PKR-QA: A Benchmark for Procedural Knowledge Reasoning with Knowledge Module Learning

Thanh-Son Nguyen[*2], **Hong Yang**[*1,2], **Tzeh Yuan Neoh**[*1,4], **Hao Zhang**[*1,2], **Ee Yeo Keat**[1,2], **Basura Fernando**[1,2,3]

[1]Centre for Frontier AI Research, Agency for Science, Technology and Research, Singapore
[2]Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore
[3]College of Computing and Data Science, Nanyang Technological University, Singapore
[4] Harvard University
Nguyen_Thanh_Son@a-star.edu.sg, Fernando_Basura@a-star.edu.sg.

Abstract

We introduce PKR-QA (*Procedural Knowledge Reasoning Question Answering*), a new benchmark for question answering over procedural tasks that require structured reasoning. PKR-QA is constructed semi-automatically using a *procedural knowledge graph* (PKG), which encodes task-specific knowledge across diverse domains. The PKG is built by curating and linking information from the COIN instructional video dataset and the ontology, enriched with commonsense knowledge from ConceptNet and structured outputs from Large Language Models (LLMs), followed by manual verification. To generate question-answer pairs, we design graph traversal templates where each template is applied systematically over PKG. To enable interpretable reasoning, we propose a neurosymbolic approach called *Knowledge Module Learning* (KML), which learns procedural relations via neural modules and composes them for structured reasoning with LLMs. Experiments demonstrate that this paradigm improves reasoning performance on PKR-QA and enables step-by-step reasoning traces that facilitate interpretability.

Code & Dataset —
https://github.com/LUNAProject22/KML
Extended version — https://arxiv.org/abs/2503.14957

Introduction

Interest in understanding procedural tasks is growing, driven by applications in cooking, machinery repair, medical procedures, and daily activities (Ashutosh et al. 2024), as reflected by platforms like WikiHow. Such tasks consist of sequences of steps, where each step serves a specific purpose and tools are used accordingly. Human understanding of these tasks arises from lifelong learning, enriched by commonsense knowledge of objects and their affordances, as well as the ability to reason about temporal and causal dependencies within multi-step processes. This enables individuals to infer not just what to do, but why and how each step contributes to the overarching goal. Emulating this procedural knowledge reasoning is essential for machines to assist in complex real-world tasks.

Despite the rise of visual question answering and general knowledge-based benchmarks, there remains a gap in evaluating a model's ability to perform procedural knowledge reasoning. To address this, we introduce PKR-QA (*Procedural Knowledge Reasoning Question Answering*), a new benchmark designed to assess Vision Language Models (VLMs) and Neurosymbolic (NS) methods on diverse procedural reasoning capabilities, including multi-hop, deductive, probabilistic, contextual, causal, and counterfactual reasoning. Many procedural tasks require domain-specific knowledge and an understanding of task structures, which can be effectively captured using knowledge graphs. To construct the PKR-QA questions and answers, we build a *procedural knowledge graph* (PKG) that encodes temporal relationships and causal links among key concepts. Our dataset challenges models to answer procedural questions by combining information from the video with external knowledge about the task. Unlike previous video QA benchmarks that primarily test visual comprehension, i.e., reasoning over what is in the video, our dataset emphasizes task-centric reasoning and knowledge grounded in the procedure. This introduces new challenges that jointly demands visual understanding, visual reasoning, and procedural knowledge reasoning.

While VLMs demonstrate impressive reasoning and knowledge capabilities, their internal reasoning mechanism is not that transparent. Techniques like chain-of-thought, graph-of-thought, and tree-of-thought aim to enhance VLM reasoning (Wei et al. 2022; Besta et al. 2024; Yao et al. 2023), yet the reasoning remains implicit due to the lack of constraints on intermediate variables. Moreover, reasoning and execution are often entangled within the VLM's internal mechanism. To address this, methods like ViperGPT decouple reasoning from execution by generating executable programs. Inspired by such approaches and neurosymbolic (NS) models (Johnson et al. 2017; Mascharka et al. 2018; Andreas et al. 2016; Perez et al. 2018; Hudson and Manning 2019, 2018; Chen et al. 2021; Endo et al. 2023), we propose Knowledge Module Learning (KML), which models knowledge-based relations as parameterized, trainable mod-

[*]These authors contributed equally.

ules leveraging on modern embeddings. KML generates the program of knowledge modules to answer knowledge-based reasoning questions as compositions of knowledge modules using the PKG schema and LLMs. KML explicitly answers knowledge-based reasoning questions leveraging the powerful reasoning mechanism of moderns LLMs allowing it to execute reasoning steps in a more transparent manner to derive the answer. KML supports explicit knowledge-based reasoning, constrained to a well-defined set of knowledge operations, plays a crucial role beyond procedural tasks in domains that require structured decision-making, reliability, and interpretability. KML provides explicit reasoning steps that can be systematically verified, debugged, and refined. Unlike end-to-end black-box models, which often lack transparency in their decision-making, structured reasoning through predefined operations enables more interpretable, trustworthy, and controllable AI systems. Our dataset and the approach promote these aspects.

Our contributions are three-fold. First, we introduce PKR-QA, a new benchmark to test procedural knowledge reasoning by question answering. Second, we propose KML, a neuro-symbolic method that executes LLM-generated programs over knowledge graphs using relation-specific neural modules. Third, we benchmark PKR-QA with state-of-the-art VLMs and NS methods. KML variants outperform all baselines, highlighting the benefit of structured knowledge for interpretable and controllable procedural reasoning. We believe PKR-QA will serve as a valuable resource to advance trustworthy AI in this emerging domain.

Related Work

Understanding procedural and instructional videos remains a significant challenge in computer vision. The authors in (Ashutosh et al. 2024) introduced the key step recognition and procedure-aware video representation learning approach using step transitions is presented in (Zhou et al. 2023). We leverage on these works. **Knowledge-based Visual Question Answering** has become popular in recent years. The majority of the past benchmarks on knowledge-based visual question answering are for images while more recently authors in (Wang et al. 2024) presented a benchmark to evaluate situated and open-world common-sense reasoning in videos with compositionality, temporality, and causality (Parmar et al. 2024). While the motivation of our work shares similarities with (Wang et al. 2024), there are notable distinctions. Their focus lies in situated commonsense reasoning grounded in the specific contexts depicted in videos, whereas we concentrate on testing the procedural understanding of models given some partial information such as a step of a task. **Integrating External Knowledge with Video Understanding** has been explored through knowledge graphs (KGs) for tasks like activity recognition (Ma et al. 2022) and visual commonsense reasoning (Lin et al. 2019). VidSitu (Sadhu et al. 2021) links events to semantic roles, while TVQA+(Lei et al. 2019) uses script knowledge for story-based QA. However, these works focus on descriptive reasoning rather than procedural reasoning. **Neurosymbolic** frameworks like ViperGPT (Surís, Menon, and Vondrick 2023), MoreVqa (Min et al. 2024) and (Choudhury et al. 2024) decouple reasoning (e.g., program generation) from execution (e.g., API calls). While ViperGPT uses predefined functions for visual queries, it lacks mechanisms to learn domain-specific knowledge modules or constrain reasoning to procedural logic. Similarly, chain-of-thought prompting (Wei et al. 2022) improves transparency in LLMs/VLMs but remains dependent on the model's internal knowledge, which may be unreliable for specialized domains. Our framework advances this by (1) training lightweight, interpretable Knowledge Modules directly from a domain specific KG and (2) constraining reasoning to predefined operations ensuring alignment with domain knowledge. Our work is also related to (Zhong et al. 2024; Shah et al. 2024), which employ **logical queries** to extract information from a KG. This emphasis on curated, domain-specific knowledge is especially valuable for AI assistants operating in contexts where domain expertise, rather than general commonsense reasoning, is crucial for success. Prior knowledge completion works that learns embeddings of entities and relations in KGs are also related to us (Bordes et al. 2013; Wang et al. 2014). In contrast, KML utilizes embeddings to learn relational mappings, employing a separate neural network for each relation through contrastive learning.

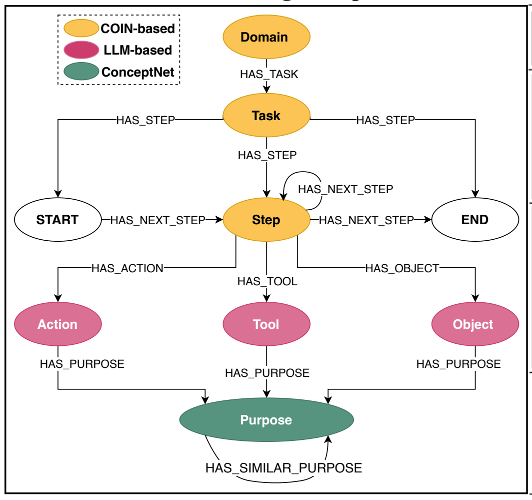
PKG Construction and Dataset Creation

Motivated by recent advances in the semi-automated construction of knowledge-based question answering datasets (Hoang et al. 2024), we introduce the *Procedural Knowledge Reasoning Question Answering* (PKR-QA) dataset. It is built on a *Procedural Knowledge Graph* (PKG) that integrates information from the COIN training set, the COIN ontology, GPT-4o-generated annotations, and external commonsense knowledge from ConceptNet, followed by human verification. To enable systematic QA generation, we define a set of question templates and associated Cypher queries (Francis et al. 2018), which are executed over the knowledge graph to retrieve correct answers. In the following sections, we detail the construction of both the knowledge graph and the QA dataset.

Defining PKG's Schema (PKGS). A knowledge graph schema provides a high-level abstraction of the graph, specifying the types of entities that exist and the valid relationships that can occur between them. It serves as a blueprint for structuring the data and interpreting the semantics of the graph. In our case, PKGS contains nodes where each node correspond to an entity type ($\mathcal{E}$) and edges represent relation types ($\mathcal{R}$) (Figure 1). The core entity types in PKGS include *Domain*, *Task*, *Step*, *Action*, *Object*, *Tool*, and *Purpose*. Relation types capture meaningful procedural links, such as task-step associations or tool usage (e.g. `HAS_TOOL`), and may carry attributes like *id*, *type*, or additional semantic metadata.

Populating PKG. Based on this schema, we instantiate PKG by populating it with specific entity and relation instances. Each entity $e \in \mathcal{E}$ has three main attributes: *type*, *name*, and *id*. Relations $r \in \mathcal{R}$ represent specific connections between entity instances. To construct this graph,

Procedural Knowledge Graph Schema



Traversal Template	Sample Question
Task $\xrightarrow{\text{HAS\_STEP}}$ Step $\xrightarrow{\text{HAS\_TOOL}}$ Tool $\xrightarrow{\text{HAS\_PURPOSE}}$ Purpose	What is the **purpose** of the **tool** used in this step?
Tool : T_1 $\xrightarrow{\text{HAS\_PURPOSE}}$ Purpose: P_1 **where** Task $\xrightarrow{\text{HAS\_STEP}}$ Step $\xrightarrow{\text{HAS\_TOOL}}$ Tool : $T1$ $\cancel{\xrightarrow{\text{HAS\_PURPOSE}}}$ Purpose : P_1	What are the **additional purposes** for which the tool shown in this video can be used, aside from its intended use in the step demonstrated?
Tool : T_1 $\xrightarrow{\text{HAS\_PURPOSE}}$ Purpose : P_1 **where** Task $\xrightarrow{\text{HAS\_STEP}}$ Step $\xrightarrow{\text{HAS\_TOOL}}$ Tool : T_2 $\xrightarrow{\text{HAS\_PURPOSE}}$ Purpose : P_2 $\xrightarrow{\text{HAS\_SIMILAR\_PURPOSE}}$ Purpose : P_3 **and** $T_1 \neq T_2$ **and** $(P_1 = P_2$ **or** $P_1 = P_3)$	what is an **alternative tool** that can be used for this step if I don't have the current tool?
Task : A_1 $\xrightarrow{\text{HAS\_STEP}}$ Step $\xrightarrow{\text{HAS\_TOOL}}$ Tool : T_1 $\xrightarrow{\text{HAS\_PURPOSE}}$ Purpose $\xleftarrow{\text{HAS\_PURPOSE}}$ Tool : T_1 $\xleftarrow{\text{HAS\_TOOL}}$ Step $\xleftarrow{\text{HAS\_STEP}}$ Task : A_2 **where** $A_1 \neq A_2$	What can be **other task** that can use the **tool** shown in this video for the same purpose?

Figure 1: (Left) Schema of the Procedural Knowledge Graph (PKG). Middle: Traversal templates. Right: Corresponding questions. Blue text indicates information grounded in the input video. Red text denotes the target answer node.

we integrate annotations from the COIN dataset (Tang et al. 2019), fine-grained information extracted using GPT-4o (Hurst et al. 2024), and commonsense knowledge from ConceptNet (Speer, Chin, and Havasi 2017).

COIN-based Data (`Domain, Task, Step`): We extract the procedural structure from the training split of the COIN dataset, which provides annotations for domains, tasks, and steps. For each unique instance of these entities, we create corresponding nodes in the graph. We define `HAS_TASK` edges to link each domain to its associated tasks, and `HAS_STEP` edges to connect tasks to their constituent steps. To capture procedural flow, we analyze step sequences from all training videos and construct `HAS_NEXT_STEP` relations that represent the temporal ordering of steps. Additionally, we introduce special `START` and `END` nodes to explicitly mark task boundaries. Each `HAS_NEXT_STEP` edge is annotated with its observed frequency to model empirical transition likelihoods between steps.

LLM-Augmented Data (`Action, Object, Tool`): To enhance procedural specificity, we use GPT-4o to extract action-object pairs from step descriptions. For instance, the phrase "remove the tire" yields the action "remove" and the object "tire". Because COIN lacks explicit tool annotations, we further prompt GPT-4o to infer potential tools for each step. The tool lists are manually verified. We standardize terms (e.g., "scissor" $\rightarrow$ "scissors"), consolidate duplicates (e.g., "marker", "marker pen"), and unify task-specific variants (e.g., "toilet detergent" $\rightarrow$ "detergent"). Some task-sensitive distinctions are preserved (e.g., "filter" $\rightarrow$ "coffee filter") when relevant to the procedural context.

ConceptNet-based Data (`Purpose`): In procedural tasks, actions are typically performed—and tools or objects used—with specific purposes. To represent such intent in our graph, we incorporate commonsense knowledge from ConceptNet, a large-scale semantic network that connects words and phrases through meaningful, human-readable relations such as `UsedFor`, `CapableOf`, and `IsA`. Specifically, we extract potential purposes of actions, tools, and objects using the `UsedFor` and `CapableOf` relations. To contextualize these purposes within specific procedural steps and tasks, we use GPT-4o to infer task- and step-specific interpretations for each entity's purpose –see the prompt template in (Nguyen et al. 2025) for more details. This step ensures that the resulting knowledge is not generic but grounded in the procedural context in which the entity is used.

To reduce redundancy and merge semantically similar purposes, we compute pairwise cosine similarities of Sentence-BERT embeddings (Reimers and Gurevych 2019) and apply a similarity threshold of 0.8, validated through manual inspection. Pairs of highly similar purposes are linked using a `HAS_SIMILAR_PURPOSE` edge. The resulting purpose-augmented graph is stored in Neo4j[1], supporting efficient querying and reasoning via Cypher. All contents are manually verified to ensure the reliability of the KG. The constructed KG contains 2,954 unique entities and 12,484 relations. Detailed distributions of entities and relations are shown in (Nguyen et al. 2025).

PKR-QA is a multiple-choice question answering dataset designed to evaluate reasoning over procedural knowledge in instructional videos. Each instance consists of a video segment, a question, five answer choices, and one correct answer. Unlike traditional video QA datasets that focus on grounding answers directly in the visual content of a single video, PKR-QA emphasizes reasoning over procedural knowledge that extends beyond the given video.

Traversal Templates for Procedural Reasoning We define questions in PKR-QA based on traversal templates which are specific reasoning patterns over PKG. For instance, the traversal `Step` $\xrightarrow{\text{HAS\_TOOL}}$ `Tool` $\xrightarrow{\text{HAS\_PURPOSE}}$ `Purpose` corresponds to the question: "What is the purpose

[1]https://neo4j.com/

of the tool used in this step?". We design 17 such traversal templates to cover a wide range of procedural reasoning types. For each template, we generate multiple question variants using GPT-4o-mini, followed by manual filtering to ensure quality and clarity. These templates form the backbone for producing diverse yet semantically consistent question-answer instances. Figure 1 presents examples of Traversal Templates alongside their corresponding questions. The complete set of 17 question templates with example questions is provided in (Nguyen et al. 2025).

Question and Answer Generation We generate questions by aligning each video segment with its corresponding `Task` and `Step` nodes in PKG, then apply traversal templates to form questions and retrieve answers. For each question, we sample one correct answer and four distractors. To ensure balanced answer distributions and reduce bias, we apply sampling strategies that equalize the frequency of each answer candidate across correct and incorrect options. Each question is paired with a Cypher query that retrieves supporting facts from PKG. These structured annotations serve as logical forms for KG-based evaluation, model supervision, and reasoning trace analysis.

PKR-QA is designed for scenarios with limited training data but having access to structured knowledge, such as a knowledge graph. We construct a training set of 1,700 samples (100 per traversal template) and a validation set of 850 samples (50 per template). The test set contains 46,921 questions, generated from all video segments in the COIN test split. This setup supports zero- and few-shot generalization.

To assess the quality of our dataset, we evaluate whether a question can be reasonably answered by a human—referred to as *plausibility*. We conducted a case study with eight participants, each answering a subset of 170 questions. For each question, participants were shown a video segment, the question, and five answer options, and were asked to select the correct answer. Each question was independently answered by three participants. A question is considered plausible if at least one participant selected the correct answer. Using this criterion, we found that 92.4% of the questions are plausible. Additionally, random baselines perform at chance level ($\sim$ 20% accuracy), suggesting low annotation artifacts or answer biases.

Knowledge Module Learning

In this section, we introduce our proposed *Knowledge Module Learning* (KML) for procedural knowledge reasoning, which effectively handles uncertain (probabilistic) inputs while producing interpretable reasoning outputs using a small number of trainable parameters. We train a collection of neural networks known as Neural Knowledge Module (**KM**) to represent each binary relation type of PKG. Then given a video and a question, we ask LLM to generate a program consisting of KM invocations to answer the questions. Then we execute those KMs sequentially with the grounded evidence extracted from the video to obtain the final answer. Next, we present the details.

Knowledge Module Learning. For each binary relation type $\mathcal{R}_k(\mathcal{E}_i, \mathcal{E}_j)$ in the PKG that maps from entity type $\mathcal{E}_i$ to $\mathcal{E}_j$, we learn a KM as a learnable neural relation ($\phi_{\mathcal{R}_k}$) with parameters ($\theta_{\mathcal{R}_k}$). KM learns to map from a given head entity $e_i \in \mathcal{E}_i$ to the corresponding set of tail entities $\mathcal{I}_j$ where $\mathcal{I}_j \subseteq \mathcal{E}_j$ of relation $\mathcal{R}_k$ as follows:

$$\phi_{R_k}(\, x(e_i;\, \theta_x);\, \theta_{R_k}) \rightarrow \mathbf{e_j} \tag{1}$$

where $x(e_i;\theta_x)$ is a d-dimensional embeddings of the head entity and $\mathbf{e_j}$ is a vector that is closer to $x(e_j)$ for all $e_j \in \mathcal{I}_j$. Here, $\mathcal{I}_j$ is the set of tail entities under relation R_k for the head entity e_i or formally $\mathcal{I}_j := \{e_j \in \mathcal{E} \mid R_k(e_i, e_j) = True\}$. Then the objective of KM learning is to make sure to learn (θ_{R_k}, θ_x) such that for each head entity embedding $x(e_i)$ can be mapped to a vector $\mathbf{e}_j$ that is representative of all corresponding tail entities $\mathcal{I}_j$. We iterate over all the triplets (including the inverse triplets) of all relation types of PKG in a batch-learning manner and train the KMs using the contrastive loss.

$$-log\frac{exp(\frac{\mathbf{e_j}\cdot x(e_j)}{\tau})}{\sum_{\forall e_p \in \mathcal{B}} exp(\frac{\mathbf{e_j}\cdot x(e_p)}{\tau})} \tag{2}$$

Here $\mathcal{B}$ is the set of tail entities in the batch of triplets excluding $\mathcal{I}_j$, $e_j \in \mathcal{I}_j$ is one of the positive target entity for e_i under R_k, and τ is the temperature. Note that we L2 normalize all embeddings and the vector $\mathbf{e_j}$ before computing the contrastive loss. Contrastive loss plays a crucial role in learning neural relation functions ϕ_{R_k} (i.e. KMs), for modelling symbolic binary relations of the form $\mathcal{R}_k(\mathcal{E}_i, \mathcal{E}_j)$.

The embedding learning function $x(;\theta_x)$ is implemented using CLIP (Radford et al. 2021) text encoder embeddings with frozen parameters (θ_x). Alternatively, we also learn the embedding function from scratch using standard implementations[2]. Similarly, we also learn the inverse KM for each inverse relation $\mathcal{R}^*{}_k(\mathcal{E}_j, \mathcal{E}_i)$ of each relation $\mathcal{R}_k$. Let us denote the set of all KMs by $\phi_R = \{\phi_{R_k} | k = 1, \cdots\}$.

At inference, KM takes an input embedding and maps that to an output embedding that represents a set of corresponding tail entities of that relation. For example, Table 3 shows the semantic meaning of each output embedding that maps to a set of tail entities of relation `HAS_TOOL` for given input embedding of the Step entity. We experimented with different neural configurations of multi-layered perceptron (MLP) and found that two-layered MLP with *Tanh* activation performs the best for learning KMs. Next, we present how to answer questions using KMs and LLM generated programs.

Question Answering. Given the video v, the question Q, and options $Y = \{y_1, \cdots, y_n\}$ ($n = 5$) we prompt a LLM $\phi()$ to find the relevant entity type (E_g) that should be grounded in the video to answer the question.

$$\phi(Q, \text{PKGS}) \rightarrow E_g \tag{3}$$

For example, E_g can be a task, step, object or action. Then we find the entity instance(s) that is present in the given video V of the entity type E_g using a vision foundation model (VLM) as follows:

$$\text{VLM}(V, E_g) \rightarrow \mathbf{x_e} \tag{4}$$

[2]We learn these using torch.nn.Embedding

where $\mathbf{x_e} \in \mathcal{R}^{\mathcal{C}}$ represents the score vector for the entity type (e.g., step distribution) across all categories of that entity type. We assume there are C categories for entity type E_g. To obtain estimates about the grounding entity E_g, we use *ProceduralVRL* (P.VRL) (Zhong et al. 2023) a VLM tailored for procedural tasks.

Given the collection of Knowledge Module names (ϕ_R) and the question Q, and the grounded entity type E_g, we use LLM (denoted by $\phi_G()$) to generate a list of sequence of Knowledge module invocations (known as program or P_1) to answer the question. Using a similar concept to chain-of-thought (Wei et al. 2022; Wang et al. 2022), tree-of-thought (Yao et al. 2023), and graph-of-thought (Besta et al. 2024) we ask the LLM to generate multiple alternative programs to answer the same questions.

$$\phi_G(Q, \phi_R, E_g) = [P_1 = \langle \phi_{r_i}, \phi_{r_j}, \phi_{r_k}, ... \rangle, \cdot, \quad P_l = \langle \phi_{r_i}, \phi_{r_j}, \phi_{r_k}, ... \rangle] \tag{5}$$

Here P_l is the specific module invocations and each P_l may result in different answers following an alternative-thought of reasoning approach. The LLM module $\phi_G()$ can be implemented with any LLM with good reasoning capability. We use a special prompt that invokes deep understanding and consistent knowledge graph traversal and alternative thought invocations– see (Nguyen et al. 2025) for more details on the prompt. As an example, to answer the question `what is an alternative tool that can be used for this step?` it generates the following program of modules.

```
HAS_TOOL(Step) → Tool
HAS_PURPOSE(Tool) → Purpose
SIMILAR_PURPOSE(Purpose) → Purpose
PURPOSE_TO_TOOL(Purpose) → Tool
```

where PURPOSE_TO_TOOL(Purpose) is the inverse relation of HAS_PURPOSE(Tool). LLM may generate multiple programs that leads to the answer, typically 3-5 alternatives for complex problems. To predict the answer, we execute each program in order, inputting the grounded entity representation z_i into the first module. For each program (P_1 to P_l), z_i is fed into the first module to obtain intermediate embedding, invoking all modules sequentially. For example $z_j = \phi_{r_i}(z_i)$ then $z_k = \phi_{r_j}(z_j)$ and finally $z_f = \phi_{r_k}(z_k)$. Therefore the *final embedding* representing the answer to the given question is z_f. We can also inspect the meaning of each intermediate representation as shown in Table 3 allowing more interpretable reasoning that can handle uncertain inputs.

To compute the first input embedding z_i, we use the grounded top-K entity instance and weight the embeddings of each grounded entity instance category name as follows:

$$z_i = S \times \mathbf{X}_e{}^T \tag{6}$$

$$\mathbf{X}_e = [x(e_1), x(e_2), \cdots, x(e_k)] \tag{7}$$

where e_k is the k-th category of the entity type E_g and $S = [s_1, s_2, \cdots, s_k]$ is a vector containing top-k scores for each grounded entity-type category (e.g. Step categories).

Inference. Given the options $Y = \{y_1, \ldots, y_n\}$, we obtain the embeddings of Y, i.e., $x(Y) = [x(y_1), x(y_2), \cdots x(y_n)]$. Then we compute the cosine similarity between the *final embedding* z_f and $x(Y)$ and apply softmax to predict the answer index for each program. When there are alternative programs, we take the maximum score from all programs as the final answer.

VQA Training for KML. We also fine-tune the KMs using a few examples of the video-question-answer using our dataset. We compute the cosine similarity between z_f and $x(Y)$ and apply softmax to predict the answer index.

$$\hat{y} = softmax\left(cosine\_sim\left(x(Y), z_f\right)\right) \tag{8}$$

Then we train all KMs jointly using the cross-entropy loss over the correct answer index. At test time, we predict the right answer using the argmax of the cosine similarity scores. One of the advantages of KML (auto-program) approach is that, given the relation types, we do not need to manually select any neural modules, or generate programs manually. We use LLMs such as GPT, DeepSeek or Mistral. More implementation details of KML is presented in (Nguyen et al. 2025).

Experiments

Experimental Setup

We conduct experiments on the PKR-QA benchmark to evaluate the effectiveness of VLMs and Neurosymbolic methods for procedural knowledge-based question answering. We use NVIDIA A100 GPUs (80GB VRAM) for conducting experiments with VLMs, NVIDIA GeForce RTX 2080 Ti and A5000 GPUs for training KML, and RTX 3090 GPUs (24GB VRAM) for training knowledge graph embedding methods.

VLM Evaluation Settings. We evaluate VLMs under five different settings to assess their procedural knowledge reasoning ability: (1) **Zero-shot**: VLMs take a video segment (8 uniformly sampled frames), a question, and options as input; (2) **VLM+P.VRL**: VLMs take a video segment (8 uniformly sampled frames), a question, options, and the top-5 step/task categories from P.VRL as input; (3) **KG-training**: VLMs are tuned using the triplet instances of our PKG as questions using LoRA (Hu et al. 2021) and then evaluate the fine-tuned VLMs with PKR-QA; (4) **QA training**: VLMs are tuned using LoRA for 100 epochs with 4 randomly sampled frames as input, demonstrating the impact of few-shot fine-tuning. All models are trained using the training set of 1,700 question-answer pairs; (5) **KG+QA training**: VLMs are tuned using triplet instances following KG-training, and 1,700 question-answer pairs as described in QA training.

Neurosymbolic Methods. We explore three variants of KML, differing in how the embedding function $x(;\theta_x)$ is implemented. In **KML-F-CLIP**, $x(;\theta_x)$ is a frozen CLIP text encoder. In **KML-CLIP**, we initialize $x(;\theta_x)$ with CLIP embeddings and fine-tune its parameters. In **KML-Rand**, θ_x is learned from scratch (i.e., randomly initialized). We compare our KML against the following NS methods:

Inference by Graph Propagation (IGP):We implement a simple NS baseline that uses a given program and a set of grounded entities with associated probabilities or logits to answer questions. Given a directed knowledge graph with binary relations, we propagate uncertainty from grounded entities through the relations specified in the program. Using breadth-first traversal, we accumulate scores at each target entity by summing the propagated logits. This approach resembles probabilistic logic-based inference.

KG Embedding Methods: We compare against standard KG embedding models, including TransE (Bordes et al. 2013), TransH (Wang et al. 2014), and RotatE (Sun et al. 2019). After training, we use LLM-generated programs (as in KML) to perform multi-hop reasoning. These models are selected for their support of compositional reasoning. Embedding dimensions are tuned on a validation set, yielding optimal sizes of 256 for TransE, 100 for TransH, and 256 for RotatE. We also include variants with CLIP-initialized entity and relation embeddings to enable direct comparison with CLIP-based KML models.

Modern NS methods: We compare with modern NS methods including ViperGPT (Surís, Menon, and Vondrick 2023) that uses the power of LLM and vision models, and MAC (Hudson and Manning 2018). We evaluate MAC on a classification-based VQA task, using a single image frame and a question (without answer choices) as input. The model predicts from 2,079 answer classes aggregated from the dataset. We use GloVe (Pennington, Socher, and Manning 2014) embeddings for text and extract visual features with a pretrained ResNet101 (He et al. 2016). We tuned the embedding size, MAC hidden size, and the number of MAC layers, selecting the best setup based on validation performance. The comparison with NS methods such as (Jaiswal et al. 2025; Li et al. 2025) are left for future work.

Metrics. Since VLM-generated text may not exactly match the predefined multiple-choice options, we adopt the filtering and MCQ accuracy computation strategy from (Yue et al. 2024; Lin et al. 2023). We report both overall accuracy and mean accuracy for each model. **Accuracy** is computed as the average score across all test samples, while **mean accuracy** (mAcc)s is the average of per-template accuracies, providing equal weight to each traversal template.

Analysis and Discussion

Benchmarking VLMs on PKR-QA. Table 1 compares the performance of various VLMs across five training and inference settings, revealing several key insights. **Providing predicted step and task** information from ProceduralVRL (VLM+P.VRL) consistently improves performance over the **zero-shot** setting. These gains highlight the importance of grounded procedural context in enhancing reasoning, even for strong models like MiniCPM-V and Qwen2.5-VL. **Training on KG triplets** yields some improvement over zero-shot baselines, though the gains are modest and less consistent, suggesting that aligning symbolic representations with multimodal inputs remains non-trivial. **QA-based fine-tuning** leads to larger improvements, particularly for MiniCPM-V and Qwen2.5-VL. These gains indicate that VLMs are capable of adapting to task-specific supervision, even when provided with a relatively small number of QA pairs (1,700 samples). The best performance is achieved when **combining both KG and QA training**, with

Setting	**Model**	**Acc**	**mAcc**
Zero-shot	DeepSeek-VL2 (27.4B) (Wu et al. 2024)	58.4	55.4
	MiniCPM-V (8B) (Yao et al. 2024)	62.6	59.7
	mPLUG-Owl3 (7B) (Ye et al. 2024)	$\underline{63.1}$	$\underline{60.2}$
	Qwen2.5-VL (7B) (Bai et al. 2025)	59.6	57.8
	VideoChat2-HD (7B) (Li et al. 2023)	61.2	58.4
VLM + P.VRL	DeepSeek-VL2 (27.4B) (Wu et al. 2024)	64.5	59.9
	MiniCPM-V (8B) (Yao et al. 2024)	67.4	63.8
	mPLUG-Owl3 (7B) (Ye et al. 2024)	65.5	61.6
	Qwen2.5-VL (7B) (Bai et al. 2025)	$\underline{69.4}$	$\underline{65.8}$
	VideoChat2-HD (7B) (Li et al. 2023)	65.5	59.9
KG-training	MiniCPM-V (8B) (Yao et al. 2024)	63.5	61.0
	mPLUG-Owl3 (7B) (Ye et al. 2024)	64.8	61.4
	Qwen2.5-VL (7B) (Bai et al. 2025)	$\underline{67.3}$	$\underline{64.1}$
QA training	MiniCPM-V (8B) (Yao et al. 2024)	71.1	71.4
	mPLUG-Owl3 (7B) (Ye et al. 2024)	71.8	72.4
	Qwen2.5-VL (7B) (Bai et al. 2025)	$\underline{73.6}$	$\underline{73.4}$
KG+QA training	MiniCPM-V(8B) (Yao et al. 2024)	72.1	72.1
	mPLUG-Owl3 (7B) (Ye et al. 2024)	73.1	**73.8**
	Qwen2.5-VL (7B) (Bai et al. 2025)	**74.2**	**73.8**

Table 1: Comparison of VLMs in different settings. Underlined scores denote the best-performing method within each setting, while bold scores highlight the best overall.

Setting	**Model**	**Acc**	**mAcc**
No Training / No Program	ViperGPT	41.6	40.9
	GPT-4o + P.VRL	$\underline{71.2}$	$\underline{69.0}$
Program Only	IGP (+P.VRL)	62.8	60.0
QA training Only	MAC	11.6	20.0
KG-training	TransE	63.6	51.6
	TransH	73.1	66.3
	RotatE	41.6	29.2
	TransE+CLIP	56.8	45.4
	TransH+CLIP	70.6	65.9
	RotatE+CLIP	48.8	35.2
	KML-F-CLIP (Ours)	74.6	$\underline{71.6}$
	KML-Rand (Ours)	73.5	70.0
	KML-CLIP (Ours)	$\underline{75.3}$	71.5
KG+QA training	KML-F-CLIP (Ours)	76.7	75.3
	KML-Rand (Ours)	77.4	76.3
	KML-CLIP (Ours)	**78.1**	**77.1**

Table 2: Performance comparison of NS methods.

Qwen2.5-VL reaching 74.2/73.8, indicating that integrating structured procedural knowledge with task-specific examples provides complementary benefits for enhancing procedural understanding in VLMs.

Benchmarking Neurosymbolic Methods. As shown in Table 2, all KML variants outperform all baselines, confirming the benefit of executing LLM-generated KG-traversal programs with relation-specific KMs for procedural reasoning QA. KML achieves the highest performance under KG+QA training using KML-CLIP variant. The performance of KML-Rand is not far from KML-F-CLIP and interestingly, when tuned with KG+QA training the KML-Rand performs better than KML-F-CLIP under the same setting. **ViperGPT** performs poorly on our benchmark due to the absence of built-in reasoning operators and reliance on predefined program templates that are not suited for the structured reasoning required in procedural knowledge tasks. **IGP** suffers from combinatorial explosion and poor grounding, often generating irrelevant results from uncertain starting nodes. In contrast, KML handles such uncertainty better by grounding traversal via learned knowledge modules. **MAC**, originally designed for visual reasoning tasks (e.g., answering questions like “What is to the left of the green box?”), fails in our setting due to its lack of access to external procedural knowledge. Among **KG embedding methods**, TransE and TransH perform better due to their additive or projective composition, aligning well with our program-based reasoning. RotatE underperforms, likely because its complex-valued, rotation-based composition is less effective for multi-hop reasoning. CLIP-based initialization yields mixed results—improving RotatE but degrading TransE and TransH—indicating varying alignment with visual semantics across models. We also evaluate **GPT-4o** using top-5 predicted step/task category names from P.VRL. It achieves 71.2% accuracy and 69.0% mean accuracy.

KML’s Interpretability. One key advantage of KML is its interpretability, as illustrated in the qualitative example in Table 3. We observe step-by-step reasoning and intermediate interpretations from the learned embeddings, offering insight into the model’s decision process. The output entities represented by the output vectors of each KM seems reasonable and accurate for the given task.

KML’s Generalizability. To assess KML’s generalizability, we added ten binary relations from the **STAR benchmark**. KML-F-CLIP achieved 74.9% on Interaction and 76.7% on Feasibility questions, outperforming prior bests of 71.8% (Jaiswal et al. 2025) and 62.4% (Yu et al. 2023). For Sequence and Prediction questions, it scored 57.3% and 49.8%, respectively. We also trained KML-F-CLIP using a **GPT-4o-generated KG** from 7,687 WikiHow tasks (57,027 steps, 8.7M triplets) capturing tools, actions, objects, and purposes. Evaluated on our PKR-QA dataset, it achieved 73.9% mean accuracy, rising to 74.9% with KG+QA training, suggesting that while generic KGs help, domain-specific modules remain advantageous.

KML’s Robustness. Figure 2 (left) shows KML’s performance using top-k step/task predictions ($k = 1$ to 5). Using all top-5 predictions yields the best QA performance, while top-1 predictions still achieve strong results, demonstrating the model’s robustness to imperfect inputs. Figure 2 (right) reports a moderate correlation (0.24) between step prediction accuracy and QA accuracy, suggesting KML does not heavily depend on input quality, benefiting from embedding-based reasoning and generalizable knowledge modules.

KML Ablation. We train KML modules from scratch using only QA training, with KML-F-CLIP achieving 59.3% mean accuracy—highlighting the value of training on PKG data. KML allows exploration of multiple programs per question (see (Nguyen et al. 2025)), though gains over single-program use are marginal. It also supports expert program editing for improved reliability. Evaluating different LLMs for program generation in KML-F-CLIP, GPT-4o leads with 71.6% accuracy, followed by LLaMA-3-8B (68.4), DeepSeek-V2.5 (66.5), Mistral-7B (66.2), and Qwen-2.5 (63.7), showing KML’s robustness across LLMs.

	Q. What is the other task that use the tool in this video for the same purpose?	Task: Make Orange Juice Step: pour the orange juice into the cup
HAS_TOOL	**TOOL_TO_STEP**	**STEP_TO_TASK**
Out=**Tool**	Out=**Step**	Out=**Task**
cup (0.361)	pour into the ingredients (0.337)	MakeCookie (0.221)
mug (0.273)	pour in after mix it (0.314)	MakeCocktail (0.208)
measuring cup (0.253)	add some ingredients to the tea (0.308)	MakeHomemadeIceCream (0.191)
yogurt (0.249)	add some ingredients in the coffee (0.307)	MakeChocolate (0.188)
bottle (0.223)	pour the ingredients into the bowl (0.293)	MakeCoffee (0.185)

Table 3: Three-hop Reasoning using KML-F-CLIP: The step-by-step reasoning outputs of KML with estimated probability value over the domain of relation using embeddings.

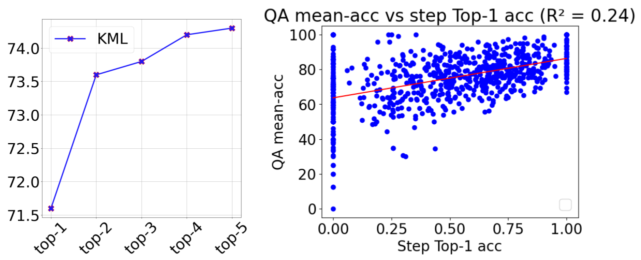


Figure 2: (Left) QA performance of KML-F-CLIP using top-1 to top-5 grounded entities from P.VRL. (Right) Correlation between step prediction accuracy and QA accuracy.

Conclusion

Our PKR-QA is a benchmark for procedural knowledge reasoning where VLMs demonstrate capability in procedural knowledge. However, their closed-loop reasoning lack the transparency, controllability, and domain-specific constraints required for mission-critical applications such as medical procedures or industrial automation. In contrast, the proposed Knowledge Module Learning (KML) framework explicitly grounds reasoning in procedural knowledge graphs and decouples hypothesis generation (via LLM-based program synthesis) from structured execution. By constraining reasoning to predefined operations, KML ensures reliability and interpretability without sacrificing performance. Our experiments highlight the promise of such neurosymbolic architectures, where lightweight, domain-aware modules (trained directly on KG relations) performs well in knowledge-intensive reasoning.

Acknowledgments

This research is supported by the National Research Foundation, Singapore, under its NRF Fellowship (Award# NRF-NRFF14-2022-0001) and by funding allocation to Basura Fernando by the A*STAR under its SERC Central Research Fund (CRF), as well as its Centre for Frontier AI Research.

References

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.

Ashutosh, K.; Ramakrishnan, S. K.; Afouras, T.; and Grauman, K. 2024. Video-mined task graphs for keystep recognition in instructional videos. *Advances in Neural Information Processing Systems*, 36.

Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Song, S.; Dang, K.; Wang, P.; Wang, S.; Tang, J.; et al. 2025. Qwen2. 5-VL Technical Report. *arXiv preprint arXiv:2502.13923*.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17682–17690.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Chen, W.; Gan, Z.; Li, L.; Cheng, Y.; Wang, W.; and Liu, J. 2021. Meta module network for compositional visual reasoning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 655–664.

Choudhury, R.; Niinuma, K.; Kitani, K. M.; and Jeni, L. A. 2024. Video Question Answering with Procedural Programs. In *European Conference on Computer Vision*, 315–332. Springer.

Endo, M.; Hsu, J.; Li, J.; and Wu, J. 2023. Motion question answering via modular motion programs. In *International Conference on Machine Learning*, 9312–9328. PMLR.

Francis, N.; Green, A.; Guagliardo, P.; Libkin, L.; Lindaaker, T.; Marsault, V.; Plantikow, S.; Rydberg, M.; Selmer, P.; and Taylor, A. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*, 1433–1445.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hoang, L.; Liausvia, F.; Liu, Y.; and Nguyen, T.-S. 2024. Semi-automated Construction of Complex Knowledge Base Question Answering Dataset Using Large Language Model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 230–248. Springer.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hudson, D.; and Manning, C. D. 2019. Learning by abstraction: The neural state machine. *Advances in neural information processing systems*, 32.

Hudson, D. A.; and Manning, C. D. 2018. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. GPT-4o System Card. *arXiv preprint arXiv:2410.21276*.

Jaiswal, S.; Roy, D.; Fernando, B.; and Tan, C. 2025. Learning to Reason Iteratively and Parallelly for Complex Visual Reasoning Scenarios. *Advances in Neural Information Processing Systems*, 37: 137965–137998.

Johnson, J.; Hariharan, B.; Van Der Maaten, L.; Hoffman, J.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision*, 2989–2998.

Lei, J.; Yu, L.; Berg, T. L.; and Bansal, M. 2019. Tvqa+: Spatio-temporal grounding for video question answering. *arXiv preprint arXiv:1904.11574*.

Li, C.; Sugandhika, C.; Ee, Y. K.; Peh, E.; Zhang, H.; Yang, H.; Rajan, D.; and Fernando, B. 2025. IMoRe: Implicit Program-Guided Reasoning for Human Motion Q&A. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12987–12996.

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

Lin, B. Y.; Chen, X.; Chen, J.; and Ren, X. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*.

Lin, J.; Yin, H.; Ping, W.; Lu, Y.; Molchanov, P.; Tao, A.; Mao, H.; Kautz, J.; Shoeybi, M.; and Han, S. 2023. VILA: On Pre-training for Visual Language Models. arXiv:2312.07533.

Ma, Y.; Wang, Y.; Wu, Y.; Lyu, Z.; Chen, S.; Li, X.; and Qiao, Y. 2022. Visual knowledge graph for human action reasoning in videos. In *Proceedings of the 30th ACM International Conference on Multimedia*, 4132–4141.

Mascharka, D.; Tran, P.; Soklaski, R.; and Majumdar, A. 2018. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4942–4950.

Min, J.; Buch, S.; Nagrani, A.; Cho, M.; and Schmid, C. 2024. Morevqa: Exploring modular reasoning models for video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13235–13245.

Nguyen, T.-S.; Yang, H.; Neoh, T. Y.; Zhang, H.; Keat, E. Y.; and Fernando, B. 2025. Neuro Symbolic Knowledge Reasoning for Procedural Video Question Answering. *arXiv preprint arXiv:2503.14957*.

Parmar, P.; Peh, E.; Chen, R.; Lam, T. E.; Chen, Y.; Tan, E.; and Fernando, B. 2024. Causalchaos! dataset for comprehensive causal action question answering over longer causal chains grounded in dynamic visual scenes. *Advances in Neural Information Processing Systems*, 37: 92769–92802.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of* *the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Perez, E.; Strub, F.; De Vries, H.; Dumoulin, V.; and Courville, A. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Sadhu, A.; Gupta, T.; Yatskar, M.; Nevatia, R.; and Kembhavi, A. 2021. Visual semantic role labeling for video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5589–5600.

Shah, M.; Cahoon, J.; Milletari, M.; Tian, J.; Psallidas, F.; Mueller, A.; and Litombe, N. 2024. Improving LLM-based KGQA for multi-hop Question Answering with implicit reasoning in few-shot examples. In Biswas, R.; Kaffee, L.-A.; Agarwal, O.; Minervini, P.; Singh, S.; and de Melo, G., eds., *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, 125–135. Bangkok, Thailand: Association for Computational Linguistics.

Speer, R.; Chin, J.; and Havasi, C. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Surís, D.; Menon, S.; and Vondrick, C. 2023. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11888–11898.

Tang, Y.; Ding, D.; Rao, Y.; Zheng, Y.; Zhang, D.; Zhao, L.; Lu, J.; and Zhou, J. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1207–1216.

Wang, A.; Wu, B.; Chen, S.; Chen, Z.; Guan, H.; Lee, W.-N.; Li, L. E.; and Gan, C. 2024. SOK-Bench: A Situated Video Reasoning Benchmark with Aligned Open-World Knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13384–13394.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wu, Z.; Chen, X.; Pan, Z.; Liu, X.; Liu, W.; Dai, D.; Gao, H.; Ma, Y.; Wu, C.; Wang, B.; et al. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Yao, Y.; Yu, T.; Zhang, A.; Wang, C.; Cui, J.; Zhu, H.; Cai, T.; Li, H.; Zhao, W.; He, Z.; et al. 2024. MiniCPM-V: A GPT-4V Level MLLM on Your Phone. *arXiv preprint arXiv:2408.01800*.

Ye, J.; Xu, H.; Liu, H.; Hu, A.; Yan, M.; Qian, Q.; Zhang, J.; Huang, F.; and Zhou, J. 2024. mplug-owl3: Towards long image-sequence understanding in multi-modal large language models. In *The Thirteenth International Conference on Learning Representations*.

Yu, S.; Cho, J.; Yadav, P.; and Bansal, M. 2023. Self-chained image-language model for video localization and question answering. *Advances in Neural Information Processing Systems*, 36: 76749–76771.

Yue, X.; Ni, Y.; Zhang, K.; Zheng, T.; Liu, R.; Zhang, G.; Stevens, S.; Jiang, D.; Ren, W.; Sun, Y.; Wei, C.; Yu, B.; Yuan, R.; Sun, R.; Yin, M.; Zheng, B.; Yang, Z.; Liu, Y.; Huang, W.; Sun, H.; Su, Y.; and Chen, W. 2024. MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI. In *Proceedings of CVPR*.

Zhong, Y.; Yu, L.; Bai, Y.; Li, S.; Yan, X.; and Li, Y. 2023. Learning procedure-aware video representation from instructional videos and their narrations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14825–14835.

Zhong, Z.; Zhong, L.; Sun, Z.; Jin, Q.; Qin, Z.; and Zhang, X. 2024. SyntheT2C: Generating Synthetic Data for Fine-Tuning Large Language Models on the Text2Cypher Task. *arXiv preprint arXiv:2406.10710*.

Zhou, H.; Martín-Martín, R.; Kapadia, M.; Savarese, S.; and Niebles, J. C. 2023. Procedure-aware pretraining for instructional video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10727–10738.