We also compare two models –VLaMP( $G, A_k, O_k$ ), which uses both action and observation history, and VLaMP( $A, A_k$ ), which only uses action history. As the noise increases, the gap in the performance of the model with access to visual history and the one without, rises. This provides evidence for usefulness of observations for robustness against segmentation classification error.

**Errors increase towards the tail of the activities.** In Fig. 6, we shows mAcc w.r.t. the number of steps  $k$  in the history. The performance drops towards the tail of the activities inspite of access to longer history. Furthermore, this drop is most significant for the activities that have many steps like make pancakes, add oil to car, etc. .



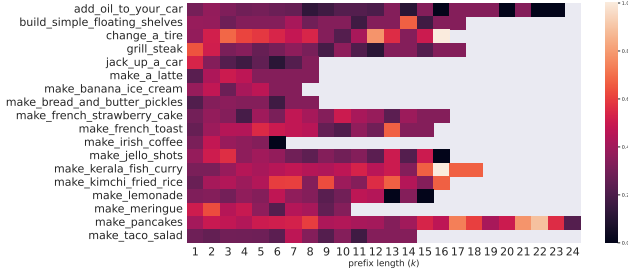


Figure 6: **Prediction in the tail of long activities is challenging.** mAcc for  $l = 3$  w.r.t. the number of steps in history ( $k$ ).

## 6. Conclusion

Visual Planning for Assistance is a new and intuitive formulation to support planning from raw visual observations for assisting humans in day-to-day activities. We benchmark VPA using standard prior work and a new multi-modal sequence modeling formation of VLaMP. The novel decomposition of a VLaMP policy into video action segmentation and forecasting leads to several efficiency and modeling benefits. Particularly, this allows to leverage pre-trained LM that leads to significant performance gains. Alternative decompositions and self-supervised pre-training objectives for PTLMs are interesting ways forward for VPA.

## References

- [1] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what?-anticipating temporal occurrences of activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5343–5352, 2018. 2
- [2] Armen Aghajanyan, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, Mandar Joshi, Gargi Ghosh, Mike Lewis, et al. Cm3: A causal masked multimodal model of the internet. *arXiv preprint arXiv:2201.07520*, 2022. 3
- [3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022. 3
- [4] Gedas Bertasius, Hyun Soo Park, X Yu Stella, and Jianbo Shi. First-person action-object detection with egonet. In *RSS*, 2017. 2
- [5] Jing Bi, Jiebo Luo, and Chenliang Xu. Procedure planning in instructional videos via contextual modeling and model-based policy learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15611–15620, 2021. 2, 3, 4, 7
- [6] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say:

- Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*, 2022. 2, 3
- [7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 12, 13
- [8] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure planning in instructional videos. In *European Conference on Computer Vision*, pages 334–350. Springer, 2020. 2, 3, 4, 7, 8, 12, 15
- [9] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Sriniwas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 2021. 3, 4
- [10] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 5
- [11] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018. 2, 3
- [12] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision. *arXiv preprint arXiv:2006.13256*, 2020. 2
- [13] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*, 2023. 3
- [14] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022. 2
- [15] Yazan Abu Farha, Qiuhong Ke, Bernt Schiele, and Juergen Gall. Long-term anticipation of activities with cycle consistency. *arXiv preprint arXiv:2009.01142*, 2020. 2
- [16] Antonino Furnari, Sebastiano Battiato, Kristen Grauman, and Giovanni Maria Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 2017. 2
- [17] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *ICCV*, 2019. 2
- [18] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13505–13515, 2021. 2
- [19] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson

- Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. 1, 2, 3, 14
- [20] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018. 4
- [21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5, 12, 14
- [22] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, et al. Language is not all you need: Aligning perception with language models. *arXiv preprint arXiv:2302.14045*, 2023. 3
- [23] 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. 2, 3, 4, 12, 13
- [24] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 3
- [25] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *NeurIPS*, 2021. 3, 4
- [26] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022. 3
- [27] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *ICML*, 2018. 4
- [28] 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 preprint arXiv:1712.05474*, 2017. 4
- [29] Anastasis Kratsios, Behnoosh Zamanlooy, Tianlin Liu, and Ivan Dokmanić. Universal Approximation Under Constraints is Possible with Transformers, Feb. 2022. [arXiv:2110.03303 \[cs, math\]](https://arxiv.org/abs/2110.03303). 2
- [30] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014. 2
- [31] 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. 3
- [32] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022. 3
- [33] Siyuan Brandon Loh, Debaditya Roy, and Basura Fernando. Long-term action forecasting using multi-headed attention-based variational recurrent neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2419–2427, 2022. 2
- [34] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016. 2
- [35] Weichao Mao, Ruta Desai, Michael Louis Iuzzolino, and Nitin Kamra. Action dynamics task graphs for learning plannable representations of procedural tasks. *arXiv preprint arXiv:2302.05330*, 2023. 2
- [36] Esteve Valls Mascaró, Hyemin Ahn, and Dongheui Lee. Intention-conditioned long-term human egocentric action anticipation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6048–6057, 2023. 2
- [37] Esteve Valls Mascaró, Hyemin Ahn, and Dongheui Lee. Intention-conditioned long-term human egocentric action anticipation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6048–6057, January 2023. 15
- [38] Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. A variational auto-encoder model for stochastic point processes. in 2019 *ieec*. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3160–3169. 2
- [39] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640, 2019. 3
- [40] Ron Mokady, Amir Hertz, and Amit H Bermano. Clip-cap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021. 3
- [41] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 163–172, 2020. 2
- [42] Fabian Paischer, Thomas Adler, Vihang Patil, Angela Bitto-Nemling, Markus Holzleitner, Sebastian Lehner, Hamid Eghbal-Zadeh, and Sepp Hochreiter. History compression via language models in reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17156–17185. PMLR, 17–23 Jul 2022. 2
- [43] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. [https://d4mucfpksywv.cloudfront.net/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf), 2019. 7

- [44] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. 3
- [45] Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. proscript: Partially ordered scripts generation via pre-trained language models. *arXiv preprint arXiv:2104.08251*, 2021. 4
- [46] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. *ICCV*, 2019. 4
- [47] Fadime Sener, Rishabh Saraf, and Angela Yao. Transferring knowledge from text to video: Zero-shot anticipation for procedural actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2
- [48] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long-range video understanding. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 154–171. Springer, 2020. 2
- [49] Fadime Sener and Angela Yao. Zero-shot anticipation for instructional activities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 862–871, 2019. 2
- [50] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022. 3
- [51] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR, 2022. 4
- [52] Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, and Animesh Garg. Plate: Visually-grounded planning with transformers in procedural tasks. *IEEE Robotics and Automation Letters*, 7(2):4924–4930, 2022. 2, 7, 14
- [53] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021. 4
- [54] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019. 2, 3, 4, 7
- [55] 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. 3
- [56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 4, 5, 14
- [57] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. 4
- [58] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017. 2
- [59] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016. 2
- [60] Luca Weihs, Unnat Jain, Iou-Jen Liu, Jordi Salvador, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. Bridging the imitation gap by adaptive insubordination. In *NeurIPS*, 2021. 4
- [61] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020. 7
- [62] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 318–335. Cham, 2018. Springer International Publishing. 5, 12
- [63] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv:2109.14084*, 2021. 6, 7, 12
- [64] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are Transformers universal approximators of sequence-to-sequence functions? Feb. 2022. 2
- [65] He Zhao, Isma Hadji, Nikita Dvornik, Konstantinos G Derpanis, Richard P Wildes, and Allan D Jepson. P3iv: Probabilistic procedure planning from instructional videos with weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2938–2948, 2022. 2, 7, 14
- [66] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3537–3545, 2019. 2, 3, 4, 7

## Appendix – Pretrained Language Models as Visual Planners for Human Assistance

We structure the supplementary material as follows:

- A. Necessary details and clarifications that we couldn’t include due to space in the main paper.
- B. Implementation details of DDN [8]. Two new baselines – GPT3-based [7] language-only method and another based on most-probable actions.
- C. Step-by-step algorithm for inference, for reproducibility and technical details.
- D. Optimization, hardware, and training details associated with training VLaMP.
- E. Comparisons to Ego4D’s LTA benchmark.
- I. Expanded empirical results, benchmarking the above additional baselines, and deep-dive into error analysis.

We will open-source our model code and weights, as well as the splits of the dataset to enable further research on VPA.

### A. Clarifications

Due to space, some of the explanations in the main paper might be insufficient or unclear in a first read. Hence, we pre-emptively clarify some of these nuances in this section.

- VLaMP uses a randomly initialised forecasting model (L790) → This refers to VLaMP when its forecasting module is trained with random initialization, i.e., the transformer of the same architecture trained from scratch instead of LM pre-training.
- Per-second accuracy of video-action segmentation (L798-799) → The finetuned VideoCLIP [63] for video-action segmentation outputs an action label for each second of a video. We report this accuracy following the convention in the original VideoCLIP paper, averaged over all videos in the test set.
- (L615-617) A more accurate representation should include a the log-sum-exp operator. The correct expression is

$$L_{\text{act}}(\hat{h}_j) = h_j \cdot \hat{h}_j - \log \sum_{p=1}^{V_{\text{LM}}} \exp(h_p \cdot \hat{h}_j).$$

### B. DDN and Additional Baselines

In this section we include more information about baselines that we benchmark on VPA in experiments (Sec. 5). *First*, we include necessary details of reproducing the DDN [8] and how we keep it consistent and fair to the proposed VLaMP. *Second*, we provide two additional baselines – a heuristic baseline, which leverages the structure of our goal-oriented activities for generating plans and a prompt-based baseline using a large LM. *Finally*, we briefly discuss

Data	Model	$l = 1$		$l = 3$	
		{n/m}Acc	SR	mAcc	mIOU
COIN	GPT-3 [23]	12.1	0.5	14.3	22.3
	VLaMP (ours)	<b>67.2</b>	<b>25.5</b>	<b>51.6</b>	<b>59.1</b>

Table 5: **Performance of VLaMP vs. GPT-3 prompting.** Inspired from [23], we sub-sample 310 videos from the test set of COIN to test a prompting-based, language-only baseline using GPT-3 [7], and compare it with VLaMP on the same videos. Such an application of prompting with GPT-3 does not perform well on VPA.

prior procedural planning methods, which we choose not to compare with.

#### B.1. DDN [8]

**Technical Background.** Chang *et al.* [8] proposed Dual Dynamics Network (DDN) for procedural planning. The objective is to learn a latent space representation of observations and actions in addition to a dynamics and conjugate dynamics model that operate over this latent space. The latent representations and recurrent RNN-based dynamics model are learned together by minimizing a joint loss over predicted observations and actions. Such dynamics modeling in latent space is similar in spirit to the forecasting module in VLaMP (Eq. (7)).

**Implementation.** As shown in Fig. 8, we instantiate DDN for VPA by using an LSTM-based [21]  $f_{\text{seq}}$  in the forecasting module, which operates over the observation representations obtained using the same observation encoder  $f_{\text{obs}}$  consisting of pretrained S3D [62] and a mapper as VLaMP and action representations from an embedding layer-based action encoder  $f_{\text{act}}$ . Unlike VLaMP, where the mapper aims to project the visual observation representations into the input space of the pretrained LM, the mapper in DDN only provides trainable parameters to finetune the frozen S3D representations for downstream dynamics model. Both  $f_{\text{seq}}$  and  $f_{\text{act}}$  are initialized with random weights. Just as VLaMP, DDN is trained using cross-entropy loss for predicted actions and mean-squared error for predicted observation representations to jointly learn  $f_{\text{obs}}$ ,  $f_{\text{act}}$ , and the sequence model. At inference, the model is unrolled autoregressively (with beam search as shown in Algo. 1), for prediction of both action and observation representations. These design choices are consistent with VLaMP.

#### B.2. Additional Baseline: GPT-3 Planner (Tab. 5)

Following Huang *et al.* [23], where the authors use a LLM as zero-shot planners, we too also experiment with prompting a frozen pretrained large language model (GPT-3) for VPA. Specifically, a goal prompt and the current his-

**Algorithm 1: Inference for VLaMP and baselines**

```

Data: encoded representations for history  $H_k = (h_1, \dots, h_n)$ , beam size  $B$ 
Result: plan for next  $l$  steps  $\hat{\mathcal{T}} = a_{k+1}, \dots, a_{k+l}$ 
1  $\mathcal{H}_0 \leftarrow \{H_k\};$  // Initialize the set of encoded trajectories with the history
2 for  $i = 1, \dots, l;$  // predict actions for  $l$  steps
3 do
4    $\mathcal{H}_i \leftarrow \{H \diamond f_{\text{enc}}(a) \mid H \in \mathcal{H}_{i-1}, a \in \mathcal{A}\};$  // All single action extension at  $i$ -th step
5    $\Phi_i \leftarrow \{\phi(H) \mid H \in \mathcal{H}_i\};$  // score each trajectory
6    $\tilde{\mathcal{H}}_i, \tilde{\Phi}_i \leftarrow \text{top}(B, \text{sorted}(\mathcal{H}_i, \Phi_i));$  // Keep  $B$  highest scoring trajectories
7    $\tilde{\mathcal{H}}_i^0 \leftarrow \tilde{\mathcal{H}}_i;$ 
8   for  $u = 1, \dots, \delta;$  // Predict  $\delta$  observation representations autoregressively
9   do
10     $\tilde{\mathcal{H}}_i^u \leftarrow \{H \diamond f_{\text{seq}}(H) \mid H \in \tilde{\mathcal{H}}_i^{u-1}\}$ 
11  end
12   $\mathcal{H}_i \leftarrow \tilde{\mathcal{H}}_i^\delta;$ 
13 end
14  $\hat{\mathcal{T}} \leftarrow \text{readout}(l, \text{top}(l, \text{sorted}(\tilde{\mathcal{H}}_l, \tilde{\Phi}_l)));$  // Read out last  $l$  actions from the top scoring beam

```

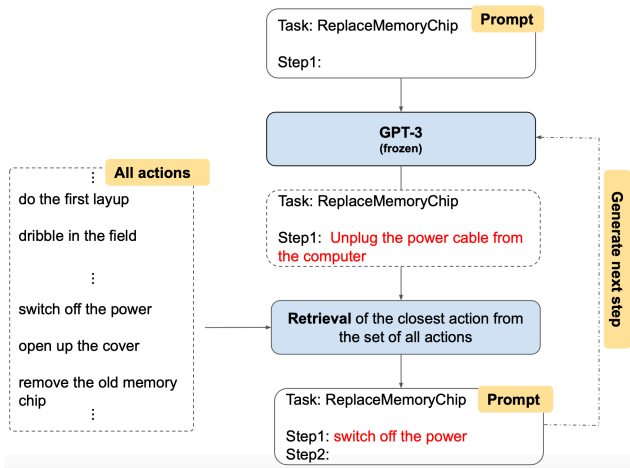


Figure 7: **GPT-3 as a planner based on [23]**. A language-only baseline using GPT-3 on COIN. GPT-3 is prompted autoregressively to generate next action based on the goal and the history of actions taken for the goal.

tory of previously predicted actions (if available) are given as a prompt to the GPT-3 model [7]. Then the next actions for the given goal are generated autoregressively, consistent with other baselines like VLaMP. As can be seen in Fig. 7, this model has 2 stages: 1) next-action generation, and 2) action retrieval. In the next-action generation stage, the model is given the prompt and generates the next action. In the action retrieval stage, the generated next action is compared to all possible actions and the action that has the closest similarity to the generated action, is chosen and placed in the prompt. This step is required since we evaluate the generated plans by comparing with ground truth actions for the goal, where actions belong to a closed set as described in Sec. 3.2. We use *text-davinci-003* backend for generation, and *text-embedding-ada-002* for embedding, which is

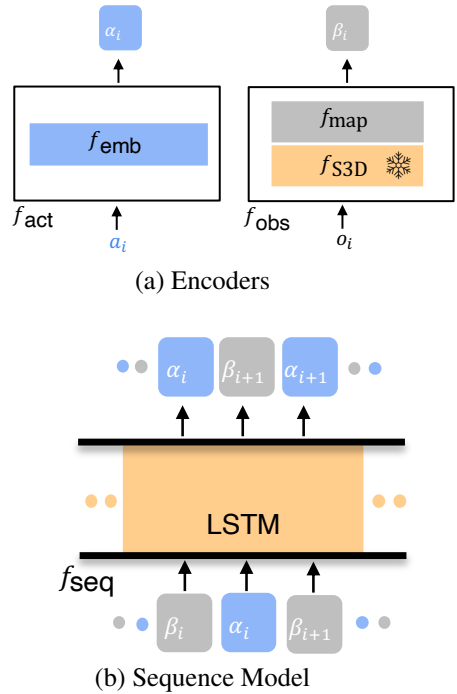


Figure 8: **DDN Implementation**. The method utilizes LSTM in the forecasting module for VPA. Consistent to and analogous of VLaMP’s Fig. 3 in the main paper.

used in combination with cosine similarity to retrieve the closest action. We perform our generation step zero-shot without giving any examples, as GPT3 can already follow the given prompt template, and we only generate one action at the time – in an autoregressive manner.

We sub-sampled the COIN test set, and compared VLaMP with the aforementioned GPT3-based model. In order to make the dataset suitable for language-only instantiation of VPA, we removed videos that had less than four

unique actions. We then evaluate both models on this subset of COIN. As can be seen in Tab. 5, VLaMP surpasses the GPT3-based baseline on all measures.

### B.3. Additional Baseline: ‘Most Probable Action’

We already benchmark two intuitive, heuristic baselines as discussed in Sec. 5.1 and Tab. 3 of the main paper, particularly, denoted as *random* and *random w/ goal*.

Here, we introduce an additional, heuristic baseline called ‘most probable action’. The key insight here is that procedural activities are *highly structured*, i.e., certain actions occur together or occur in a certain order. We bake this into a new baseline that captures this structure through the probability distribution of the next action given the current action.

To this end, the intuitive *most probable action* baseline picks the most probable next action  $a_j|a_i$  from the action set  $\mathcal{A}$ . Akin to random w/ goal baseline, we also evaluate a goal-conditioned most probable action baseline, that uses a goal-specific set of actions  $\mathcal{A}_G \subset \mathcal{A}$  during sampling. Since these most probable baselines, provide a probability distribution over the actions, we can employ beam search (for fairness, with the same beam size same as VLaMP) and pick the highest scoring plan. The results of this are included in Tab. 6 (an expanded version of Tab. 3 from the main paper).

### B.4. On Porting More Baselines

Next, we briefly include some procedural planning approaches and reasons why they cannot be directly leveraged for rigorous and fair evaluation. Wherever possible, we include our best attempts to compare with them.

**PlaTe** [52]: This is similar to DDN, albeit with a Transformer [56] as the sequence model instead of an LSTM [21]. However, unlike DDN (and VLaMP), PlaTe uses separate Transformer-based models for state and action prediction. We adopt an approach that allowed us to tap into this while being consistent and fair in evaluation. Therefore, instead of directly adapting PlaTe for VPA as we did with DDN, we provide an ablation on VLaMP, which uses a Transformer trained from scratch as the sequence model (row **R** in Tab. 4).

**P3IV** [65]: This employs a significantly different modeling framework compared to DDN and PlaTe. Specifically, P3IV leverages a memory-augmented transformer as the sequence model and a probabilistic generative model to capture the noise and variability in predicted sequences. The authors report significant performance gain on the task of procedure planning, over DDN and PlaTe. However, P3IV relies on the visual observation of the goal already completed, even at inference time. This is necessary to condition their generative model towards encoding multiple plans from start to goal. Since P3IV needs the observations of

goal completed, it is incompatible to the motivation and the very premise of VPA.

## C. Inference for VPA (Alg. 1)

In order to predict a sequence of next actions, we run the sequence model, autoregressively to predict both the action and observation tokens, with beam search on the action sequence. The inference algorithm is detailed in Algo. 1. We first encode the history into a sequence of representations  $H_k$  as described in Sec. 4.3, and initialize our set of encoder trajectories  $\mathcal{H}_0$  using this single representation trajectory (line 1 in Alg. 1). Then we start the inference procedure that runs for  $l$  steps (line 2). At each step  $i$  we first infer the next action and then also predict the representations for the observation that follows it. To do the former, each representation trajectory in  $\mathcal{H}_i$  is extended with the representations of each action in the action set  $\mathcal{A}$  (line 4). At this point, if, for instance,  $\mathcal{H}_{i-1}$  had  $n$  trajectories, then after line 4,  $\mathcal{H}_i$  will have  $n \times |\mathcal{A}|$  trajectories. This is a temporary blow-up— at line 6, we score all  $n \times |\mathcal{A}|$  trajectories and keep only top  $B$  trajectories. Here, to balance diversity, we keep no more than  $b$  trajectories with exactly same history. The parameter  $b$  is usually referred to as *per node beam size*. Once we have  $B$  trajectories in  $\mathcal{H}_i$ , we auto-regressively predict the next  $\delta$  tokens corresponding to the next observation, thus completing one out of the  $l$  steps of inference. This process is repeated  $l$  times to generate a plan consisting of  $l$  actions. We make this process efficient by storing the hidden state of the transformer and limiting the forward pass only on the new representations at each step. This is a common practice for transformer based models in NLP. Due to beam search, the inference process is slower than training as shown in Tab. 7.

## D. VLaMP Training (Tab. 7)

Unlike inference where a video with  $K$  steps results into  $K - 4$  examples, during training, like with language model pre-training, we use a single forward pass to compute loss for all tokens. Moreover, inference also uses beam search making it more memory intensive. Thus, the training is much faster and cheaper as compared to the inference. The details of the compute used for each training and inference run is shown in Tab. 7.

## E. Comparison to Ego4D LTA Benchmark

In prior work, Ego4D’s Long-term Action Anticipation [19] benchmark task is likely the most relevant to VPA. Hence, we dedicate a discussion of similarities and contrasts. We hope this helps the reader accurately place these two tasks in our community’s diverse research goals and directions.

Dataset	Method	$l = 1$		$l = 3$		$l = 4$		
		nAcc	SR	mAcc	mIOU	SR	mAcc	mIOU
CrossTask	Random	0.9 ± 0.0	0.0 ± 0.0	0.9 ± 0.0	1.5 ± 0.0	0.0 ± 0.0	0.9 ± 0.0	1.9 ± 0.0
	Random w/ goal	13.2 ± 0.2	0.3 ± 0.0	13.4 ± 0.0	23.6 ± 0.1	0.0 ± 0.0	12.7 ± 0.0	27.8 ± 0.1
	Most probable	10.4 ± 0.0	1.7 ± 0.0	6.1 ± 0.0	9.9 ± 0.0	1.3 ± 0.0	5.5 ± 0.0	13.9 ± 0.0
	Most probable w/ goal	12.4 ± 0.0	2.4 ± 0.0	8.9 ± 0.0	15.5 ± 0.0	1.5 ± 0.0	7.9 ± 0.0	20.5 ± 0.0
	DDN [8]	33.4 ± 0.5	6.8 ± 0.3	25.8 ± 0.5	35.2 ± 0.6	3.6 ± 0.2	24.1 ± 0.4	37.0 ± 0.4
	VLaMP (ours)	<b>50.6 ± 1.4</b>	<b>10.3 ± 0.4</b>	<b>35.3 ± 1.1</b>	<b>44.0 ± 1.0</b>	<b>4.4 ± 0.2</b>	<b>31.7 ± 1.0</b>	<b>43.4 ± 0.9</b>
COIN	Random	0.1 ± 0.0	0.0 ± 0.0	0.1 ± 0.0	0.2 ± 0.0	0.0 ± 0.0	0.1 ± 0.0	0.2 ± 0.0
	Random w/ goal	24.5 ± 0.2	1.7 ± 0.0	21.4 ± 0.1	42.7 ± 0.1	0.3 ± 0.0	20.1 ± 0.1	47.7 ± 0.1
	Most probable	0.7 ± 0.0	1.6 ± 0.0	4.3 ± 0.0	6.8 ± 0.0	1.6 ± 0.0	8.2 ± 0.0	15.3 ± 0.0
	Most probable w/ goal	23.9 ± 0.0	10.9 ± 0.0	18.0 ± 0.0	24.9 ± 0.0	9.1 ± 0.0	16.3 ± 0.0	32.2 ± 0.0
	DDN [8]	29.3 ± 0.3	10.1 ± 0.4	22.3 ± 0.4	32.2 ± 0.6	7.0 ± 0.3	21.0 ± 0.4	37.3 ± 0.3
	VLaMP (ours)	<b>45.2 ± 0.8</b>	<b>18.3 ± 0.1</b>	<b>39.2 ± 0.3</b>	<b>56.6 ± 0.5</b>	<b>9.0 ± 0.3</b>	<b>35.2 ± 0.2</b>	<b>54.2 ± 0.5</b>

Table 6: **Expanded version of Tab. 3.** The mean ± std. error of mean for various planning metrics obtained using 5 runs with different random seed are shown for VLaMP and various baselines. Note that the action history and observations are provided using the output of the action segmentation model and hence are noisy compared to the ground truth history.

	beam size ( $B$ )	per node beam size ( $b$ )	GPU	GPU memory	Num GPUs (inference)	Num GPUs (training)	Avg. time (training)	Avg. time (inference)	batch size (training)
CrossTask	10	3	NVIDIA A100	80GB	1 GPU/model	1 GPU/model	2 s/batch	7.4 s/example	4
COIN	3	3	NVIDIA A100	80GB	3 GPUs/model	1 GPU/model	2 s/batch	6.1 s/example	4

Table 7: Hyperparameters and compute information for VLaMP.

Consistent to VPA, LTA also focuses on predicting a sequence of future actions given prior visual context for free-form human interaction. Unlike LTA, VPA specifically entails *goal-oriented* activities and indeed a natural language goal prompt is of key importance to the definition of VPA. So while the forecasting suite in Ego4D aspires to understand human motion, we are instead keen to create assistive agents that can interact and assist humans in their tasks.

Since LTA does not allow access or model the user’s goal, recent approaches for LTA including the winning model for Ego4D 2022 LTA challenge – ICVAE [37] have to go via an additional step of inferring the intention of the user. This provides more impetus to our goal-conditioned and human-assistive design choice and motivation for VPA. This is empirically backed as well, as we show in ablation (row 1 in Tab. 4) – goal-conditioning is crucial for VPA.

## I. Additional Quantitative Results<sup>10</sup>

**Most probable action baseline.** As shown in Table 6, the performance of our intuitive heuristic baselines – *most probable action w/ goal*, and *Random w/ goal* (i.e. the baselines with actions restricted to the set of actions seen with the corresponding goal) is quite high for COIN dataset. We find that this is due to the relatively small cardinality of the

action set for goals in COIN, i.e. the average size of  $\mathcal{A}_G$  for different  $G$ s.

**Action distribution analysis.** We dig deeper into the above finding in Fig. 9(a). Particularly, we plot the distribution of number of actions  $|\mathcal{A}_G|$  w.r.t  $G$  and find them to be quite different in COIN vs. CrossTask. Here,  $|\mathcal{A}_G| \in \mathcal{A}$  represents the set of goal-specific actions from the larger set of actions  $\mathcal{A}$  for each dataset. Specifically, the average size of  $\mathcal{A}_G$  is 4.3 and 7.3, respectively for COIN and CrossTask, reducing the difficulty of VPA in COIN. Also, notice the long-tails in the distribution of CrossTask, making it even more challenging.

**Zooming into the tails and higher errors.** In Fig. 9(b), we plot the number of steps in the history ( $k$ ) vs. the mean accuracy (mAcc), averaged across all goals in CrossTask on VPA. We find that plan generation with *longer history leads to higher errors* as well as higher variance in performance. We believe this trend emerges due to *two reasons*.

*First*, the presence of repetitive steps in certain goals is high in longer history. Moreover, we find that longer the history the wider is the space of possible plans (intuitively, multiple modes exist in the plan distribution landscape), which lead to higher variance. We illustrate this in Fig. 10 with a qualitative result, for the example goal of ‘making kimchi fried rice’, the action ‘stir mixture’ repeatedly occurs between various actions involving the addition

<sup>10</sup>We number this section as I, to be consistent with the main paper references.

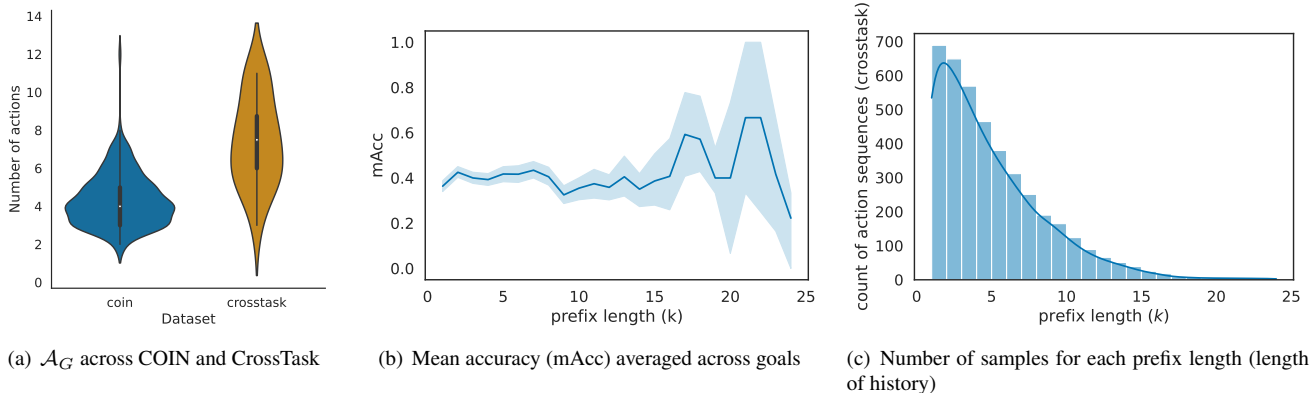


Figure 9: **Zooming into the tails.** (a) Average size of goal-specific action set  $\mathcal{A}_G$  across COIN and CrossTask datasets. COIN has a relatively smaller mean than CrossTask, which reduces the difficulty of VPA on COIN. (b) Mean accuracy (mAcc) vs. the history length  $k$  of various goals from CrossTask. Interestingly, plan generation for goals with longer history is difficult and prone to higher errors as reflected in the mAcc (reasoning included in Sec. I). (c) Longer sequences are also less frequent in the dataset. This contributes to high variance in performance for such sequences.

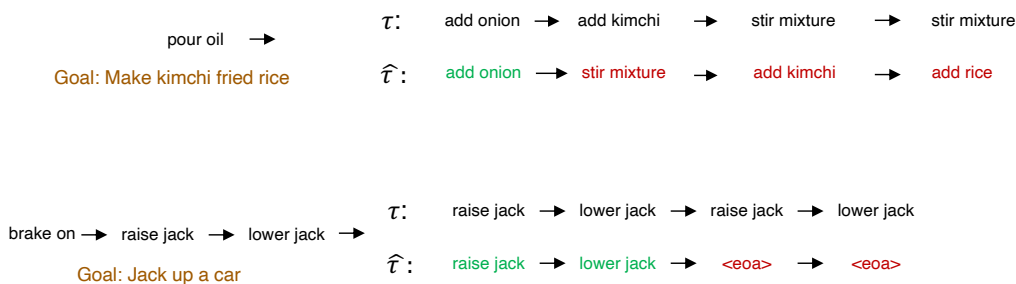


Figure 10: **Qualitative Error Analysis for VPA.** Ground truth plan  $\mathcal{T}$  and the predicted plan  $\hat{\mathcal{T}}$  by VLaMP for the goal prompt of “making kimchi fried rice” (top) and “Jack up the car” (bottom). Errors made by VLaMP can be attributed to *repetitions in actions*. Details are included in Sec. I. Briefly, (1) uncertainty in the number of times actions are repeated and (2) existence of equivalent plans for achieving the same goal, are contribute heavily to the errors for VPA. In the top, note the action ‘stir mixture’ is repeated consecutively in the ground truth, but the model predicts it only once. Moreover, both the ground truth and the predicted plans have correct steps for adding kimchi and onion but their *order is different*. Similar repetitions result into errors for the goal of jacking up the car.

of ingredients like onion, kimchi, rice, etc. However, the number of times *stir mixture* occurs varies sporadically. For instance, for the ground truth plan in the first example in Fig. 10, the ‘stir mixture’ is missing between ‘add onion’ and ‘add kimchi’, but occurs twice after ‘add kimchi’, before adding other ingredients. Due to this sporadic variability, the predicted plan gets IoU of 75% on this example, but mAcc and SR of 25% and 0, respectively. Another common source of errors is repetition of sub-sequences of actions depending on the visual signal in the ground truth. Specifically, as seen in the second example in Fig. 10, which shows an action trajectory for the goal of ‘jack up a car’, the sub-sequence (‘raise jack’, ‘lower jack’), is repeated three times. In this example, the repetition is due to overshooting

the target height of the raised car. However, for a planning model, that only sees the visual input till  $k = 3$  or time  $t_k$ , it is not possible to guess whether the car will overshoot (undershoot, respectively) the target height after the next application of ‘raise jack’ (‘lower jack’, respectively).

Second, as analysed in Fig. 9(c), *longer trajectories are exponentially less frequent* in the dataset – forming the tail of the data distribution of action sequences in the dataset. This also contributes to high variance in performance for such sequences.

**Segmentation errors.** As we note in Sec. 5.3 and Fig. 5 of the main paper, segmentation errors are detrimental for VPA. As the mis-classification error in the segmentation model increases, the difference in the performance of



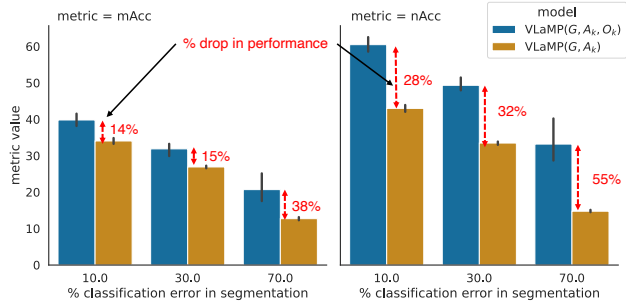


Figure 11: **Effect of segmentation errors.** The figure zooms in on two metrics mAcc and nAcc from Figure 5. As the classification error in segmentation, which is shown along the x-axis increases, the performance gap between the model with access to observation history VLaMP ( $G, A_k, O_k$ ) and that with access only to the action history VLaMP ( $G, A_k$ ) increases.

VLaMP ( $G, A_k, O_k$ ), i.e. the model with access to observation history, and VLaMP ( $G, A_k$ ), the model working only on the action history, increases. A detailed version of Fig. 5 is included in Fig. 11 with a focus on mean accuracy (mAcc) and next-step accuracy (nAcc)<sup>11</sup>. Moreover, since the observation history has higher influence on predicting the immediate next action as discussed in Sec. 5.3, the performance drop due to segmentation classification error is higher in nAcc as compared to mAcc.

<sup>11</sup>Refer to the definitions of metrics in Sec. 3.2