Training-free Online Video Step Grounding

Luca Zanella[1] **Massimiliano Mancini**[1] **Yiming Wang**[2]
Alessio Tonioni[3] **Elisa Ricci**[1,2]
[1]University of Trento [2]Fondazione Bruno Kessler [3]Google
https://lucazanella.github.io/baglm/

Abstract

Given a task and a set of steps composing it, Video Step Grounding (VSG) aims to detect which steps are performed in a video. Standard approaches for this task require a labeled training set (*e.g.*, with step-level annotations or narrations), which may be costly to collect. Moreover, they process the full video offline, limiting their applications for scenarios requiring online decisions. Thus, in this work, we explore how to perform VSG *online* and *without training*. We achieve this by exploiting the zero-shot capabilities of recent Large Multimodal Models (LMMs). In particular, we use LMMs to predict the step associated with a restricted set of frames, without access to the whole video. We show that this online strategy without task-specific tuning outperforms offline and training-based models. Motivated by this finding, we develop Bayesian Grounding with Large Multimodal Models (BAGLM), further injecting knowledge of past frames into the LMM-based predictions. BAGLM exploits Bayesian filtering principles, modeling step transitions via (i) a dependency matrix extracted through large language models and (ii) an estimation of step progress. Experiments on three datasets show superior performance of BAGLM over state-of-the-art training-based offline methods.

1 Introduction

Grounding procedural steps in videos is crucial for enabling machines to follow along and assist humans in complex tasks like cooking a recipe, assembling furniture, or performing maintenance work. This ability is particularly valuable for real-time procedural guidance in AR/XR applications, where recognizing task progress allows users wearing headsets or smart glasses to receive timely, step-specific instructions. Specifically, the task of Video Step Grounding (VSG) takes as input a list of procedural steps extracted from an instructional article (*e.g.*, a recipe or how-to guide), and a video performing the same task, with the goal of identifying which of the steps are performed in the video.

Existing VSG approaches align procedural steps descriptions with their corresponding video frames [4, 9, 14, 18]. However, these strategies face two key limitations. First, they need a training set, entailing the cost of collecting (and potentially annotating) it. Moreover, a training set could bias models toward the specific videos and procedural tasks that are depicted, limiting their generalization capability. Second, they assume offline processing, where the entire video is available ahead of time. This makes them unsuitable for real-world applications processing a live video stream.

To overcome these limitations, we explore how to address VSG *online* and *without training*, *i.e.*, operating on a streaming video (Fig. 1). This objective is challenging as it requires performing predictions with partial evidence (*i.e.*, having access to only a subset of the video frames) and without the possibility of extracting task-specific representations that would be typically learned from a dedicated training set. In this scenario, we explore the zero-shot capabilities of powerful Large Multimodal Models (LMMs) [2, 13, 6] for VSG. Specifically, given the set of steps, we prompt LMMs to predict the step corresponding to the current video segment. Surprisingly, models such

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

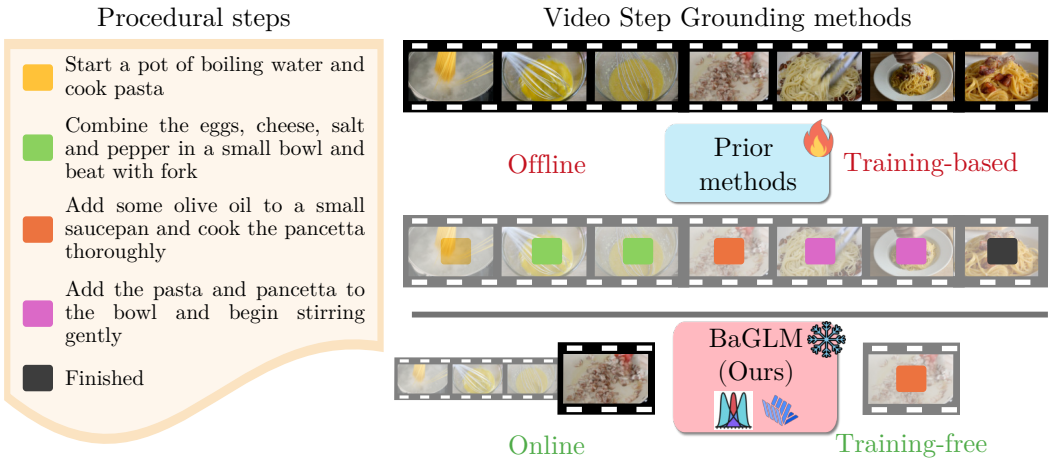

Figure 1: We tackle **Video Step Grounding** with **BAGLM**, a *training-free* approach which combines Bayesian filtering with Large Multimodal Models to enable *online* inference over video streams.

as InternVL [6] could already surpass state-of-the-art training-based methods [14, 18], despite only having access to a single segment at a time. This highlights a strong potential for addressing VSG without training.

Building on the zero-shot capabilities of LMMs, we show how performance can be further improved by modeling temporal dependencies across steps and leveraging past information. To this end, we propose Bayesian Grounding with Large Multimodal Models (BAGLM), a training-free approach for VSG that combines Bayesian probabilistic modeling with LMMs. We harness a Large Language Model (LLM) to estimate a dependency matrix, capturing whether one step is a prerequisite of another. From this matrix, we compute transition probabilities to each step in the current video segment (*i.e.*, the *predict* step of the Bayesian filter). This prior refines the LMM direct prediction (*i.e.*, the *update* step), injecting past temporal knowledge into the model's output. The transition model is updated over time, following the progress of each step estimated by the LMM. On three publicly available datasets (HT-Step [1], CrossTask [36], Ego4D Goal-Step [27]), BAGLM consistently outperforms existing methods with significant margins, achieving state-of-the-art results on this challenging task.

We make four key **contributions**: ① We present the first study to address VSG in an online, training-free setting, eliminating the need for data collection and better aligning with practical application needs; ② We show that the zero-shot LMMs can surpass specialized, training-based methods, revealing their potential for addressing VSG. ③ We propose a method, BAGLM, which incorporates priors from past video frames into LMMs through Bayesian filtering, modeling the temporal dependencies across steps via LLM queries; ④ We extensively evaluate BAGLM on three datasets, showing that it outperforms state-of-the-art offline methods.

2 Related work

Video Step Grounding. Earlier works in VSG adopted weakly-supervised learning approaches. For instance, Zhukov *et al.* [36] proposed to learn a model from instructional narrations and a list of steps derived from temporal constraints, sharing components across tasks with similar actions or objects. Han *et al.* [9] proposed a co-training framework that combines a Temporal Alignment Network (TAN) with a dual-encoder architecture, predicting step boundaries by aligning videos and narrations, using pseudo-labels derived from cross-modal agreement. VINA [18] considered step descriptions from WikiHow [29] and learned to temporally ground them in videos without manual supervision. Recent methods have increasingly leveraged language models and large-scale pretraining. MPTVA [4] introduced a multi-pathway alignment strategy using LLM-filtered narration summaries and multiple sources of alignment, merging them to create robust pseudo-labels for training. NaSVA [14] addressed multi-sentence grounding by leveraging LLMs to transform noisy ASR transcripts into procedural steps, aligning them with video content using a narration-based similarity score.

Together, these approaches illustrate the progress from weakly supervised models leveraging narrations to more sophisticated ones that integrate language models, multimodal alignment, and robust pseudo-labeling. However, most methods still rely on extensive training, domain-specific fine-tuning, and access to the full video, assumptions that limit their real-world applicability. To the best of our knowledge, BAGLM is the first training-free online solution that addresses these limitations.

Video-Language Alignment refers to the task of measuring the semantic consistency between a video and its corresponding textual description. Early approaches [10, 26] tackled this by relying on the cosine similarity between video frames and captions within the embedding space of CLIP [23]. However, these methods are inherently limited by the well-known shortcoming of CLIP, *i.e.*, by its inability to effectively capture temporal dynamics in text descriptions. As a result, recent works have shifted toward leveraging LMMs [15, 30, 31, 12, 33] and adopting metrics such as VQAScore [16], which are obtained from video question answering to better account for the temporal dimension.

Building on these recent studies, we propose to employ an LMM to assess the alignment between instructional steps and temporal video segments, requiring fine-grained video understanding. VSG is particularly challenging because key steps in instructional tasks often involve similar objects or scenes. For instance, "inserting a screw" or "aligning parts" may look similar in tasks like "Assemble a chair" and "Fix a table", making them hard to distinguish without nuanced semantic understanding.

3 On using Large Multimodal Models for Video Step Grounding

Large Multimodal Models are powerful pretrained models that have demonstrated impressive zero-shot performance on a wide variety of tasks without further tuning. As previous solutions for VSG are training-based, we wonder whether off-the-shelf LMMs can address VSG. This section begins by providing a formal definition of the VSG task, clearly distinguishing between its offline and online settings (Sec. 3.1). We then present the findings of our preliminary empirical investigation into the use of LMMs for addressing VSG, along with key insights gained from this study (Sec. 3.2).

3.1 Video Step Grounding

Given a video of a task composed of a series of actions, video step grounding aims to detect which actions (or steps) appear in the video. Formally, let us denote the set of steps composing a task as $\mathcal{A} = \{a_i\}_{i=1}^{K}$, where each step $a_i \in \mathcal{A}$ is expressed in the natural language space $\mathcal{L}$, and K is the number of steps. Moreover, let us denote with $\mathbf{V}$ a video in the space $\mathcal{V}$, split into T non-overlapping segments $\mathcal{S} = \{\mathbf{S}_t\}_{t=1}^{T}$. VSG aims to identify which steps in $\mathcal{A}$ are shown in the video.

The offline setting of VSG, as addressed by previous works [14, 18], assumes access to the whole video (*i.e.*, the full set $\mathcal{S}$) when performing the task. In this work, we focus on *online* VSG, assuming a video stream where segments arrive one after the other, and we perform the task having only access to the current segment and the segments preceding it, *i.e.*, $\mathcal{S}_{1:t} = \{\mathbf{S}_1, \dots, \mathbf{S}_t\}$. Thus, online VSG aims to predict whether a segment $\mathbf{S}_t \in \mathcal{S}$ depicts a step $a_i \in \mathcal{A}$, given only the previous video segments $\mathcal{S}_{1:t}$. While we do not exploit this possibility, a model may also use the current segment to update predictions on past ones, contrary to the single prediction of the offline setting.

3.2 Large Multimodal Models are strong baselines for VSG

Let us define a large multimodal model $f_{\texttt{LMM}}$ via three elements: the visual encoder $f_{\texttt{vid}}$, the text encoder $f_{\texttt{txt}}$, and a text decoder $f_{\texttt{dec}}$. The encoders map their respective inputs into a shared d-dimensional embedding space, *i.e.*, $f_{\texttt{vid}} : \mathcal{V} \to \mathbb{R}^d$ and $f_{\texttt{txt}} : \mathcal{L} \to \mathbb{R}^d$. The decoder maps the visual and textual inputs into a probability simplex $\Delta^{|\mathcal{W}|}$, over the LLM vocabulary $\mathcal{W}$[1], *i.e.*, $f_{\texttt{dec}} : \mathbb{R}^d \times \mathbb{R}^d \to \Delta^{|\mathcal{W}|}$. The next token is sampled from this probability vector.

Preliminary experiment. We propose to frame the problem of VSG as a multi-choice question answering task where the LMM, prompted with the current video segment and all possible steps, has to predict as answer either one of them or "none". Formally, given the current segment $\mathbf{S}_t$, we prompt the LMM $f_{\texttt{LMM}}$ using information regarding the task and step, obtaining the score for a step a_i as:

$$f_{\texttt{LMM}}(\mathbf{S}_t, \pi_{\texttt{VSG}})[i] = f_{\texttt{dec}}\left(f_{\texttt{vid}}(\mathbf{S}_t), f_{\texttt{txt}}(\pi_{\texttt{VSG}})\right)[i] \tag{1}$$

[1]For simplicity, we omit the words' tokenization, assuming text prompts and videos are encoded equally.

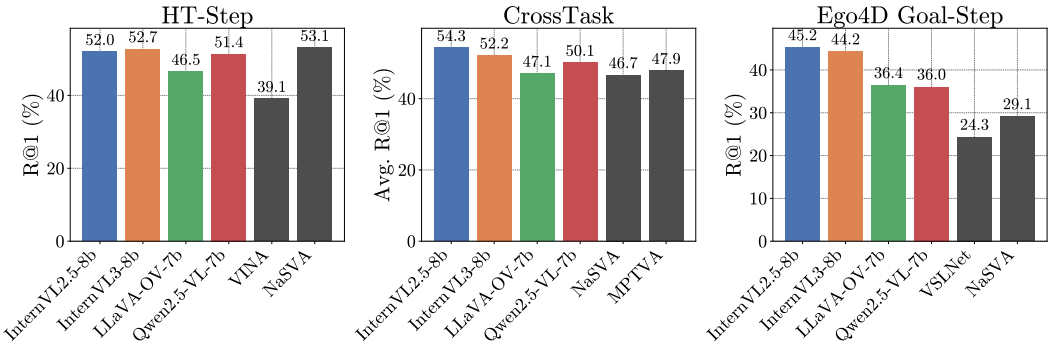

Figure 2: Comparison of VSG performance on HT-Step, CrossTask, and Ego4D Goal-Step datasets, prompting LMMs with step options and video segments in an online fashion. For reference, we also show the performance of the top two performing methods from the state of the art (dark bars).

where $\pi_{\texttt{VSG}}$ is the task prompt (see details in Appendix) and $f_{\texttt{dec}}(\cdot,\cdot)[i]$ denotes the probability that the next predicted token is i (corresponding to step a_i), normalizing the scores across the multi-choice options. Note that, to account for no-step occurring, we include an additional option "none of the above" ($f_{\texttt{dec}}(\cdot,\cdot)[K+1]$). By normalizing the LMMs' probabilities over each choice, we map the segment into a probability simplex over the steps and the "none" option, *i.e.*, $f^{\texttt{VSG}}_{\texttt{LMM}} : \mathcal{V} \times \mathcal{A} \rightarrow \Delta^{K+1}$.

Datasets & metrics. We evaluate methods on three public datasets: *CrossTask* [36], *HT-Step* [1], and *Ego4D Goal-Step* [27]. *HT-Step* is a benchmark for procedural step grounding [1], where the goal is to align steps from an instructional article with an input how-to video. The dataset provides two types of test sets: one for seen activities and another for unseen activities. The seen validation and test splits follow [18], each containing 600 videos in total, with 5 videos per activity across 120 activities. We evaluate with the validation set of seen classes, as the evaluation server hosting the test sets for the seen and unseen classes is unavailable.

CrossTask is an established instructional video benchmark for zero-shot step localization [36]. It contains about 4.8k instructional videos, covering 18 primary tasks and 65 related tasks. Only videos in the primary tasks are annotated as steps with temporal segments from a predefined taxonomy. We follow the same evaluation set as indicated in [14], using videos from primary tasks.

Ego4D Goal-Step [27], a subset of the Ego4D [8], includes 851 videos averaging 26 minutes in length. Unlike CrossTask and HT-Step, Ego4D videos are collected without predefined tasks. Annotators label them hierarchically, first identifying goals (*e.g.*, *Makes the bread*), then steps (*e.g.*, *Prepares the bread*), and finally substeps (*e.g.*, *weigh the dough*). We evaluate on its validation split.

For both HT-Step and CrossTask, we follow the standard evaluation protocol [14, 4], providing for each video the full set of steps for its task as multiple-choice options in the prompt. On average, each task includes about 10 steps in HT-Step and 7.5 in CrossTask. For Ego4D Goal-Step, we use step-level descriptions only (excluding substeps) and apply text normalization with `spaCy`[2]: we lowercase, lemmatize (preserving plural nouns and verbal adjectives), and normalize whitespace and punctuation. After preprocessing, the average number of steps per video is 17.

Following [14, 4], we report Recall@1 (R@1) on HT-Step and Ego4D Goal-Step, measuring whether the top-scoring timestamp for each step falls within the ground-truth interval. For CrossTask, we report Average Recall@1 (Avg.R@1), computed by averaging per-task R@1 across all primary tasks.

Discussion. Fig. 2 shows the results on the three datasets, using four LMMs with strong performance in video understanding benchmarks [22]: LLAVA-OneVision-Qwen2-7B [13], Qwen2.5-VL-7B-Instruct [2], InternVL2.5-8B [6], and InternVL3-8B [35]. For reference, we also include the top two state-of-the-art methods on each dataset, considering both in-domain ones, *i.e.*, trained and evaluated on the same dataset (VINA [18] and NaSVA [14] on HT-Step, VSLNet [34] on Ego4D Goal-Step) and out-of-domain ones (*i.e.*, NaSVA on CrossTask and Ego4D Goal-Step, MPTVA [4] on CrossTask). Among the LMMs, InternVL2.5-8B scores the best performance on CrossTask and Ego4D Goal-Step, outperforming MPTVA by 6.4% and NaSVA by 16.1%, respectively. However, on HT-Step, NaSVA

[2]We use the model available at `https://spacy.io/models/en#en_core_web_sm`

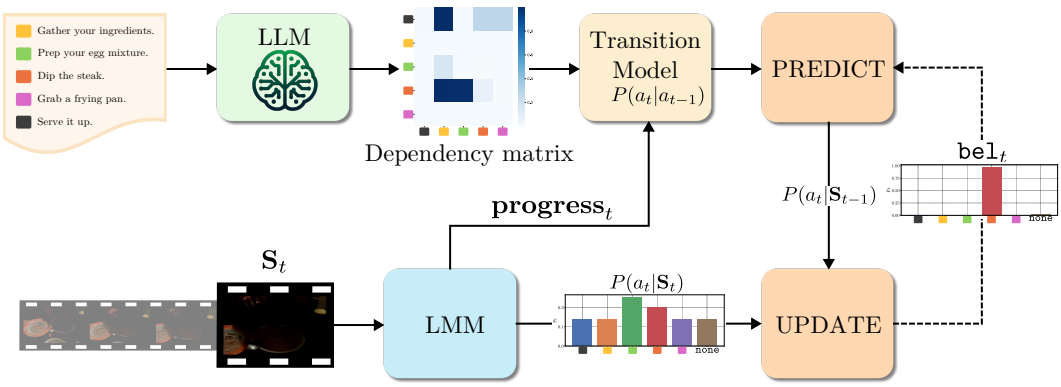

Figure 3: **Overview of BAGLM.** Given a sequence of steps, an LLM is used to estimate a dependency matrix among them. This matrix is used to compute step transition probabilities employed during the **predict** step of a Bayesian filter. As the video progresses, the transition model is updated using estimates of each step's progress from an LMM. The **update** step of the filter merges this with the predictions from the LMM, refining the output.

slightly surpasses InternVL2.5-8B (53.1 vs. 52.0). Overall, the LMMs outperform prior methods on CrossTask and Ego4D Goal-Step, while showing comparable performance on HT-Step.

Remarks. Considering the lack of task-specific tuning, these results demonstrate that zero-shot LMMs perform surprisingly well on VSG, accessing only the current segment. A natural question is whether we can (i) inject information from past segments into LMM's predictions, refining them, while (ii) maintaining the zero-shot, training-free advantages of LMMs. In the following, we explore how we can achieve this by drawing inspiration from Bayesian filtering principles.

4 Bayesian Grounding with Large Multimodal Models

In Sec. 3 we have shown that LMMs are effective at VSG without any task-specific tuning. However, they act without memory of past knowledge, performing step prediction by only looking at the current segment. Therefore, uncertain predictions (*e.g.*, due to the segment acquisition) cannot benefit from past evidence (*e.g.*, step performed in the previous segment), leading to potential model mistakes.

Formally, Eq. (1) provides an estimate for $P(\mathtt{A} = a_t|\mathbf{S}_t)$, where $\mathtt{A}$ is a discrete random variable taking values in the set $\mathcal{A} \cup \mathtt{none}$ with $\mathtt{none}$ denoting any action outside the set of steps. Differently, we would like to perform step prediction conditioned on all previous segments, estimating $P(\mathtt{A} = a_t|\mathcal{S}_{[1:t]})$,without the need of storing the whole history. Inspired by Bayesian filtering, a probabilistic technique addressing sequence modeling from past observations [5], we propose *Bayesian Grounding with Large Multimodal Models (*BAGLM*)*. BAGLM (Fig. 3) estimates the transition probabilities across steps through the step dependencies estimated by an LLM, to refine LMMs' predictions.

In the following, we will first present the generic formulation of Bayesian filtering in Sec. 4.1, and how we revisit the predict (Sec. 4.2) and update (Sec. 4.3) steps for VSG using LMMs and LLMs.

4.1 Bayesian filtering

Let $\mathbf{x}$ be a state of a process, $\mathbf{X}_t$ be its corresponding random variable at time t, and $\mathbf{z}_i$ be the observation at time $i \leq t$. The goal of Bayesian filtering [5] is to compute the posterior:

$$\mathtt{bel}_t(\mathbf{x}) = P(\mathbf{X}_t = \mathbf{x}|\mathbf{z}_1, \cdots, \mathbf{z}_t),$$

via two essential steps: the **predict** step first computes a prior over the possible predictions using past estimations; the **update** step then estimates the prior with the current observations. In the following, we will denote $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \cdots, \mathbf{z}_t\}$ for simplicity. Using the chain rule, we can write the posterior as:

$$\mathtt{bel}_t(\mathbf{x}) = P(\mathbf{X}_t = \mathbf{x}|\mathbf{z}_{1:t}) = \frac{1}{P(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \cdot P(\mathbf{z}_t|\mathbf{X}_t = \mathbf{x}, \mathbf{z}_{1:t-1}) \cdot P(\mathbf{X}_t = \mathbf{x}|\mathbf{z}_{1:t-1}), \quad (2)$$

where the first term is a normalization factor, the second is the likelihood of the current observation given the past ones and the current state, and the last is the prior over the states from the observations.

To further simplify Eq. (2), we consider two *assumptions*: (i) we have a hidden Markov observation model, and thus $P(\mathbf{z}_t|\mathbf{X}_t = \mathbf{x}, \mathbf{z}_{1:t-1}) = P(\mathbf{z}_t|\mathbf{X}_t = \mathbf{x})$; (ii) we have an initial prior over the states independent from the first observation, *i.e.*, $P(\mathbf{X}_0 = \mathbf{x}|\mathbf{z}_1) = P(\mathbf{X}_0 = \mathbf{x})$.

Adding the first assumption to the second term and using the Chapman-Kolmogorov equation on $P(\mathbf{X}_t = \mathbf{x}|\mathbf{z}_{1:t-1})$, we obtain:

$$\texttt{bel}_t(\mathbf{x}) = \overbrace{\underbrace{\frac{1}{P(\mathbf{z}_t|\mathbf{z}_{1:t-1})}}_{\text{normalization factor}} \cdot \underbrace{P(\mathbf{z}_t|\mathbf{X}_t = \mathbf{x})}_{\text{likelihood}}}^{\text{update product}} \cdot \overbrace{\sum_{\mathbf{x}_i \in \mathcal{X}} \underbrace{P(\mathbf{X}_t = \mathbf{x}|\mathbf{X}_{t-1} = \mathbf{x}_i)}_{\text{transition model}} \cdot \underbrace{\texttt{bel}_{t-1}(\mathbf{x}_i)}_{\text{accumulated belief}}}^{\text{predict step}}, \quad (3)$$

where $\mathcal{X}$ is the set of possible states. The second term corresponds to the **predict** step, computing an estimate of the current state using the prior belief. The transition model describes the likelihood of a state given the previous one, while accumulated belief denotes the likelihood of the previous state as recursively accumulated via Eq. (3). The first term refers to the **update** step, where the predicted state probability is multiplied by the likelihood and normalization factor to obtain the final estimate.

A Bayesian filtering view on VSG. To adapt Eq. (3) to VSG we must define our states and observations. The state is what we want to estimate, *i.e.*, the step a in the current segment. The observation is the input we receive from the environment: the segment $\mathbf{S}$ itself. Thus, we obtain the update step as:

$$\texttt{bel}_t(a) = P(\texttt{A}_t = a|\mathcal{S}_{1:t}) = \frac{P(\mathbf{S}_t|\texttt{A}_t = a, \mathcal{S}_{1:t-1})}{P(\mathbf{S}_t|\mathcal{S}_{1:t-1})} \cdot \sum_{a_i \in \mathcal{A}} P(\texttt{A}_t = a|\texttt{A}_{t-1} = a_i) \cdot \texttt{bel}_{t-1}(a_i), \quad (4)$$

where $\texttt{A}_j$ is a random variable over the possible steps for the j^{th} segment. We keep the original model's assumptions: an initial prior over steps independent of the first segment, and conditional independence of the current segment given the step (which holds for step-level semantics). In the following, we detail the implementation of each component in Eq. (4).

4.2 PREDICT: modeling dependencies among steps via language and progress priors

A peculiarity of VSG is that actions depend on each other: these dependencies provide priors on the actions performed in future segments, allowing us to build a transition model, needed in Eq. (4). We exploit the internal knowledge of an LLM to estimate such dependencies. Specifically, we query the LLM to identify when a step must be completed before another can occur (*i.e.*, is a prerequisite). As the dependency might be ambiguous, we instruct the LLM to estimate a probability rather than a binary score, resulting in a matrix $\mathbf{D} \in \mathbb{R}^{K \times K}$, where each entry $\mathbf{D}_{i,j}$ is the probability that step a_j is a prerequisite of step a_i. We initialize the transition matrix as $\mathbf{T} = \mathbf{D}^\top$, allowing for self-transitions (*i.e.*, $\mathbf{T}_{i,i} = 1$), and transitions from all steps to those with no prerequisites (*i.e.*, $\mathbf{T}_{i,j} = 1$ if $\sum_{j=1}^{K} \mathbf{T}_{i,j} = 0$), normalizing $\mathbf{T}$ across rows.

Modeling the action progress. The transition matrix $\mathbf{T}$ is static and purely based on the task description, without reflecting how the likelihood of a step evolves during the video. For example, if *boil water* is a prerequisite for *cook pasta* and *boil water* has not yet finished, the video cannot transition to *cook pasta*. Instead, it is more likely to continue showing *boil water* or switch to a parallel steps like *prepare the sauce*. Conversely, once *cook pasta* is completed, it becomes unlikely that the video will return to an earlier step as *boil water*. We therefore adjust $\mathbf{T}$ accounting for both step dependencies and the estimated step progress, introducing two scores: *readiness* and *validity*. Intuitively, a step is ready when its prerequisites are sufficiently complete, while a step is a valid candidate for a segment if its successors have not yet been completed [24].

To achieve this, we query the LMM to infer the execution progress of each step within a video segment. Given a segment $\mathbf{S}$ and the prompt $\pi_{\texttt{prog}}$, we define the estimated progress for step a_i as:

$$\mathbf{progress}_t[i] = \sum_{j=0}^{9} j \cdot f_{\texttt{LMM}}(\mathbf{S}_t, \pi_{\texttt{prog}})[j],$$

where $f_{\texttt{LMM}}(\mathbf{S}_t, \pi_{\texttt{prog}})[j]$ denotes the model's output probabilities over the vocabulary. We treat the model's probability distribution over the tokens $\{0, 1, \ldots, 9\}$, as the distribution over progress levels. From the latter and $\mathbf{D}$, we can compute whether an action is ready to be performed.

Step readiness. For each step a_i, we compute readiness as the weighted maximum progress of its prerequisite steps, *i.e.*,

$$\mathbf{r}_t[i] = \frac{\sum_{j=1}^{K} \mathbf{D}_{i,j} \cdot \max_{\tau<t} \mathbf{progress}_\tau[j]}{\sum_{j=1}^{K} \mathbf{D}_{i,j}}, \tag{5}$$

where we measure the progress of a step as its maximum value across all preceding segments (*i.e.*, $\tau < t$), performing a weighted average across all steps. With Eq. (5), we get high values in case all predecessors of an action have a high progress, and low otherwise.

Step validity. Contrary to the readiness, the step validity is given by:

$$\mathbf{v}_t[i] = \frac{\sum_{j=1}^{K} \mathbf{D}_{j,i} \cdot (1 - \max_{\tau<t} \mathbf{progress}_\tau[j])}{\sum_{j=1}^{K} \mathbf{D}_{j,i}}. \tag{6}$$

This value is high when no successors of a_i in $\mathbf{D}$ have been executed yet, and low otherwise. Finally, we adjust the transition matrix $\mathbf{T}$ by accounting for readiness and validity of each step, *i.e.*,

$$\tilde{\mathbf{T}}_t[i,j] = \frac{\mathbf{T}[i,j] \cdot \mathbf{r}_t[j] \cdot \mathbf{v}_t[j]}{\sum_{k=1}^{K} \mathbf{T}[i,k] \cdot \mathbf{r}_t[k] \cdot \mathbf{v}_t[k]}. \tag{7}$$

Predict step. Exploiting the transition model of Eq. (7), the predict step of Eq. (4) becomes:

$$\texttt{predict}_t(a_i) = \sum_{a_j \in \mathcal{A}} \tilde{\mathbf{T}}_t[j,i] \cdot \texttt{bel}_{t-1}(a_j). \tag{8}$$

4.3 UPDATE: using LMM estimates to re-weigh the belief over steps

In the **update step**, we multiply the step prior for the observation likelihood $P(\mathbf{S}_t|\mathtt{A}_t = a)$ and a normalization factor independent from the steps. However, due the cardinality of $\mathcal{V}$, computing $P(\mathbf{S}_t|a_t, \mathcal{S}_t)$ is intractable. On the other hand, we follow Eq. (1), estimating $(\mathbf{S}_t|\mathtt{A}_t = a)$ directly from the LMM. Formally, we use the Bayes rule and write:

$$P(\mathbf{S}_t|\mathtt{A}_t = a_i) = \frac{P(\mathbf{S}_t)}{P(\mathtt{A}_t = a)} \cdot P(\mathtt{A}_t = a_i|\mathbf{S}_t) = \frac{P(\mathbf{S}_t)}{P(\mathtt{A}_t = a_i)} \cdot f_{\texttt{LMM}}(\mathbf{S}_t, \pi_{\texttt{VSG}})[i], \tag{9}$$

where $P(\mathbf{S}_t)$ is a prior on the segments independent from the steps, $P(\mathtt{A}_t = a_i)$ is a prior over the steps independent from the observation. We replace $P(\mathtt{A}_t = a|\mathbf{S}_t)$ with the LMM prediction. While for $P(\mathtt{A}_t = a_i)$ there are various possible choices, in our approach, we consider the prior to be uniform, ablating this choice in the Appendix.

Final filtering model. Considering the uniform prior over the states and merging Eq. (8) into Eq. (9), we obtain the final belief $\texttt{bel}_t(a_i)$ for step a_i and segment $\mathbf{S}_t$ as:

$$\texttt{bel}_t(a_i) = \frac{1}{\mathcal{Z}} \cdot f_{\texttt{LMM}}(\mathbf{S}_t, \pi_{\texttt{VSG}})[i] \mathcal{Z} \cdot \sum_{a_j \in \mathcal{A}} \tilde{\mathbf{T}}_t[j,i] \cdot \texttt{bel}_{t-1}(a_j), \tag{10}$$

where $\mathcal{Z}$ is a normalization factor containing all elements independent of the specific step a_i.

5 Experiments

In this section, we describe our experimental protocol and present the comparison w.r.t. the state of the art (Sec. 5.1). Finally, we perform a detailed study on BAGLM (Sec. 5.2). We use the same datasets and metrics described in Sec. 3 in our experiments.

Implementation details. Our method is implemented considering InternVL2.5-8B [6] as our LMM, based on the results of Sec. 3. We employ LLaMA3-70B-Instruct [7] as our LLM of choice to derive

Table 1: Comparison between state-of-the-art offline methods and our online method BAGLM.

(a) HT-Step and Ego4D Goal-Step

Method	HT-Step ↑ R@1	Ego4D Goal-Step ↑ R@1
Offline		
VSLNET [34]	-	24.3†
TAN* [9]	30.7	-
VINA [18]	39.1	-
NASVA [14]	53.1†	29.1†
Online		
NASVA [14]	46.1†	24.2†
BAGLM	**57.4**	**43.3**

(b) CrossTask

Method	↑ Avg. R@1
Offline	
Zhukov *et al.* [36]	22.4
HT100M [20]	33.6
VIDEOCLIP [32]	33.9
MCN [3]	35.1
DWSA [25]	35.3
MIL-NCE [19]	40.5
VT-TWINS [11]	40.7
UNIVL [17]	42.0
VINA [18]	44.8
NASVA [14]	46.7
MPTVA [4]	47.9
Online	
BAGLM	**59.8**

our transition model. To test our model, we split videos into sequences of non-overlapping 2-second segments, providing them as input to the LMM one after the other. We ran all experiments on a single NVIDIA H100 64GB GPU, except for LLaMA3-70B-Instruct [7], which required 4 H100 GPUs.

Baselines. We compare our method with several state-of-the-art approaches for VSG: Zhukov *et al.* [36], HT100M [20], VideoCLIP [32], MCN [3], DWSA [25], MIL-NCE [19], VT-TWINS [11], UniVL [17], VINA [18], TAN* [9, 1], NaSVA [14], and MPTVA [4]. We also implement an online variant of NaSVA [14], which introduces causal masking in the transformer encoder's self-attention layers to restrict attention to past segments only. Reported results are taken from the original papers, except for NaSVA on HT-Step and Ego4D Goal-Step, and VSLNet [34] on Ego4D Goal-Step, where we use the authors' released code. These are marked with † in the tables.

All baselines are training-based and offline, except for our online variant of NaSVA, and use HowTo100M [20] as training set or pre-training, except for VSLNet (trained on Ego4D Goal-Step), for Zhukov *et al.* [36] and DSWA [25] (trained on CrossTask) and for MPTVA [4] (trained on a subset of HowTo100M). For CrossTask, VideoCLIP [32] and UniVL [17] perform a further fine-tuning step on CrossTask data.

5.1 Comparison with state-of-the-art methods

Tab. 1a and Tab. 1b report the results of our evaluation in the three considered datasets. Overall, BAGLM, built on top of InternVL2.5-8B, outperforms all offline methods that rely on noisy supervision from narrations without manual annotations. Notably, it outperforms the current state-of-the-art method, NaSVA, by 4.3% on HT-Step, which consists of videos from HowTo100M, the same dataset used for NaSVA's self-training. The improvements of BAGLM over NaSVA are especially significant on CrossTask and Ego4D Goal-Step, with gains of 13.1% and 14.2%, respectively. The margins in CrossTask are remarkable as there exist methods (*i.e.*, VideoCLIP, UniVL) that are specifically fine-tuned for this domain. The same applies to Ego4D Goal-Step. This dataset is particularly challenging for models trained on HowTo100M due to the distribution shift introduced by its egocentric videos. On this dataset, BAGLM also outperforms VSLNet by a significant margin (+19%), despite VSLNet being trained on data from the same domain. Notably, all these results have been achieved in an online setting, with all competitors having access to the whole video, contrary to BAGLM. When evaluated under the same online setting, NaSVA's performance drops to 46.1 R@1 on HT-Step and 24.2 on Ego4D Goal-Step (-7% and -4.9% compared to its offline variant), remaining well below BAGLM (-11.3% and -19.1%). These results further highlight the effectiveness of our approach in the challenging online scenario.

Table 2: Ablation study on the transition model.

Readiness	Validity	HT-Step	CrossTask	Ego4D
		55.9	58.0	42.1
✓		57.0	58.8	42.0
	✓	56.4	58.8	43.1
✓	✓	**57.4**	**59.8**	**43.3**

Table 3: Ablation study on varying the LLM.

Dataset	LLaMA-3.3-70B	GPT-4.1-mini
HT-Step	**57.4**	57.1
CrossTask	59.8	**60.9**
Ego4D	**43.3**	40.6

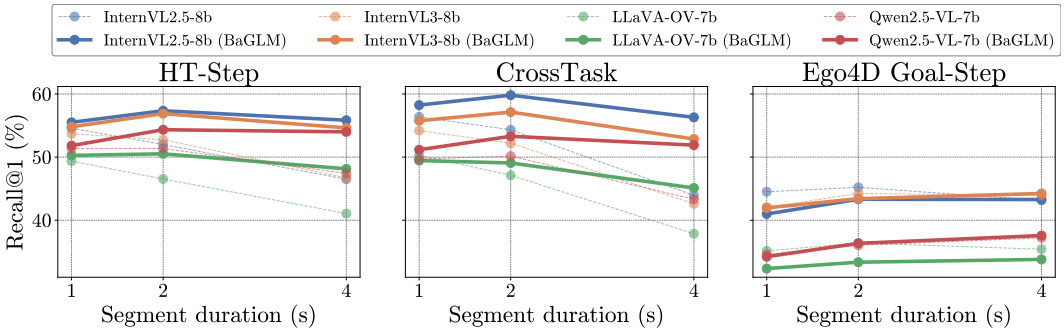

Figure 4: Ablation study on varying the segment duration and on the used LMM.

5.2 Ablation studies

In this section, we analyze the key components of BAGLM, evaluating different configurations of the transition model, different LLMs for generating the dependency matrix, and experiments with oracle step dependencies and progress. Additional design choices are discussed in the Appendix.

Transition Model. The transition model estimates the conditional probability of moving from one step to the next and is used in the predict step of Bayesian filtering, as per Eq. (8). We first evaluate a static transition matrix, initialized from the dependency matrix. We then study the effect of including the readiness score (*i.e.*, if prerequisites are not completed, Eq. (5)), the validity one (*i.e.*, if the step should not be re-executed, Eq. (6)), and both. As shown in Tab. 2, readiness improves the static model by 1.1% and 0.8% on two datasets and performs similarly to the static model on the third (-0.1%). Thus, accounting for the dependency matrix (and if prerequisites are met) tends to improve performance by reducing the score of non-executable actions.

Validity improves the static model by +0.5%, +0.8%, and +1.0% across the three datasets. By considering the status of actions that occur at a later time, it influences the predictions of their prerequisites (*e.g.*, *turn on the stove*) and prevents them from being repeated unnecessarily.

The best results come from combining both, improving performance by 1.5%, 1.8%, and 1.2%. This highlights the benefit of including both types of estimated priors when updating the transition model.

Choice of LLMs. Tab. 3 analyzes how BAGLM is affected by the LLM used to generate dependencies between steps. We compare LLaMA-3.3-70B-Instruct [7] and GPT-4.1-mini [21]. The two models achieve comparable overall performance: LLaMA-3.3-70B performs better on HT-Step (+0.3%) and Ego4D GoalStep (+2.7%) but worse on CrossTask (-1.1%). This trend suggests that LLaMA-3.3-70B performs better on datasets with more generic step descriptions, whereas proprietary models like GPT-4.1-mini are more effective at capturing dependencies among more atomic actions.

LMMs and segment duration. In Fig. 4, we show how BAGLM's performance varies w.r.t. the segment duration (from 1 to 4 seconds), considering different LMMs: InternVL2.5 8B [6], InternVL3 8B [35], LLaVA-OneVision 7B [13], and Qwen2.5 7B [2].

From the figure, we can see three trends. First, BAGLM consistently improves the performance of the underlying LMM it is applied to across all segment durations on both HT-Step and CrossTask, with gains becoming more pronounced as segment duration increases. This is related to the second trend, the impact of the segment duration. Segments that are too short may lack sufficient visual cues to evaluate the video-step alignment or the progress, while longer ones may span over multiple actions, making it harder to localize steps precisely. Using 2-second segments offers the best trade-off,

Table 4: Results with oracle dependencies and step progress.

Progress Oracle	Dep. Matrix Oracle	HT-Step	CrossTask	Ego4D
		57.4	59.8	43.3
✓		44.9	38.6	36.0
	✓	54.6	58.3	45.2
✓	✓	**62.6**	**66.9**	**82.2**

improving performance by +1.9%, +1.6%, +0.3%, and +2.3% across the three datasets compared to 1-second segments.

Third, BAGLM does not provide clear advantages on Ego4D Goal-Step (*i.e.*, -1.9%, -0.8%, -3%, and +0.3%). This can be attributed to challenges specific to this dataset. Ego4D Goal-Step videos are significantly longer on average (28 minutes vs. 6 minutes and less than 5 minutes for HT-Step and CrossTask, respectively) and contain coarser step annotations (53 seconds per segment on average vs. 16 seconds and 10.7 seconds for HT-Step and CrossTask, respectively). These longer videos and broader annotations result in more generic step descriptions, which in turn make it harder to infer accurate dependencies between steps and to estimate progress, as we will analyze in the following.

Use of annotated step boundaries from datasets. Given the different impact of BAGLM across datasets, we analyze the performance of the Bayesian filtering formulation without potential noise from the dependency matrix and/or the estimation of step progresses. With this aim, we conduct an analysis similar to Tab. 2, this time using step dependencies and action progress derived directly from the original datasets. For each video, we construct a chain of steps based on its ground truth temporal order. Note that while these dependencies reflect the observed execution order, *they are not* true semantic constraints, as the same steps may occur in different orders in other videos. Progress is computed as the normalized fraction of completion between a step's start and end timestamps.

Results of this oracle experiment are presented in Tab. 4, showing consistent gains across all datasets (*e.g.*, +5.2% on HT-Step), even the challenging Ego4D Goal-Step (*i.e.*, +38.9%). This confirms that the Bayesian filtering formulation is effective, and that jointly improving the elements used to estimate the transition matrix (*i.e.*, progress, dependency) would further boost the results of BAGLM.

6 Conclusion

In this work, we introduced BAGLM, a novel training-free approach for online video step grounding. BAGLM uses Bayesian filtering to integrate information from past video frames into LMM predictions. It consists of two key components: a transition model, initialized from step dependency matrices generated by LLMs and updated over time using action progress and dependency constraints; and an observation model, implemented as an LMM, which refines predictions as the video unfolds. We evaluated BAGLM on HT-Step, CrossTask, and Ego4D Goal-Step, showing that it outperforms training-based methods without requiring additional training or data collection.

Limitations. We identify two main limitations of our approach. First, BAGLM fully relies on a pre-trained LLM for estimating the step dependency matrix, so its effectiveness is closely tied to the quality of the model predictions. As more advanced LLMs become available, we expect that BAGLM performance will improve accordingly. Second, prior estimation within our Bayesian framework may be improved with more specific task knowledge. For example, we estimate step progress by querying an LMM: future works could refine this approach by incorporating priors on step durations.

Acknowledgements. This work was supported by the *Ministero delle Imprese e del Made in Italy* (IPCEI Cloud DM 27 giugno 2022 – IPCEI-CL-0000007) and by the European Union (Next Generation EU). Additional support was provided by the EU projects *ELIAS* (No. 101120237) and *ELLIOT* (No. 101214398). We acknowledge ISCRA for awarding this project access to the LEONARDO supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CINECA (Italy), as well as the EuroHPC Joint Undertaking for awarding us access to MareNostrum5 at BSC, Spain. The author also thanks Andrea Pilzer for valuable discussions on efficient video decoding.

References

[1] Triantafyllos Afouras, Effrosyni Mavroudi, Tushar Nagarajan, Huiyu Wang, and Lorenzo Torresani. Ht-step: Aligning instructional articles with how-to videos. *NeurIPS*, 2023.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv*, 2025.

[3] Brian Chen, Andrew Rouditchenko, Kevin Duarte, Hilde Kuehne, Samuel Thomas, Angie Boggust, Rameswar Panda, Brian Kingsbury, Rogerio Feris, David Harwath, et al. Multimodal clustering networks for self-supervised learning from unlabeled videos. In *ICCV*, 2021.

[4] Yuxiao Chen, Kai Li, Wentao Bao, Deep Patel, Yu Kong, Martin Renqiang Min, and Dimitris N Metaxas. Learning to localize actions in instructional videos with llm-based multi-pathway text-video alignment. In *ECCV*, 2024.

[5] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 2003.

[6] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv*, 2024.

[7] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv*, 2024.

[8] 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 *CVPR*, 2022.

[9] Tengda Han, Weidi Xie, and Andrew Zisserman. Temporal alignment networks for long-term video. In *CVPR*, 2022.

[10] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv*, 2021.

[11] Dohwan Ko, Joonmyung Choi, Juyeon Ko, Shinyeong Noh, Kyoung-Woon On, Eun-Sol Kim, and Hyunwoo J Kim. Video-text representation learning via differentiable weak temporal alignment. In *CVPR*, 2022.

[12] Baiqi Li, Zhiqiu Lin, Deepak Pathak, Jiayao Li, Yixin Fei, Kewen Wu, Xide Xia, Pengchuan Zhang, Graham Neubig, and Deva Ramanan. Evaluating and improving compositional text-to-visual generation. In *CVPR*, 2024.

[13] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-onevision: Easy visual task transfer. *Transactions on Machine Learning Research*, 2025.

[14] Zeqian Li, Qirui Chen, Tengda Han, Ya Zhang, Yanfeng Wang, and Weidi Xie. Multi-sentence grounding for long-term instructional video. In *ECCV*, 2024.

[15] Zhiqiu Lin, Xinyue Chen, Deepak Pathak, Pengchuan Zhang, and Deva Ramanan. Revisiting the role of language priors in vision-language models. *arXiv*, 2023.

[16] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *ECCV*. Springer, 2024.

[17] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. Univl: A unified video and language pre-training model for multimodal understanding and generation. *arXiv*, 2020.

[18] Effrosyni Mavroudi, Triantafyllos Afouras, and Lorenzo Torresani. Learning to ground instructional articles in videos through narrations. In *ICCV*, 2023.

[19] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *CVPR*, 2020.

[20] 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 *ICCV*, 2019.

[21] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. `https://openai.com/index/gpt-4-1/`, 2024.

[22] Chiara Plizzari, Alessio Tonioni, Yongqin Xian, Achin Kulshrestha, and Federico Tombari. Omnia de egotempo: Benchmarking temporal understanding of multi-modal llms in egocentric videos. *CVPR*, 2025.

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[24] Yuhan Shen and Ehsan Elhamifar. Progress-aware online action segmentation for egocentric procedural task videos. In *CVPR*, 2024.

[25] Yuhan Shen, Lu Wang, and Ehsan Elhamifar. Learning to segment actions from visual and language instructions via differentiable weak sequence alignment. In *CVPR*, 2021.

[26] Yaya Shi, Xu Yang, Haiyang Xu, Chunfeng Yuan, Bing Li, Weiming Hu, and Zheng-Jun Zha. Emscore: Evaluating video captioning via coarse-grained and fine-grained embedding matching. In *CVPR*, 2022.

[27] Yale Song, Eugene Byrne, Tushar Nagarajan, Huiyu Wang, Miguel Martin, and Lorenzo Torresani. Ego4d goal-step: Toward hierarchical understanding of procedural activities. *NeurIPS*, 2023.

[28] 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 *CVPR*, 2019.

[29] wikiHow. wikihow: How-to instructions you can trust. `https://www.wikihow.com/`.

[30] Jay Zhangjie Wu, Guian Fang, Haoning Wu, Xintao Wang, Yixiao Ge, Xiaodong Cun, David Junhao Zhang, Jia-Wei Liu, Yuchao Gu, Rui Zhao, et al. Towards a better metric for text-to-video generation. *arXiv*, 2024.

[31] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. *arXiv*, 2024.

[32] 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. In *EMNLP*, 2021.

[33] Luca Zanella, Massimiliano Mancini, Willi Menapace, Sergey Tulyakov, Yiming Wang, and Elisa Ricci. Can text-to-video generation help video-language alignment? *CVPR*, 2025.

[34] Hao Zhang, Aixin Sun, Wei Jing, Liangli Zhen, Joey Tianyi Zhou, and Rick Siow Mong Goh. Natural language video localization: A revisit in span-based question answering framework. *IEEE TPAMI*, 2021.

[35] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv*, 2025.

[36] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *CVPR*, 2019.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: This paper addresses the task of Video Step Grounding (VSG), which aims to ground natural-language steps composing a task into a video performing the task. Existing works are primarily offline and training-based, limiting their real-world applicability. Motivated by the promising performance of prompting Large Multimodal Models (LMMs) in performing VSG, we propose a novel training-free method BAGLM leveraging LMMs with Bayesian filtering to further reduce uncertain prediction. BAGLM scores new state-of-the-art VSG performance on three main benchmark datasets, despite of a more restrict setting. Our ablation study also proves our key methodological design.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have created a dedicate paragraph in Sec. 6, where we explicitly discussed main limitations in terms of assumption and identified main technical aspects that impact performance.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We leverage established theoretical results from Bayesian filtering, applying them to construct our method BAGLM.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe our method in full details with both equations. The experimental protocol is clearly explained in terms of datasets, metrics, and implementation details. The codebase will be released upon acceptance to ensure experimental result reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is available at `https://github.com/lucazanella/baglm`.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental details are provided in the Sec. 5. The codebase will also be released upon acceptance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: We follow the evaluation protocol of prior works (*e.g.*, NaSVA [14]) for fair comparison.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided essential information regarding the compute resources in Sec. 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have carefully inspected the code of ethics, and believe our work conforms to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the broader impacts in the Appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We use existing, publicly available dataset that already includes appropriate safeguards. As our work does not involve the release of high-risk models or data, we deem additional safeguards not required.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have properly referenced all the software, data, and related works that supported the development and evaluation of BAGLM in the Appendix. CrossTask follows BSD 3-Clause License, HT-Step follows Creative Commons Attribution-NonCommercial 4.0 International Public License and Ego4D Goal-Step follows MIT License Moreover, we publicly release our dataset and code.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The code and instructions for running our BAGLM will be released alongside the manuscript upon acceptance. Essential implementation details are already included in the manuscript at the submission time.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Our method exploits the internal knowledge of LLMs to obtain the dependencies among steps. The prompts used are listed in the Appendix.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

In the Appendix, we first provide the prompts used in BAGLM (Sec. A). We then present additional experimental analyses (Sec. B), including the type of prompt used to compute Video Step Grounding scores, the prior used in the update step of our method, and step localization results. We extend these analyses with evaluations on the COIN dataset to demonstrate the generalization of BAGLM across domains and tasks, robustness analyses of LLM-generated dependency matrices on HT-Step, and analyses of step prediction performance by temporal position and by the number of candidate steps. We also describe the computational cost of BAGLM during inference (Sec. C), discuss the broader social impacts of our work (Sec. D), and list the licenses and URLs for all assets used (Sec. E). Finally, we provide qualitative results (Sec. F).

A Prompts

We use three types of prompts in our approach. For each, the Large Multimodal Model (LMM) is prompted to generate an answer by selecting from a list of options. We then compute the model's score for each choice by applying a softmax to the logits of the first generated token and taking the probability assigned to the first token of each option. Video frames are positioned relative to the text according to the input formatting required by the underlying LMM.

The task prompt $\pi_{\texttt{VSG}}$ is used to estimate the probability that a given video segment corresponds to each step in the task, or to the "none" class. The LMM is prompted with the current video segment, the task `goal` (*i.e.*, the procedural task), and all candidate `steps`, and must select either one of the steps or the "none" option as the answer.

$\pi_{\texttt{VSG}}$

You are watching a video segment of someone attempting to `{goal}`.

What is the main action being performed in this exact moment?

Options:
`{steps}`

Answer with only the letter label of the correct option (e.g., A, B, ..., Z, AA, AB, etc.), with no extra text.

The progress prompt $\pi_{\texttt{prog}}$ is used to estimate the execution progress of a specific step within a video segment. The LMM is prompted with the current video segment, the task `goal`, and a given `step`, and must select a progress value from 0 to 9. Here, 0 indicates that the action is not occurring, 1 means the action is about to begin, 5 corresponds to the middle of the action, and 9 means the action has just finished.

$\pi_{\texttt{prog}}$

You are watching a short video clip.

The goal of the person in the video is: `{goal}`

The specific action of interest is: `{step}`

Rate how far along this action is in terms of execution progress, using a scale from 0 to 9:
- 0 = the action is not present in this clip
- 1 = the action is just beginning or about to begin
- 5 = the action is halfway complete
- 9 = the action is just finishing or about to finish

Respond with a single number from 0 to 9. Do not include any other text.

Table 5: Comparison of the off-the-shelf InternVL2.5-8B model using different prompts for Video Step Grounding. Queries / **S** indicates the number of LMM queries per video segment **S**.

Prompt	Queries / S	HT-Step R@1 ↑	CrossTask Avg. R@1 ↑	Ego4D Goal-Step R@1 ↑
$\pi_{\texttt{VSG}}^{\texttt{bin}}$	K	**52.5**	**56.0**	41.8
$\pi_{\texttt{VSG}}$	1	52.0	54.3	**45.2**

Finally, the prerequisite prompt $\pi_{\texttt{prereq}}$ is used to estimate step dependencies within a task. The LLM is prompted with the task `goal`, a `step`, and a candidate `prerequisite`, and must answer either "Yes" or "No".

$\pi_{\texttt{prereq}}$

You are performing the task: `{goal}`

Is the following step strictly required before another?

Prerequisite candidate: `{prerequisite}`
Target step: `{step}`

Answer "Yes" if the target step cannot be completed correctly without first completing the prerequisite step. Otherwise, answer "No".

Answer:

B Additional analyses

In this section, we present additional analyses of BAGLM, including ablations on the prompts used to compute Video Step Grounding scores, the choice of step priors, and step localization performance. We also evaluate generalization on the COIN dataset, analyze the robustness of LLM-generated dependency matrices on HT-Step, and study step prediction performance by temporal position and by the number of candidate steps.

Binary vs multi-choice VSG prompt. While we compute VSG scores by framing the problem as a multi-choice question answering task, an alternative is to treat it as a binary question answering task. In this case, we can ask the LMM whether a given video segment corresponds to a specific step, querying the LMM for each step independently. We compare these approaches by evaluating the off-the-shelf model InternVL2.5-8B using the multi-choice prompt $\pi_{\texttt{VSG}}$ and the binary prompt $\pi_{\texttt{VSG}}^{\texttt{bin}}$:

$\pi_{\texttt{VSG}}^{\texttt{bin}}$

You are watching a video segment of someone attempting to `{goal}`.

Is the person currently performing the action: "`{step}`"?

Answer with "Yes" or "No" only.

As shown in Table 5, using the multi-choice formulation improves performance by 3.4% on Ego4D Goal-Step. On HT-Step, performance is comparable (a slight drop of 0.5%), while on CrossTask, the binary formulation performs better by 1.7%. However, the binary formulation requires querying the LMM K times per video segment, once per candidate step, whereas the multi-choice formulation requires only one query per segment.

For example, for the HT-Step video `-25-1nb0ki0`, which contains 283 segments (each 2 seconds long) and 10 steps, using $\pi_{\texttt{VSG}}$ requires 24.79 seconds in total (approximately 0.0876 seconds per

Table 6: Ablation study on the step prior $P(\mathtt{A}_t = a_i)$.

	HT-Step R@1 ↑	**CrossTask** Avg. R@1 ↑	**Ego4D Goal-Step** R@1 ↑
1) $f_{\mathtt{LMM}}(\mathbf{S}_t, \pi_{\mathtt{next}})[i]$	50.5	52.6	28.1
2) $\sum_{a_j \in \mathcal{A}} P(\mathtt{A}_t = a_i \vert \mathtt{A}_{t-1} = a_j) \cdot P(\mathtt{A}_{t-1} = a_j)$	55.6	57.3	39.9
3) $\frac{1}{S+1}$	**57.4**	**59.8**	**43.3**

Table 7: Comparison between NaSVA and our BAGLM on the HT-Step dataset.

Method	**HT-Step** (↑ mAP@IoU)			
	@0.3	@0.5	@0.7	@[0.3–0.7]
NASVA [14]	13.7	6.0	1.4	6.7
BAGLM	**15.9**	**6.7**	**1.5**	**7.7**

segment), while $\pi^{\mathtt{bin}}_{\mathtt{VSG}}$ takes 214.04 seconds (about 0.756 seconds per segment) with one NVIDIA H100 GPU with 64GB of memory. Given the comparable accuracy and significantly lower inference time, we adopt $\pi_{\mathtt{VSG}}$ as our default prompt for computing Video Step Grounding scores.

Prior over the states $P(\mathtt{A}_t = a_i)$. In Eq. (9), describing the update step, we normalize the scores using a general prior $P(\mathtt{A}_t = a_i)$ over all possible states. In this section, we explore alternatives to the adopted uniform prior in our method.

Table 6 shows the results of our analyses. Row (1) reports results using a context-dependent prior, where the LMM is prompted with $\pi_{\mathtt{next}}$ to predict the probability distribution over the next most likely action given the current segment, the task `goal`, and all candidate `steps`:

$\pi_{\mathtt{next}}$

You are watching a video segment of someone attempting to `{goal}`.

What is the most likely next step in the sequence?

Options:
`{steps}`

Answer with only the letter label of the correct option (e.g., A, B, ..., Z, AA, AB, etc.), with no extra text.

Predicting the next action is often similar to predicting the current one, resulting in a prior distribution close to the VSG scores. As a result, the probability associated with some true positive steps will be erroneously lowered in favor of the "none" class.

Row (2) presents the results of using a prior initialized from a uniform distribution and updated at each timestamp by applying the state transition model. In this prior, transitions to "none" are often the most likely as they are not penalized by step validity or readiness. This lowers the likelihood of the "none" class in the posterior, promoting false positive predictions.

Row (3) shows the performance using a uniform prior (our adopted one), which assigns equal probability to all steps. This setting achieves the best results among all the considered prior options.

Step localization. While we report performance using Recall@1 and Avg. Recall@1 as our main metrics in the main manuscript, following the evaluation protocol from [14, 4], we also evaluate step localization performance (*i.e.*, predicting the start and end timestamps of steps) by leveraging the final belief distribution for each video segment. This results in an alignment matrix of size $T \times K$, where T is the number of video segments and K is the number of task steps. Following [18], we extract temporal segments from this matrix using a 1D blob detection approach. Specifically, we apply Laplacian of Gaussian (LoG) filters at 13 different scales, corresponding to Gaussian standard deviations ranging from 1 to 480. The confidence score for each predicted segment is computed as the average step probability across all video segments within the predicted temporal boundaries. We

Table 8: Ablation study on varying the used LMM on HT-Step. Bold indicates the best result within each LMM group (*i.e.*, with and without BAGLM).

Method	HT-Step (↑ mAP@IoU)			
	@0.3	@0.5	@0.7	@[0.3–0.7]
LLAVA-OV-7B [13]	14.1	5.5	1.3	6.6
+ BAGLM	**14.7**	**5.8**	**1.5**	**6.9**
QWEN2.5-VL-7B [2]	15.5	6.1	1.4	7.3
+ BAGLM	**15.8**	**6.6**	**1.7**	**7.7**
INTERNVL3-8B [35]	15.6	5.8	1.4	7.2
+ BAGLM	**17.5**	**6.5**	**1.6**	**8.1**
INTERNVL2.5-8B [6]	15.0	6.0	1.3	7.0
+ BAGLM	**15.9**	**6.7**	**1.5**	**7.7**

Table 9: Comparison between NaSVA (best offline competitor) and our online method BAGLM on the COIN dataset.

Method	↑ R@1
Offline	
NASVA [14]	36.8
Online	
NASVA [14]	32.1
INTERNVL2.5-8B [6]	47.9
BAGLM	**51.0**

compare our method against NaSVA [14], the best method (see Tab. 1 in the main manuscript) with publicly available code. NaSVA uses a transformer-based architecture where steps act as queries that iteratively attend to video features, producing an alignment matrix between steps and video segments. To ensure a fair comparison, we derive the start and end timestamps for each step by applying the same 1D blob detection procedure used in our method to NaSVA's similarity matrix, computed from its final transformer layer. Before applying blob detection, we convert the similarity scores for each segment into a valid probability distribution over steps by applying a softmax function.

We conduct this analysis on the HT-Step dataset [1] and report results in Tab. 7. We evaluate localization performance using the HT-Step protocol, which measures article-grounding mean Average Precision (mAP) at various IoU thresholds. This involves treating each step as a class-agnostic text query, computing AP per activity, and averaging the results to obtain overall mAP. BAGLM performs better than NaSVA at all IoU thresholds (*i.e.*, +2.2%, +0.7%, and +0.1%).

We complement the results of Fig. 4 in the main manuscript by evaluating how BAGLM performs when applied to different LMMs, *i.e.*, InternVL2.5 8B [6], LLaVA-OneVision 7B [13], Qwen2.5 7B [2], and InternVL3 8B [35], using mAP as the evaluation metric. From Tab. 8, we can see that adding BAGLM to any LMM consistently improves performance. The improvements are largest at lower IoU thresholds (*e.g.*, +1.9% at 0.3 w.r.t. the off-the-shelf InternVL3 8B) and gradually diminish at higher thresholds (*e.g.*, +0.2% at 0.7).

COIN. To further demonstrate the generalizability of BAGLM, we evaluate it on the COIN [28] dataset, which contains instructional YouTube videos spanning various domains. We use the test split, consisting of 2,797 videos, of which 2,354 were successfully downloaded (the remainder being unavailable or set to private). These videos cover 12 domains (*i.e.*, Dish, Drink and Snack, Electrical Appliance, Furniture and Decoration, Gadgets, Housework, Leisure and Performance, Nursing and Care, Pets and Fruit, Science and Craft, Sport, and Vehicle) and are categorized into 180 tasks.

We compare BAGLM against three baselines: (1) InternVL2.5-8B [6] (off-the-shelf), (2) NaSVA [14] (best offline competitor), (3) an online version of NaSVA. As shown in Tab. 9, BaGLM outperforms NaSVA by 14.2 (offline) and 18.9 (online), and achieves a gain of 3.1 over InternVL2.5-8B. These results suggest that our method generalizes well across diverse domains and tasks.

Dependency matrix robustness. We evaluate the robustness of LLM-generated dependency matrices on the HT-Step dataset, which includes 120 cooking-related tasks, each with 5 videos (600 videos

Table 10: Dependency violations and fraction of tasks with violations for different threshold values.

Metric	Model	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
Violated dependencies (%)	LLaMA-3.3	26.1	8.6	8.4	8.4	8.4	8.4	8.4	8.3	8.3	8.1	8.1
	GPT-4.1-mini	26.1	15.4	12.5	11.6	10.8	10.2	9.6	8.9	8.2	7.6	0.0
Tasks w/ violation (%)	LLaMA-3.3	100.0	72.0	71.3	71.3	69.3	69.3	69.3	68.7	68.7	68.7	68.7
	GPT-4.1-mini	100.0	74.0	65.3	64.0	59.3	54.0	51.3	48.0	44.7	42.0	0.0

Table 11: R@1 performance across step positions on HT-Step.

Model	Bin 0	Bin 1	Bin 2	Bin 3	Bin 4
InternVL2.5-8b	53.6	52.7	54.6	49.7	49.3
BAGLM	56.1	57.2	60.2	56.8	56.6

in total). For each task, we convert the soft dependency matrix generated by either GPT-4.1-mini or LLaMA-3.3-70B-Instruct into binary (hard) dependencies at various thresholds. To evaluate robustness, we measure the ratio of violated dependencies. A dependency $\mathbf{D}_{i,j}$, indicating that step a_j is a prerequisite for step a_i, is considered violated if a_i occurs before the first occurrence of a_j in any of the videos for that task (*i.e.*, the prerequisite should always occur before the dependent step whenever both are present).

Tab. 10 reports the percentage of violated dependencies and the percentage of tasks with at least one violation for both LLaMA-3.3-70B-Instruct and GPT-4.1-mini. Both models perform well across different tasks, with roughly 10% of dependencies being violated depending on the threshold. In particular, LLaMA-3.3-70B-Instruct shows a flat violation ratio ($\sim$8%) across a wide range of thresholds, suggesting that its outputs are mostly binary (*i.e.*, close to 0 or 1). In contrast, GPT-4.1-mini shows a smoother decline in violation rate as the threshold increases, indicating a more graded confidence distribution. Even at low thresholds, a portion of tasks (28% for LLaMA-3.3 and 26% for GPT-4.1-mini at threshold 0.1) have no violated dependencies, suggesting that the predicted matrices can approximate the oracle matrix.

Step prediction by temporal position. We analyze how step prediction performance varies across the temporal positions of steps in HT-Step. For each step, we normalize its occurrence within the video to the range $[0, 1]$ and divide these positions into five bins, then compute R@1 for each bin. Tab. 11 reports R@1 across these bins, from early (bin 0) to late (bin 4) steps. The off-the-shelf model (InternVL2.5-8B), which does not leverage past context, maintains relatively stable performance. It performs better on early and mid-range steps (bin 0: 53.6%, bin 2: 54.6%) than on the final steps (bin 4: 49.3%), suggesting that initial to mid-range steps are more semantically distinctive while later steps are harder to recognize.

In contrast, BAGLM shows an upward trend from early to mid-range bins, likely due to its use of accumulated temporal context: R@1 increases by +1.1% from bin 0 to bin 1 and +3.0% from bin 1 to bin 2. Its performance then decreases toward the final bin, likely due to limitations inherited from the base model (*i.e.*, InternVL2.5-8B performs worst in the final part of the video) as well as error propagation.

Step prediction by number of candidate steps. We analyze how performance changes with the number of candidate steps in the HT-Step dataset by grouping videos based on the number of steps in their tasks. For each group, we compute the average R@1 using the off-the-shelf InternVL2.5-8B model (Fig. 5). The results show a non-monotonic trend: except for tasks with very few steps (fewer than 4), there is no clear decline in performance as the number of steps increases. This suggests that InternVL2.5-8B can handle prompts with many candidate steps. Note that some bins contain few videos (*e.g.*, 14 and 29 for tasks with 17-18 steps), which can cause large fluctuations in the measurements (*e.g.*, from 57.1% to 34.5%).

C Inference speed analysis

We measure inference speed on a single NVIDIA H100 GPU with 64GB of memory. Figure 6 shows the average processing time per video segment for three main operations: video loading and preprocessing, computing VSG scores, and estimating action progress. These estimates were

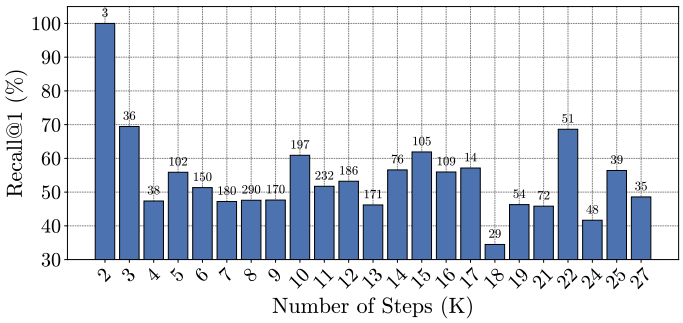

Figure 5: R@1 on the HT-Step dataset, grouped by the number of candidate steps in each task.

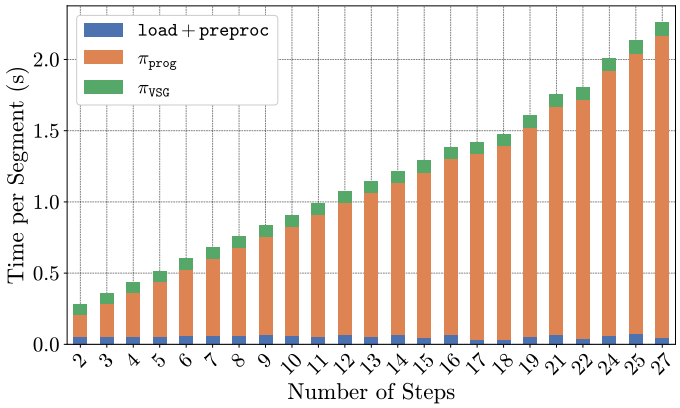

Figure 6: Average processing time per video segment for computing VSG scores, estimating action progress, and video loading/preprocessing, for HT-Step videos grouped by their number of steps.

computed on HT-Step videos with the same number of steps. Our results show that the total processing time per segment is largely dominated by action progress estimation, which grows linearly with the number of steps in the task. Importantly, the majority of the segments are processed in under 2 seconds, implying that the 2-second segments can be processed in real time.

As an example, consider the video `-25-1nbOki0` from the HT-Step dataset. This video is 567 seconds long and corresponds to the task *Make Nigerian Style Jollof Rice*, which includes 10 steps. We generate the step dependency matrix for this task using GPT-4.1-mini with prompt $\pi_{\texttt{prereq}}$, requiring 87.84 seconds to compute. This matrix is generated only once per task and reused for all related videos. We process each 2-second video segment as follows: we compute VSG scores using InternVL2.5-8B with prompt $\pi_{\texttt{VSG}}$, taking 0.088 seconds per segment. We estimate action progress using prompt $\pi_{\texttt{prog}}$, requiring 0.814 seconds per segment. Additional time is spent on model initialization (performed once) and video decoding and preprocessing. Using the `torchcodec`[3] library, decoding and preprocessing take approximately 1.201 seconds per segment on CPU, with possible speedups using GPU acceleration. In comparison, the Bayesian filtering step is computationally negligible. Overall, excluding step dependency generation and model loading, each 2-second segment of this video can be processed in approximately 2.1 seconds.

D Broader impact

Our approach enables online understanding of instructional video steps without requiring task-specific training. This makes it suitable for deployment in real environments, providing feedback to users performing tasks. Such capabilities can benefit a wide range of applications, including education, skill development, and assistive technologies, especially in contexts where annotated data is limited or users require immediate feedback (*e.g.*, remote learning, home cooking, or physical rehabilitation).

[3] `https://docs.pytorch.org/torchcodec/stable/index.html`

Table 12: List of URLs and licenses for all assets used.

Name	URL	License
LLaVA-OV-7B	https://huggingface.co/lmms-lab/llava-onevision-qwen2-7b-ov	Apache 2.0
Qwen2.5-VL-7B	https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct	Apache 2.0
InternVL2.5-8B	https://huggingface.co/OpenGVLab/InternVL2_5-8B	MIT
InternVL3-8B	https://huggingface.co/OpenGVLab/InternVL3-8B	MIT
CrossTask	https://github.com/DmZhukov/CrossTask	BSD-3-Clause
HT-Step	https://github.com/facebookresearch/htstep	CC-BY-NC 4.0
Ego4D Goal-Step	https://github.com/facebookresearch/ego4d-goalstep	MIT
COIN	https://github.com/coin-dataset/annotations	CC BY-NC 4.0

However, as our method depends on large pretrained multimodal models, it may also inherit their limitations, such as biased predictions or inconsistent performance across different cultural or domain-specific tasks. Future work should explore strategies for mitigating such biases, ensuring equal performance across diverse user populations.

E Assets

Tab. 12 lists URLs and licenses for all the assets used in the paper.

F Qualitative results

Fig. 7 showcases qualitative results produced by BAGLM on two test videos: one from the HT-Step dataset (task: *Make Milanesa*) and one from the CrossTask dataset (task: *Make a Latte*). For each video, we plot the ground truth step boundaries and the step probabilities over video segments, as predicted by our method and by the off-the-shelf version of the LMM. We also show keyframes at selected time points, indicated by arrows.

In the *Make Milanesa* video, the off-the-shelf LMM assigns a high probability to the step *Dip the steak* at the very beginning of the video, as the person is seen picking up the steak from a plate, as shown by the corresponding keyframe. For the same step, BAGLM initially assigns a low probability but increases it over time, consistent with the true presence of the step in the video. Around timestamp `00:20`, the off-the-shelf LMM mistakenly assigns a high probability to step *Prep your egg mixture*, likely due to the presence of the egg mixture in the video segment. In contrast, BAGLM consistently predicts *Dip the steak* at this point, leveraging prior context to avoid a false positive. At timestamp `00:30`, although the ground truth still indicates *Dip the steak* is in progress, the step is already completed and BAGLM correctly detects this.

In the *Make a Latte* video, BAGLM effectively suppresses probabilities for steps that fall outside their actual temporal boundaries. Leveraging LMMs to compute VSG scores proves particularly beneficial in distinguishing visually similar actions such as *pour milk* and *pour espresso*. Importantly, steps not present in the video are not falsely identified by BAGLM, showcasing its precision.

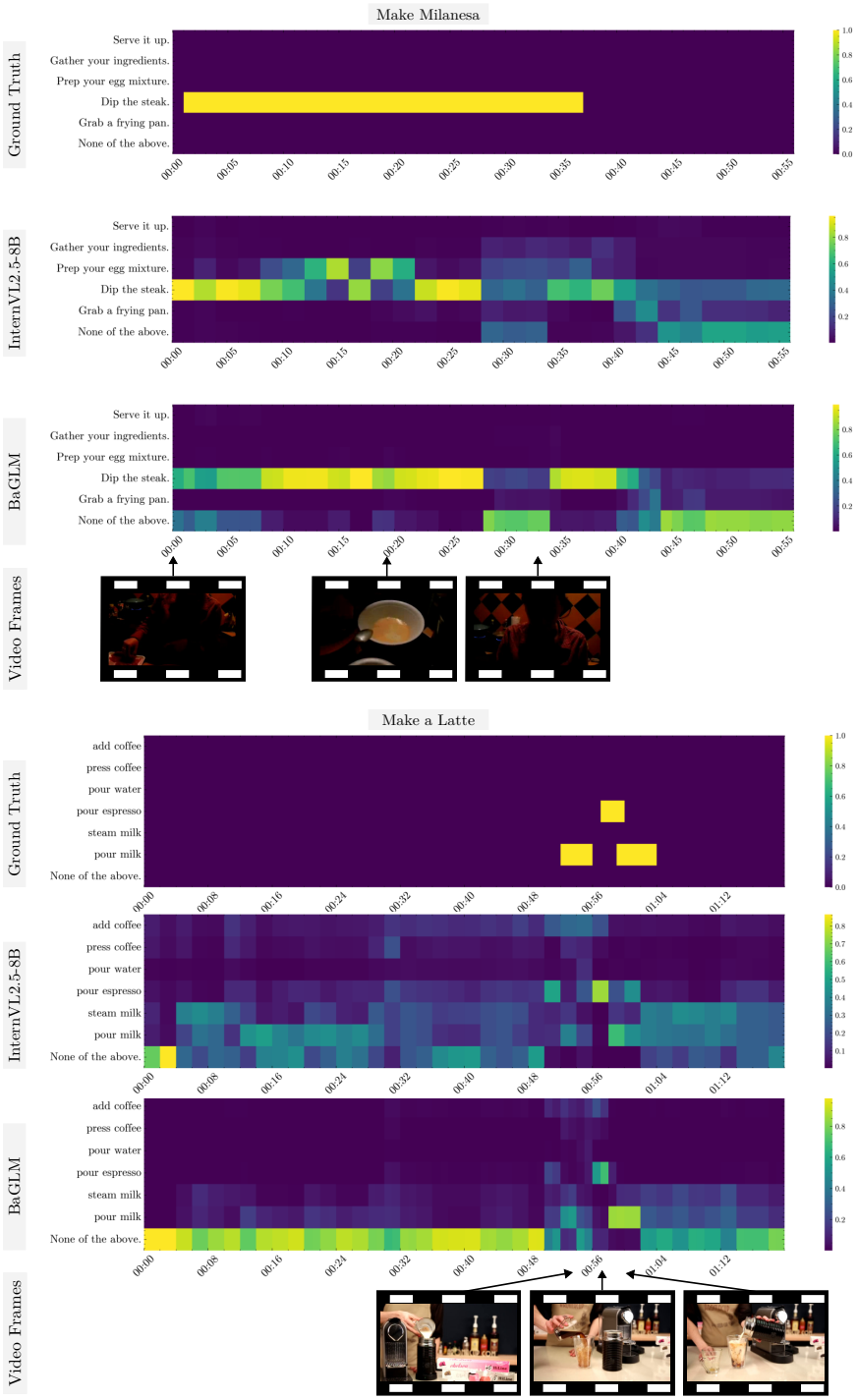

Figure 7: Qualitative results of BAGLM on test videos from HT-Step (*Make Milanesa*) and CrossTask (*Make a Latte*). Ground truth step boundaries and predicted step probabilities per segment are shown for both BAGLM and the off-the-shelf LMM. Arrows point to the timestamps of selected keyframes.