Contents

A Discussion

A.1 Limitation

Programmatically generated tasks can be unrealistic and biased. Programmatically generated tasks can lack the complexity and variability found in real-world data. These tasks might not capture the nuances of real-world scenarios, leading to models that perform well on synthetic data but fail in practical applications. The constraints and rules defined in the code may oversimplify the tasks, making them easier for models to solve compared to real-world tasks. This can result in overestimating a model's capabilities. The rules and logic used to generate tasks can inadvertently introduce biases. For example, if the code disproportionately generates certain types of objects or scenarios, the model may not be adequately tested on a diverse range of tasks.

Designing the task space is challenging. Identifying and defining the relevant attributes for each task type (e.g., object recognition) requires deep domain knowledge and understanding of what aspects are critical for evaluating model performance. The task space must be comprehensive enough to cover various scenarios but not so complex that it becomes infeasible to manage or evaluate. Striking this balance is a significant challenge. The task space should be designed to ensure comprehensive coverage of all relevant scenarios and diversity in the types of tasks. This requires meticulous planning and consideration of all possible task variations.

Adding new task generators requires coding skills. Adding new task generators involves programming and understanding the underlying framework used for task generation. This requires technical expertise, which may not be available for all communities and can be a barrier for non-technical researchers who might have valuable insights and ideas for new tasks but lack the coding ability to implement them.

Query results approximation can be inaccurate. Efficient query results approximation within certain budgets might sometimes yield inaccurate results, especially when the budget limits are constrained. This inaccuracy can stem from several factors. First, the models that embed tasks into vectors may not fully capture all the details and nuances between different tasks. Second, the algorithms used for querying might have inherent limitations or room for improvement, affecting the precision of the results. Addressing these issues requires ongoing refinement of both the task embedding models and the query algorithms to enhance their ability to deliver accurate approximations under varying computational budgets.

A.2 Potential negative social impact

Misuse for malicious benchmarks. TASKVERSE's ability to generate a vast number of tasks could be misused to create benchmarks specifically designed to trick or expose vulnerabilities in AI systems. Malicious actors might use this capability to create benchmarks that mislead researchers or lead to the development of AI models with undesirable biases or vulnerabilities.

Reinforcement of biases and discrimination. If TASKVERSE's task generators are not carefully designed and curated, they could inadvertently perpetuate existing biases present in the source data. This could lead to the development of AI models that are biased against certain groups of people or perpetuate harmful stereotypes.

Overreliance on synthetic tasks. The focus on synthetic task generation could lead to a disconnect between evaluation results and real-world performance. Overreliance on synthetic tasks might create a false sense of progress and hinder the development of AI models that can effectively address real-world challenges.

Data contamination. Fine-tuning models on synthetic tasks generated by TASKVERSE could lead to data contamination, where the model learns to exploit the specific patterns and biases of the synthetic data rather than generalizing to real-world scenarios. This could result in models that perform well on synthetic benchmarks but poorly in practical applications.

Access and fairness. While TASKVERSE aims to democratize AI evaluation, the technical expertise required to implement new task generators could create barriers for researchers and practitioners from underrepresented groups, leading to a lack of diverse perspectives and potentially reinforcing existing inequalities.

A.3 Future work

Supporting natural language user queries. We plan to enable natural language queries, allowing users to specify evaluation needs in plain language. This will leverage language models to translate instructions into actionable query commands, making the system more accessible and user-friendly. This enhancement will democratize access to model evaluation, streamline the process, and reduce barriers for non-technical users, fostering a more inclusive evaluation ecosystem.

Expanding the TASKVERSE system. To further enhance the capabilities of TASKVERSE, we plan to extend it across a broader range of scenarios and model types. This involves integrating support for various generative models, including language models and visual generative models, which can fine-tune the evaluation of generation quality. Also, by incorporating new types of source data, we aim to enrich the diversity and relevance of the tasks generated, ensuring that the evaluation framework remains robust and comprehensive as foundation model capabilities advance. Additionally, developing new task generators will enable the creation of tasks that capture emerging AI challenges and applications, facilitating continuous adaptation to the evolving landscape of AI. This expansion will empower users from different domains to evaluate models in ways that are highly specific to their needs, ultimately contributing to more targeted and effective deployment of AI technologies.

A new workload for database study. TASKVERSE presents new opportunities for the database community to develop efficient query execution techniques on conceptual relations containing model inference results (e.g., task accuracy of many models on many tasks) that are expensive to compute and often unmaterialized when a query is issued. The idea of pre-filtering to avoid expensive computation has been proven to be effective in some database problems, such as accelerating similarity joins [31, 19] and video analytics queries [22] where computing the similarity function or running model inference on videos is expensive during query execution. In a similar vein, recent work [15, 14, 37] has proposed efficient database indexing and query execution techniques to navigate the tradeoffs between storing the model inference results on disk and computing them on-the-fly at query time. Some other efforts [3] have also proposed trading off query result accuracy for query response time. Another direction for future work is query result diversification. When a practitioner explores a set of MLMs, datasets, and tasks, they may desire to examine a diverse set of result items, e.g., tasks that are dissimilar. It would be interesting to how query result diversification techniques [10, 16] could be adapted in TASKVERSE's setting.

B Details of Task Generation

In this section, we describe the details of the programmatic task generation process in TASKVERSE. We focus on tasks of multiple-choice visual questions answering, including both image question answering (ImageQA) and video question answering (VideoQA).

B.1 Key concepts

First, we introduce several key concepts and definitions in our task generation process.

Task instance, task, and task plan. A task instance is an image/video, question, options, and ground truth answer tuple that comprises a single evaluation test-case. A task is a conceptual abstraction consisting of all task instances that share the same question and answer. Tasks are specified via task plans, which contain the required task metadata and configurations to create the actual task instances. For example, in tasks involving counting, the task plan specifies the categories of objects, their total numbers in the scene, and their positions in the image—such as two apples, one on the top right and one on the bottom left. The task instance then features an actual image of the target objects and includes a specific question and answer that is consistent with the arrangement of these objects in the scene. One such task instance might be an image with two apples, the question: "How many apples are there in the image?", and the answer: "2". Multiple task instances can be generated from a single task plan because other elements such as the image background and types of distractor objects can be randomized, as they are not specified in the task plan.

Source data. We refer to source data as the visual data and annotations that are used to generate task instances, *e.g.*, the 3D objects from Objaverse [8, 7] and their associated annotations or the real images and scene graphs from GQA [17, 23].

Task generator. Each task generator is a program that, given source data as input, generates task instances of a certain type. It achieves three main purposes: 1) it defines the schema of the task plan; 2) it can enumerate all possible task plans given the available source data; and 3) given source data and a specific task plan, it can randomly generate a task instance belonging to the task family defined by the task plan.

B.2 The generation process

Given the source data and a task generator, one can readily generate a large number of tasks. The overall generation process consists of the following steps:

Step 1: enumerate the task plans. Once the task generator is implemented, one can use it to enumerate and return all the possible task plans based on the defined schema and the source data. As each task plan consists of just the metadata of the task rather than the actual task instances, it is efficient to enumerate all the task plans and store them as a single table. Note that enumerating all possible task plans is a one-time job, since the table of task plans can be stored and reused.

Step 2: generate task instances of a task given its task plan. Another core functionality of the task generator is to generate one task instance given a valid task plan. Note that the task generator may generate many different task instances because of the randomness, *e.g.*, the negative choices can be randomly sampled from possible candidates, yet since they are all generated by the same task generator with the same task plan, they would share the question and ground truth answer and are considered belonging to the same task.

Properties. This task generation process exhibits several key properties:

- **Reproducible:** With our task generation process, the tasks are produced as a combination of the source data and the programs, therefore one can reproduce identical task instances with the same source data and the random seed of the program.
- **Scalable:** This task generation process is scalable for two reasons. First, it is *memory-friendly*. One only needs to store the source data and the annotations, as well as our

codebase. Even when one aims to evaluate a model on millions of task instances, since the task instances are reproducible, one can choose to generate the task instances on the fly rather than beforehand. Secondly, it is *easy to expand* the space of task that can be generated. One can increase the number of possible tasks by either adding new source data or new task generators.

- **Easy to update:** Benchmarks can contain unexpected errors, *e.g.*, annotation error [33], so the task generation process must be easy to update once the error is caught. Since our task generation process is transparent to the users, once an error is caught, it can immediately be attributed to either the error of the source data or bugs in the code of the task generators, and then be fixed. We welcome the whole community to report any flaw in our task generation process.
- **Structured task space:** Finally, each task generated by our approach is associated with a task plan composed of its metadata. This design offers a natural structure for the tasks so that they can be grouped by certain specifications of task metadata. It enables users to navigate wanted tasks by querying the table of task plans as querying a normal database. Also, it facilitates the diagnosis of models according to the task metadata.

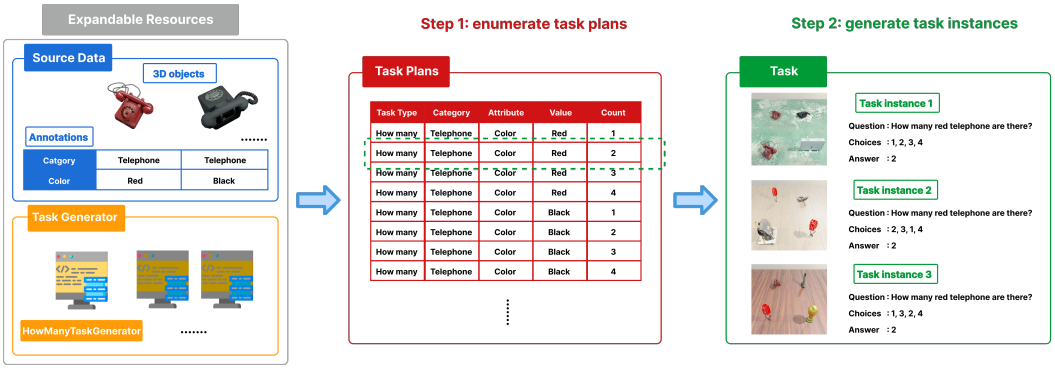


Figure 6: An illustration of core concepts and the task generation process.

C Details of Fine-grained User Query and Query Approximation Algorithms

With TASKVERSE, most user queries regarding model performance can be simply addressed by identifying the relevant task generators and a subset of the task plans to generate task instances for model investigation. However, there is a special family of fine-grained user queries regarding individual tasks and taxonomy concepts that may require a large number of tasks to be appropriately addressed. For example, *the colors that the minimum performance of models M1, M2 is larger than 50%*; such a query involves tasks related to all the color attributes and concerns the models' performance on each individual color. In this section, we outline four types of such fine-grained user queries and discuss how to address them with efficient query results approximation.

C.1 Fine-grained user query

We introduce four types of fine-grained user query. By default, the target of a query is the tasks, *e.g.*, Top K <task>; one can also query different task metadata or their products, *e.g.*, Top K <category> or Top K <category $\times$ attribute>.

Top-K query. Users may be interested in knowing the tasks or task metadata (*e.g.*, object category) that the model(s) performs the best or the worst, which can be supported by a Top-K query. An example Top-K query in natural language is, *(E1) Top 10 "how many" tasks ranked by the maximum performance of the user-specified list of models (the user specifies all models in this case) in descending order*. This query finds the top 10 tasks that all models perform the best, measured by the maximum performance of the models on each task.

Threshold query. Another useful type of query is the Threshold query, since users may want to know the tasks or task metadata on which the model's performance is larger or lower than a given threshold. An example in natural language is, *(E2) The color attributes on which the mean of the minimum performance of models M1, M2 is larger than 50%*. The query first groups tasks by their color value attribute and then aims to find the groups where the mean of the minimum performance of M1 and M2 across all tasks in the group is larger than 50%.

Built upon basic queries, one can develop new types of queries to fulfill specific needs, *e.g.*, comparing models or diagnosing the model. Here, we showcase two advanced queries based on the Threshold query: model compare and debug.

Model Comparison query. A useful type of query is to support comparing a model to another. In contrast to the traditional way of comparing models by ranking based on their performance, our *Model Comparison Query* supports finding tasks or patterns where one model performs better than the other by a given threshold. An example query is *(E3) The task types on which the mean performance of model M1 is larger than model M2*.

Model Debugging query. Model debugging is an important field of study for model evaluation [**?**], where the goal is to find patterns or subgroups where the model performs significantly worse or better than its average performance. To fulfill this need, we support *Model Debugging Queries* by leveraging the Threshold query with the threshold being a function of the model's average performance and a hyperparameter. For example, to find tasks where the model performs significantly worse than average, we can use the Threshold query and set the threshold to be $\mu - \sigma$, where μ is the averaged performance of the model and σ is the standard deviation of the model performance. An example query is *(E4) The tasks on which the performance of model M1 is lower than its average performance of all tasks by a standard deviation*.

Note that these two types of query can be similarly defined based on the Top-K query, *e.g.*, the Model Debugging query can be the top k tasks that a model performs the worst, and how to define these queries depends on the user need.

C.2 Query execution

We provide an example of the conceptual query execution process in Figure 7, which illustrates the steps required to execute query E2. Query E2 requires these steps:

1. Filter: the query filters the task plans related to "color".
2. Generate and evaluate: the query needs to generate the tasks given the obtained task plans and then evaluate model M1 and M2 against these tasks to collect their accuracy for each task.
3. Aggregate: once we obtain models' accuracy on every involved task, we perform some aggregate functions to collect the final results. We first compute the minimum accuracy of models M1 and M2 on each task. Then we average the obtained minimum accuracy over tasks within one color value group, to gather the final results for each color value group.
4. Select: for each group, the query checks whether the final result is greater than 0.5 and only keeps the groups where this filter condition holds.

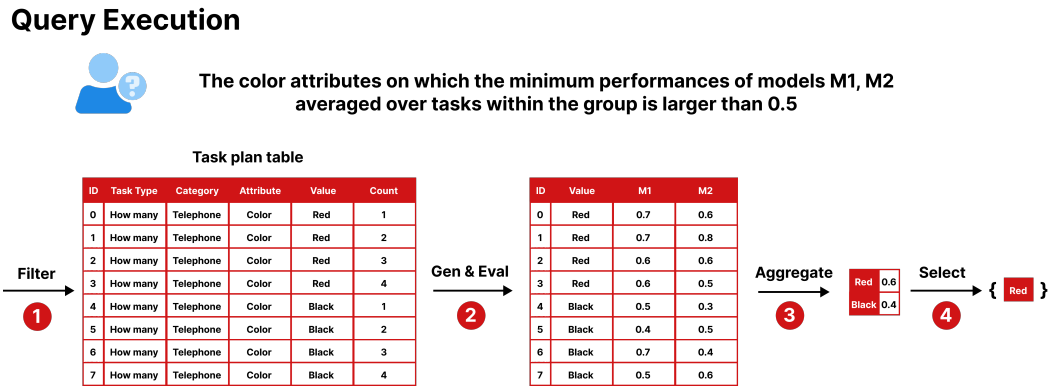


Figure 7: An illustration of the query execution process.

Incorporating frequent pattern mining. In practice, users may be more interested in knowing the patterns revealed by the returned tasks than the tasks themselves. Because each task in our system is associated with a task plan, one can apply frequent pattern mining [13, 38, 12] to extract frequent patterns from the set of task plans associated with the returned tasks. Note that frequent pattern mining can be applied to the results of any type of query as long as there is a set of associated task plans.

C.3 Efficient Query Approximation Algorithms

As the fine-grained user queries may involve a large number of tasks to evaluate and therefore likely become computationally infeasible due to the compute-intensive nature of MLMs, we study three algorithms to approximate the query results given a budget of B on the number of tasks to be evaluated.

Subset proxy. One straightforward approach to approximate the query results is to spend the budget randomly sampling B tasks and then evaluate the models against them to obtain the results. Then, we use this sampled subset as a proxy of the whole set of tasks to perform the fine-grained user query.

Fitting. Built upon the subset proxy method, the fitting method uses the evaluation results of the B randomly sampled tasks to train a model (referred to as *function approximator*) to approximate the function of interest, and then apply the model to the rest of the tasks to predict the results. In particular, the function of interest can be the model's accuracy function which inputs a task and predicts the model's accuracy, or the task aggregate function, *e.g.*, the minimum accuracy of two models as in query E2. Finally, we perform the query over all the tasks, with both actual evaluation results on B sampled tasks and values of the remaining predicted by the function approximator.

Active evaluation. The third approach, active evaluation, builds upon the fitting method but enhances it by strategically selecting tasks to improve the approximation of query results, as opposed to relying on random sampling. This method utilizes an iterative process, where each step involves selecting a batch of unevaluated tasks based on predictions made by the current function approximator.

These tasks are then evaluated, and the results are used to re-fit the function approximator with both existing and new data until the evaluation budget is exhausted. Ultimately, the query is executed using a combination of actual results from evaluated tasks and predicted results, similar to the fitting method. The task selection criteria are tailored to the specific type of query. For the Top-K query, it selects the top-K tasks most likely to fulfill the user's inquiry based on the predicted values, because these tasks are predicted to have the most significant impact on the outcome of the query, and focusing on them could help learn a function approximator with more accurate predictions in areas that are likely relevant to the actual query results. For the Threshold query, it selects the tasks whose predicted values are closest to the threshold, because these tasks are most likely to influence the decision boundary of the function approximator and thus are critical for accurately determining the boundary's position within the task space.

Implementation details. To learn a function approximator to predict the value of interest, we first need a representation of each task as the input of the approximator. We construct such representation using the task plan, question, and answer associated with each task. In particular, we convert these elements into a piece of formulated text and leverage pre-trained embedding models to calculate the text embedding as the task embedding. We adopt Gaussian Process regressor[2] because of its stable performance in our preliminary experiments, while any regression model is applicable.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html

D Details of TASKVERSE 1.0

In this section, we introduce the task generators implemented in the first version of TASKVERSE. Inspired by the model cards for model reporting [32], we make a task generator card for each implemented task generator, including information such as task type, task plan schema, *etc.*, available in the appendix, and the template can be found in Figure 8.

Task Generator Card Template

- **Basic Information**.
  - **Task Type**. The target type of task, *e.g.*, ImageQA
  - **Question Type**. The type of generated question, *e.g.*, "how many"
  - **Answer Type**. The answer type *e.g.*, integer number or object category
  - **The model capability to evaluate**. *e.g.*, counting
- **Source Data**. The source data and annotations it requires
- **Task Plan Schema**. The schema of the associated task plans
- **Partitions**. The partition of the task space.
  - **Partition 1**.
    - **Template**. Template used to generate question if available
    - **Example**. An example of generated test case
- **Limitations**
- **Recommendations**

Figure 8: Summary of task generator card sections and suggested prompts for each. Task generator cards for all the included task generators can be found in Appendix I.

D.1 Source data

3D objects with annotations. We start by selecting objects from Objaverse-LVIS, the subset of Objaverse 1.0 [8] that has been annotated with LVIS [11] categories. From the set of 47K objects spanning 1,230 categories that comprise Objaverse-LVIS, we select 1,996 objects spanning 337 categories. These objects were manually chosen for their high quality and strong category alignment. We use Blender [5], an open-source ray-tracing software, to render each object from a uniform set of surrounding viewpoints and, following manual verification, only keep renderings where the object's category and attributes are discernible. This gives us a set of viewpoint annotations that we also use when constructing 3D scenes, as they allow us to ensure that the object's category and attributes are perceivable from the camera.

Real images and videos with *Scene Graph*. We also collect real images and videos with scene graph [23] as part of our source data. In particular, we collect real images with scene graphs from the GQA dataset [23, 17] and real videos with scene graphs from the AGQA dataset [18, 35].

Additionally, we normalized the object terms across all source data and built a taxonomy containing 927 concepts and 965 edges using Wikidata and human filtering to avoid concept conflicts in options, such as listing both "apple" and "fruit" as choices.

D.2 Task generators for different scenarios

D.2.1 2D sticker image

The first scenario of TASKVERSE is *2D sticker image*, where we compose task instance images by compositing pre-rendered object images into a 2x2 or 3x3 grid. Such a simple type of image already enables the generation of basic types of visual questions regarding recognizing object categories and attributes, spatial relations, and counting. For example, one task could be *how many red telephones*

are there in the image?. We list the task generators implemented for *2D sticker image* and the statistics in Table 1.

Table 1: *2D sticker image*

Task generator	Example question	Example answer	# of tasks
how many	How many blue objects are there in the image?	2	494
	How many tables are there in the image?	4	6,136
	How many pink beverages are there in the image?	2	27,027
what	What is the object in the bottom middle part of the image?	folding chair	33,163
	What is the object to the left of the telephone?	table lamp	61,648,184
where	Where is the apple in the image?	back left	33,163
	Where is the vacuum cleaner with respect to the backpack?	left	61,648,184
what attribute	What is the material of the object in the middle part of the image?	plastic	27,027
	What is the color of the object to the left of the silverware?	gold	50,175,008
where attribute	Where is the white object in the image?	top right	27,027
	Where is the gray object with respect to the lollipop?	top	50,175,008
Total number of tasks: 223,800,421			

D.2.2 3D tabletop scene

Although *2D sticker image* is a useful setting for generating task instances with speed, the artificial way in which the scenes are constructed through image compositing limits their realism. A real-world scene would come from objects existing in a shared 3D space that is rendered through the perspective of a single camera. As such, in *2D sticker image* we are unable to understand the effects of depth, lighting and occlusion on image understanding. To remedy this, we introduce *3D tabletop scene*, a setting analogous to *2D sticker image*, wherein objects are arranged on a plane in a shared 3D scene and rendered from a fixed camera viewpoint. This allows us to port all of the task generators from *2D sticker image* while also allowing us to test 3D-specific capabilities such as relative depth.

ImageQA. Another way to generate similar yet more realistic images is to compose a 3D tabletop scene using the objects, and then render a 2D image [21]. For this *3D tabletop scene*, we can reuse task generators of *2D sticker image* with some minor modifications regarding the spatial relations. For example, the spatial relation of "in the bottom of" would become "in front of". In addition, we identify two families of task generators unique to 3D scenes: tasks regarding the size and distance of objects, which are not suitable for the 2D scenario discussed above. We list the task generators implemented for ImageQA of *3D tabletop scene* and the statistics in Table 2.

VideoQA. In addition to the aforementioned ImageQA tasks, we also build VideoQA tasks for *3D tabletop scene*. We leverage two temporal attributes, rotation and movement, which can only be identified via video, to construct video-specific task generators and evaluate the models' performance in understanding temporal dynamics. To generate these videos, we keep the same layout of the 3D tabletop scene as ImageQA, but change the positions and angles of the objects across different frames of the video to make the objects move and rotate. Our task generators then target the model's ability to understand these temporal changes in object position and orientation. We list the task generators implemented for VideoQA of *3D tabletop scene* and the statistics in Table 3.

D.2.3 Real images/videos with scene graphs

We also leverage existing manually-annotated scene graph data, *i.e.*, GQA and AGQA, to construct task generators. For ImageQA, because there are three types of nodes in the scene graph for images, *i.e.*, object, relation, and attribute, we accordingly implement three task generators to evaluate models' capability in recognizing these basic visual elements. Similarly, the scene graph for videos consists of three types of nodes, *i.e.*, object, relation, and action, we implement three task generators regarding these visual elements. We list the task generators implemented for ImageQA and VideoQA leveraging scene graphs and the statistics in Table 4&5.

Table 2: *3D tabletop scene* with images

Task generator	Example question	Example answer	# of tasks
how many	How many blue objects are there in the image?	6	494
	How many plates are there in the image?	5	6,136
	How many black furnitures are there in the image?	4	27,027
what	What is the object in the front right part of the image?	scale	33,163
	What is the object to the right of the mobile computer?	bucket	61,648,184
where	Where is the vacuum cleaner in the image?	back left	33,163
	Where is the vacuum cleaner with respect to the wine glass?	left	61,648,184
what attribute	What is the color of the object in the back left part of the image?	red	27,027
	What is the material of the object behind the plate?	wood	50,175,008
where attribute	Where is the wood object in the image?	front right	27,027
	Where is the white object with respect to the trophy?	left	50,175,008
what size	What is the smallest object in the image?	spatula	20,408
what attribute size	What is the color of the smallest object in the image?	black	16,632
where size	Where is the largest object in the image?	back left	20,408
	Where is the smallest object in the image with respect to the car?	front	56,906,016
what distance	What is the object that is farthest from the optical instrument?	juice	61,648,184
what attribute distance	What is the color of the object that is closest to the statue?	beige	50,175,008
where distance	Where is the object that is farthest from the bread in the image?	middle	61,648,184
Total number of tasks: 454,235,261			

Table 3: *3D tabletop scene* with videos

Task generator	Example question	Example answer	# of tasks
what rotate video	What is the object that is rotating counterclockwise in the video?	pants	20,408
	What is the rotating object in the video?	jewelry	20,408
what attribute rotate video	What is the color of the object that is rotating clockwise in the video?	beige	16,632
	What is the color of the rotating object in the video?	yellow	16,632
where rotate video	Where is the stepladder with respect to the rotating object in the video?	back	51,631,112
	Where is the object that is rotating counterclockwise with respect to the microscope in the video?	front left	62,221,736
what move video	What is the object that is moving left in the video?	serving tray	40,816
	What is the moving object in the video?	barrel	40,816
what attribute move video	What is the color of the object that is moving left in the video?	black	33,264
	What is the color of the moving object in the video?	white	33,264
where move video	Where is the object that is moving down located in the video?	back right	40,816
	Where is the moving object located in the video?	back right	40,816
Total number of tasks: 114,176,720			

Table 4: Real images with *Scene Graph*

Task generator	Example question	Example answer	# of tasks
what object	What is the flat object that is on the brown and wood table?	paper	25,169
what attribute	What is the material of the smooth object that is to the right of the yellow container?	plastic	20,554
what relation	What is the relation from the standing object, which the colorful and long snowboard is to the right of, to the blue and long object, which is to the left of the patterned skis?	holding	23,241
Total number of tasks: 68,964			

Table 5: Real videos with *Scene Graph*.

Task generator	Example question	Example answer	# of tasks
what object video	What is the spatial relation of the person to the closet while the person closing a closet?	floor	428,342
what relation video	What is the object that the person is behind after the person watching something in a mirror?	behind	211,983
	What is the person doing to the blanket before the person putting a phone somewhere?	touching	216,359
what action video	What action is the person doing while laughing at something?	sitting at a table	335,386
Total number of tasks: 1,192,070			

D.3 TASKVERSE-UI

The ultimate goal of our query-centric model evaluation framework is to allow diverse users, including ML practitioners and non-technical users, to understand foundation models' capabilities and

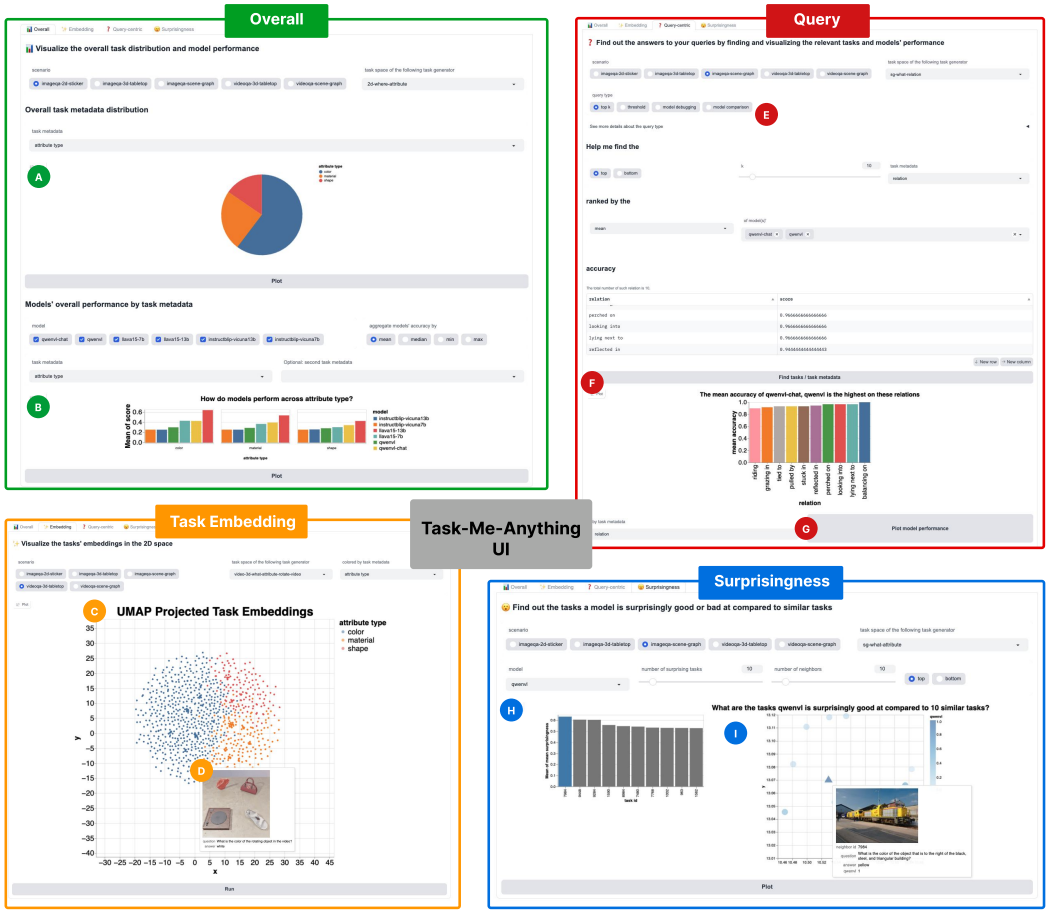


Figure 9: TASKVERSE-UI Interface.

limitations comprehensively and dynamically by answering their various case-specific queries. To achieve this overarching goal, we further break it down into three subgoals and aim to design an interactive end-user interface to achieve these goals:

G1: Support understanding of the overall task space and model performance;

G2: Enable deeper understanding of models through query-centric visualization of model performance (especially for common queries);

G3: Facilitate model debugging via discovery of surprising results.

To achieve these goals, we implemented a graphical user interface with Gradio's [1] framework and used Altair [36, 34] for all the visualizations. In this section, we describe our interface in detail and how its components aim to address our design goals. Then, we present several case studies using this interface in the next section. Our interface consists of four major components organized as different tabs:

Overall. As the name suggests, the Overall tab is designed to help users understand the overall task distribution and model performance (**G1**). It consists of two horizontal sections for visualizing overall task distribution (A) and models' overall performance (B) respectively. Section A displays a pie chart of the distribution of all tasks by metadata based on user's choice of task metadata, while Section B visualizes certain models' aggregated performance in either a bar plot or heat map according to user-selected models, aggregation method and task metadata. We choose these common chart types in hopes of supporting straightfoward understanding of the overall task space and model performance.

Task embedding. In addition to the overall task distribution, we also include the Task Embedding tab to allow users to visualize all tasks at once in a 2D embedding space (**G1**). Concretely, the Embedding tab plots the 2D embeddings of all tasks reduced by UMAP as dots in a scatter plot (C). Further, we add a descriptive tooltip for each dot that displays an example image or video along with the corresponding question-answer pair for this task (D). By visualizing all tasks in one plot and enabling detail of individual tasks on demand at the same time, we hope the interface can help users understand the entire task space well on both high and low levels.

Fine-grained user query. Most importantly, our interface supports query-centric visualizations of model performance under the Query-centric tab. While the space of possible user queries can be infinite, we define four common user queries: top k, threshold, model comparison and model debugging (Section **??**) and support corresponding visualizations (E). As these queries involve selecting a subset of tasks for visualization, we include a "Find tasks/task metadata" button to first select the relevant tasks based on the user query and return these tasks in a table (F). If the user selects task metadata, they will have the option to visualize models' performance on the selected task metadata (G). If the user chooses to find individual tasks however, they can additionally visualize the task distribution by some metadata, or find frequent patterns among tasks. By specifying a query first and visualizing models' performance only on selected tasks/task metadata, users can gain a more targted understanding of models based on what they are interested in (**G1**). In particular, the model debugging query can help the user find buggy model behaviors by identifying tasks/task metadata where the model's performance is lower than its global average accuracy by a large margin i.e. one standard deviation (**G2**).

Surprisingness. Last but not least, we include the Surprisingness tab to help users uncover tasks where models achieve surprisingly good or bad performance compared to their performance on similar tasks (**G3**). We define the "surprisingness" of a model M on a particular task T_i as the following: For a task, T_i and its K nearest neighbors tasks $\{T'_j\}$, we compute the surprisingness score as

$$s_i^M = \frac{1}{K} \sum_{j=1}^{K} \left(\text{sim}(T_i, T'_j) \times (f(T_i, M) - f(T'_j, M)) \right) \tag{1}$$

A higher score indicates the model M is much better at task T_i than the neighbor tasks, while a lower score means M is worse at T_i than the neighbors.

Under the Surprisingness tab, we display the tasks where the model achieves the highest surprisingness scores in a bar chart (H). We also make the bar chart interactive so that the user can select a particular surprising task. Then, the scatter plot on the side visualizes this model's performance on the user-selected task accordingly along with the k most similar tasks in the 2D embedding space (I). With this interactive visualization of surprising tasks, we hope to allow users to uncover unexpected model behaviors quickly.

E Details of Model and Human Performance on Random Task Instances

In this section, we present the full results of our evaluation on TASKVERSE-RANDOM with 18 MLMs and human anntators.

E.1 Raw results of TASKVERSE-RANDOM

Table 6: **TASKVERSE-RANDOM-ImageQA**. The model performance on random subsets of ImageQA tasks using both the detailed prompt and the succinct prompt. Numbers in parentheses are the number of task instances for each set.

	2D sticker image (1,500)		*3D tabletop scene* (3,300)		*Scene Graph* (900)	
	Detailed prompt	**Succinct prompt**	**Detailed prompt**	**Succinct prompt**	**Detailed prompt**	**Succinct prompt**
Human	99.40		99.73		97.33	
INSTRUCTBLIP-7B	28.27	0.60	34.48	0.45	68.33	0.11
INSTRUCTBLIP-13B	28.34	23.87	33.12	24.73	65.22	66.11
QWEN-VL	33.40	13.33	33.48	15.91	68.78	12.56
QWEN-VL-CHAT	40.40	35.87	38.88	39.36	78.33	79.45
LLAVA-7B	37.93	41.87	37.55	39.24	62.00	75.22
LLAVA-13B	45.60	43.20	43.97	42.39	79.22	82.78
GLM-4V	52.74	51.53	53.91	53.70	75.00	70.56
COGVLM2-19B	46.20	48.93	49.94	53.00	71.33	70.66
IDEFICS2-8B	49.20	49.40	50.67	50.00	74.33	74.11
PHI-3-VISION-3B	53.60	55.60	47.00	47.91	76.22	77.22
PALIGEMMA-3B	49.27	52.47	43.79	45.42	80.00	81.22
INTERNVL-CHAT-1.5-24B	58.60	57.40	61.06	59.64	84.67	82.33
LLAVA-NEXT-34B	62.80	62.33	56.33	58.06	85.66	84.89
GEMINI-PRO	30.60	31.47	33.03	31.09	56.78	60.89
QWEN-VL-MAX	55.46	53.33	53.49	55.06	85.67	89.33
GPT4V	34.60	52.40	36.73	47.55	73.44	71.78
GPT4O	45.33	54.80	46.00	58.61	76.33	77.34

Table 7: **TASKVERSE-RANDOM-VideoQA**. The model performance on random subsets of VideoQA tasks using both the detailed prompt and the succinct prompt. Numbers in parentheses are the number of task instances for each set.

	3D tabletop scene (1,800)		*Scene Graph* (900)	
	Detailed prompt	**Succinct prompt**	**Detailed prompt**	**Succinct prompt**
Human	98.33		99.33	
VIDEO-CHATGPT-7B	21.44	21.39	30.45	25.67
VIDEO-LLAVA-7B	26.00	38.78	32.11	56.67
VIDEOCHAT2-7B	30.61	28.55	37.89	32.89
VIDEO-LLAMA-2-7B	23.78	16.33	36.34	31.67
VIDEO-LLAMA-2-13B	22.67	20.23	30.78	28.45
CHAT-UNIVI-7B	29.72	25.95	50.11	45.00
CHAT-UNIVI-13B	28.17	25.67	45.22	39.89
INTERNVL-CHAT-1.5-24B	38.33	31.67	68.11	56.33
LLAVA-NEXT-34B	40.06	41.17	67.55	63.44
GEMINI-PRO	31.78	30.11	50.00	45.78
QWEN-VL-MAX	38.89	39.39	69.11	66.78
GPT4V	30.95	36.83	59.11	62.67
GPT4O	35.67	41.72	69.56	66.22

E.2 A breakdown of Table 6

Table 8: random-*2D sticker image*

	how many		what		what attribute		where		where attribute	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**
Human	100.00		98.00		100.00		100.00		99.00	
InstructBLIP-7B	23.67	0.00	24.33	0.00	39.67	0.00	27.00	1.00	26.67	2.00
InstructBLIP-13B	26.67	30.67	23.67	24.33	41.67	40.67	23.67	22.00	26.00	1.67
Qwen-VL	30.67	9.00	36.67	9.00	47.00	17.67	27.33	15.00	25.33	16.00
Qwen-VL-Chat	39.67	24.67	42.67	42.67	54.67	52.00	31.67	33.00	33.33	27.00
LLaVA-7B	42.00	40.67	40.00	45.67	48.67	49.67	31.00	39.00	28.00	34.33
LLaVA-13B	49.33	48.33	46.00	46.67	58.33	55.33	39.67	32.67	34.67	33.00
GLM-4v	56.67	58.67	57.67	56.00	62.00	59.33	46.67	42.33	40.67	41.33
CogVLM2-19B	45.67	53.33	50.00	50.00	66.67	64.67	36.33	38.00	32.33	38.67
Idefics2-8B	61.33	61.33	49.00	46.00	56.33	56.33	38.33	44.33	41.00	39.00
Phi-3-vision-3B	60.00	63.33	50.00	51.33	66.67	69.67	47.00	49.33	44.33	44.33
PaliGemma-3B	48.67	47.33	48.67	53.00	59.00	64.00	49.33	54.00	40.67	44.00
InternVL-Chat-1.5-24B	57.67	60.67	62.00	55.00	**75.33**	**72.33**	**51.33**	49.33	46.67	**49.67**
LLaVA-Next-34B	**68.33**	**64.67**	**63.33**	**62.67**	**72.00**	70.67	**57.33**	**58.33**	**53.00**	**55.33**
Gemini-Pro	33.33	34.33	32.67	38.00	32.33	33.00	26.67	28.33	28.00	23.67
Qwen-VL-Max	58.33	45.00	57.00	59.67	71.33	68.33	48.33	47.33	42.33	46.33
GPT4V	40.00	**68.67**	40.67	50.33	41.00	60.33	25.67	42.67	25.67	40.00
GPT4o	44.67	53.67	50.33	**62.33**	60.00	67.00	36.00	45.67	35.67	45.33

Table 9: random-*3D tabletop scene* part 1

	how many		what		what attribute		where		where attribute	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**
Human	99.00		100.00		100.00		99.00		100.00	
InstructBLIP-7B	32.67	0.00	28.00	0.00	45.00	0.00	25.67	1.00	27.00	2.33
InstructBLIP-13B	32.00	32.33	22.67	23.33	42.67	0.00	28.67	25.33	23.00	24.67
Qwen-VL	32.33	11.00	28.00	8.67	50.67	19.67	22.67	18.33	24.67	15.00
Qwen-VL-Chat	45.00	33.33	32.33	33.33	55.00	57.00	21.67	24.00	29.67	32.33
LLaVA-7B	38.67	39.33	32.67	40.33	57.00	54.00	27.00	27.67	26.00	26.00
LLaVA-13B	46.67	48.33	40.67	41.00	60.33	56.00	34.33	32.67	36.00	32.67
GLM-4v	**74.00**	**73.00**	55.33	47.00	67.67	65.67	40.67	38.33	33.33	36.00
CogVLM2-19B	60.67	62.67	39.67	42.67	57.67	58.33	31.67	34.00	29.67	32.00
Idefics2-8B	65.00	65.67	41.00	39.33	64.33	58.33	36.67	38.33	39.00	37.67
Phi-3-vision-3B	59.33	60.33	37.67	40.33	62.00	62.00	34.33	36.67	36.67	38.00
PaliGemma-3B	49.00	45.33	40.67	43.67	63.67	68.00	34.67	39.67	35.67	35.67
InternVL-Chat-1.5-24B	**67.00**	**67.00**	**60.33**	**56.33**	68.33	65.67	**54.67**	**55.67**	**46.67**	**46.00**
LLaVA-Next-34B	63.67	63.33	49.67	50.67	**71.33**	**71.33**	48.33	**51.00**	40.33	**49.00**
Gemini-Pro	40.00	38.67	32.67	25.00	31.33	34.67	28.00	31.00	27.67	28.00
Qwen-VL-Max	65.00	60.67	54.67	55.33	63.67	61.33	42.33	44.00	32.67	37.33
GPT4V	41.67	66.67	31.67	37.67	41.33	54.67	25.00	39.00	25.67	28.33
GPT4o	45.00	64.33	47.33	**58.67**	57.33	**68.67**	37.67	45.33	30.67	44.33

Table 10: random-*3D tabletop scene* part 2

	what distance		where distance		what attribute distance		what size		where size		what attribute size	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**
Human	100.00		99.00		100.00		100.00		100.00		100.00	
InstructBLIP-7B	17.67	0.00	38.33	0.00	51.00	0.00	30.33	0.00	32.33	1.67	51.33	0.00
InstructBLIP-13B	23.67	24.33	29.33	29.00	48.00	1.67	35.67	37.00	25.33	24.00	53.33	50.33
Qwen-VL	25.33	8.67	26.33	14.00	50.33	19.67	34.67	14.00	21.33	19.00	52.00	27.00
Qwen-VL-Chat	25.00	24.00	25.67	28.33	56.67	56.00	43.00	48.67	31.00	30.67	62.67	65.33
LLaVA-7B	28.00	30.67	26.33	25.67	49.67	48.67	43.00	44.67	29.33	34.67	55.33	60.00
LLaVA-13B	33.67	29.33	26.00	23.67	57.67	55.33	48.33	48.33	34.67	35.67	65.33	63.33
GLM-4v	45.67	46.00	18.33	25.00	53.33	57.67	72.67	**73.67**	49.67	45.33	**82.33**	**83.00**
CogVLM2-19B	**53.00**	**55.67**	26.33	35.67	63.00	63.67	70.67	**76.33**	37.67	40.33	79.33	81.67
Idefics2-8B	35.67	36.00	26.00	30.67	65.00	61.00	62.00	63.33	45.33	44.67	77.33	75.00
Phi-3-vision-3B	36.33	33.67	23.00	25.00	61.00	62.00	52.00	53.33	44.33	43.67	70.33	72.00
PaliGemma-3B	20.67	26.33	29.00	23.67	50.67	54.00	48.33	49.33	35.00	41.67	74.33	72.33
InternVL-Chat-1.5-24B	52.33	36.00	39.00	**47.00**	**69.67**	68.67	73.33	**73.67**	**57.67**	**57.67**	**82.67**	**82.33**
LLaVA-Next-34B	48.00	45.33	34.33	**40.67**	**75.00**	**74.00**	62.33	62.00	49.00	**52.67**	77.67	78.67
Gemini-Pro	39.33	31.00	25.33	24.33	38.33	36.00	34.33	29.67	26.67	26.67	39.67	37.00
Qwen-VL-Max	39.00	**53.00**	2.67	35.67	65.00	66.67	72.33	69.67	45.33	50.00	75.67	72.00
GPT4V	39.33	46.67	21.67	19.00	43.33	64.33	46.00	54.00	22.33	37.67	66.00	75.00
GPT4o	44.67	**62.33**	24.00	**41.67**	58.33	65.33	57.67	73.00	32.33	44.67	71.00	76.33

Table 11: random-Real images with *Scene Graph*

	what attribute		what object		what relation	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**
Human	96.00		99.00		97.00	
InstructBLIP-7B	65.67	0.00	79.00	0.00	60.33	0.33
InstructBLIP-13B	66.33	68.67	84.33	80.00	45.00	49.67
Qwen-VL	64.00	4.33	83.33	8.67	59.00	24.67
Qwen-VL-Chat	69.67	69.00	87.00	86.67	78.33	82.67
LLaVA-7B	70.00	65.33	85.00	84.33	31.00	76.00
LLaVA-13B	72.67	70.33	90.00	90.00	75.00	**88.00**
GLM-4v	74.33	72.00	88.67	88.00	62.00	51.67
CogVLM2-19B	70.00	71.33	92.67	93.33	51.33	47.33
Idefics2-8B	69.67	68.67	86.33	85.67	67.00	68.00
Phi-3-vision-3B	77.67	76.00	92.00	93.67	59.00	62.00
PaliGemma-3B	75.33	76.00	94.00	93.33	70.67	74.33
InternVL-Chat-1.5-24B	**80.00**	77.33	**94.67**	92.00	79.33	77.67
LLaVA-Next-34B	**78.33**	75.33	93.33	**95.33**	85.33	84.00
Gemini-Pro	51.00	50.67	71.00	68.67	48.33	63.33
Qwen-VL-Max	76.67	**81.33**	93.67	**96.00**	**86.67**	**90.67**
GPT4V	69.33	67.00	82.67	79.33	68.33	69.00
GPT4o	68.00	67.67	83.00	81.67	78.00	82.67

E.3 A breakdown of Table 7

Table 12: random-*3D tabletop scene*

	what attribute move		what attribute rotate		what move		what rotate		where move		where rotate	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**
Human	100.00		100.00		98.00		92.00		100.00		100.00	
VIDEO-CHATGPT-7B	27.00	24.33	27.00	28.33	18.33	19.00	15.67	18.67	27.33	26.33	13.33	11.67
VIDEO-LLAVA-7B	28.33	54.00	25.00	49.33	26.00	**34.00**	26.67	**35.33**	25.00	31.33	25.00	28.67
VIDEOCHAT2-7B	46.67	48.33	41.33	47.67	29.00	22.33	27.67	19.67	17.00	14.00	22.00	19.33
VIDEO-LLAMA-2-7B	28.67	24.00	27.67	25.00	22.33	19.00	23.33	16.00	20.00	7.33	20.67	6.67
VIDEO-LLAMA-2-13B	29.67	26.67	32.33	32.00	18.33	17.67	19.33	17.67	17.67	14.67	18.67	12.67
CHAT-UNIVI-7B	36.67	27.67	35.33	39.67	27.67	20.33	28.33	24.00	25.67	24.00	24.67	20.00
CHAT-UNIVI-13B	33.67	31.33	33.67	37.00	24.33	22.67	29.33	28.00	25.33	16.33	22.67	18.67
INTERNVL-CHAT-1.5-24B	52.33	43.00	56.00	49.33	26.67	21.00	31.33	22.67	31.67	28.00	32.00	26.00
LLAVA-NEXT-34B	**57.67**	**56.67**	59.00	**62.67**	28.00	29.33	30.67	29.67	**32.33**	**32.33**	**32.67**	**36.33**
GEMINI-PRO	39.33	38.67	40.33	37.67	**30.67**	28.67	27.33	25.33	27.67	29.67	25.33	20.67
QWEN-VL-MAX	**56.33**	52.67	**67.33**	**67.00**	29.00	30.00	34.00	**35.33**	26.00	25.00	20.67	26.33
GPT4V	43.67	51.00	46.67	57.33	28.00	29.33	29.67	32.00	22.00	26.00	15.67	25.33
GPT4O	47.67	46.00	54.67	**62.67**	27.33	**31.00**	34.33	**38.67**	27.00	**36.33**	23.00	**35.67**

Table 13: random-Real videos with *Scene Graph*

	what action		what object		what relation	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**
Human	100.00		98.00		100.00	
VIDEO-CHATGPT-7B	19.67	16.33	37.00	29.67	34.67	31.00
VIDEO-LLAVA-7B	29.67	58.33	31.33	62.67	35.33	49.00
VIDEOCHAT2-7B	36.33	26.33	44.33	42.67	33.00	29.67
VIDEO-LLAMA-2-7B	33.67	21.33	37.67	40.00	37.67	33.67
VIDEO-LLAMA-2-13B	30.33	23.67	39.00	36.00	23.00	25.67
CHAT-UNIVI-7B	44.67	37.67	57.33	47.67	48.33	49.67
CHAT-UNIVI-13B	38.33	25.00	58.67	52.00	38.67	42.67
INTERNVL-CHAT-1.5-24B	**72.33**	52.33	**73.00**	54.33	59.00	62.33
LLAVA-NEXT-34B	67.00	60.00	67.33	65.33	68.33	65.00
GEMINI-PRO	54.33	39.67	55.00	53.00	40.67	44.67
QWEN-VL-MAX	**67.33**	**68.67**	**69.67**	**68.00**	70.33	63.67
GPT4V	53.67	56.67	57.67	58.67	66.00	**72.67**
GPT4O	64.67	62.33	66.00	60.00	**78.00**	**76.33**

F Details of Model Performance on Taskverse 2024 benchmark

In this section, we present the full results of our evaluation on TASKVERSE-2024 with 18 MLMs.

F.1 Raw results of TASKVERSE-2024

Table 14: **2024-ImageQA**. The model performance on 2024 subsets of ImageQA tasks using both the detailed prompt and the succinct prompt. Numbers in parentheses are the number of task instances for each set.

	2D sticker image (3,279)		*3D tabletop scene* (7,095)		*Scene Graph* (1,896)	
	Detailed prompt	**Succinct prompt**	**Detailed prompt**	**Succinct prompt**	**Detailed prompt**	**Succinct prompt**
INSTRUCTBLIP-7B	22.92	1.12	24.94	0.42	43.23	0.38
INSTRUCTBLIP-13B	22.20	0.72	21.55	0.45	45.35	0.86
QWEN-VL	23.48	12.02	23.77	12.36	42.68	11.26
QWEN-VL-CHAT	25.53	24.55	24.21	25.72	53.04	52.38
LLAVA-7B	25.83	26.09	23.64	23.01	42.67	46.75
LLAVA-13B	27.03	24.85	26.25	25.78	51.01	49.10
GLM-4V	32.76	33.80	37.09	37.33	47.80	44.22
COGVLM2-19B	29.99	31.11	37.29	39.66	49.46	48.25
IDEFICS2-8B	30.50	32.20	35.32	34.80	44.40	46.61
PHI-3-VISION-3B	30.44	32.95	27.88	29.52	46.57	49.46
PALIGEMMA-3B	32.14	33.77	28.95	29.42	58.45	58.71
INTERNVL-CHAT-1.5-24B	36.78	37.08	43.53	42.55	57.08	52.01
GPT4O	31.46	40.87	31.86	47.46	52.03	53.87

Table 15: **2024-VideoQA**. The model performance on 2024 subsets of VideoQA tasks using both the detailed prompt and the succinct prompt. Numbers in parentheses are the number of task instances for each set.

	3D tabletop scene (2,394)		*Scene Graph* (1,173)	
	Detailed prompt	**Succinct prompt**	**Detailed prompt**	**Succinct prompt**
VIDEO-CHATGPT-7B	12.86	10.87	15.56	13.87
VIDEO-LLAVA-7B	19.59	22.14	21.68	34.54
VIDEOCHAT2-7B	21.69	17.42	30.31	19.01
VIDEO-LLAMA-2-7B	18.79	10.59	27.32	20.36
VIDEO-LLAMA-2-13B	17.31	12.34	23.17	15.10
CHAT-UNIVI-7B	16.48	14.47	36.44	26.66
CHAT-UNIVI-13B	17.84	15.36	27.30	20.60
INTERNVL-CHAT-1.5-24B	23.67	23.58	54.04	38.02
GPT4O	26.96	34.53	57.88	58.23

F.2 A breakdown of Table 14

Table 16: 2024-*2D sticker image*

	how many		what		what attribute		where		where attribute	
	DP	SP	DP	SP	DP	SP	DP	SP	DP	SP
InstructBLIP-7B	25.25	0.00	16.74	0.00	28.77	0.00	22.75	2.15	21.07	3.46
InstructBLIP-13B	22.11	0.00	14.83	0.00	33.33	0.00	20.60	1.57	20.13	2.04
Qwen-VL	23.27	4.13	21.15	7.64	35.16	11.42	19.74	18.03	18.08	18.87
Qwen-VL-Chat	28.88	21.12	23.49	24.52	36.53	35.31	21.32	24.03	17.45	17.77
LLaVA-7B	29.04	31.19	19.53	23.20	31.20	29.07	26.90	25.75	22.48	21.23
LLaVA-13B	30.86	34.32	21.73	21.44	40.03	34.40	24.46	19.46	18.08	14.62
GLM-4v	43.40	45.05	29.37	31.28	42.31	44.44	27.18	27.18	21.54	21.07
CogVLM2-19B	41.09	43.23	26.87	27.61	42.16	44.75	23.03	23.61	16.82	16.35
Idefics2-8B	42.90	41.91	24.96	26.28	41.70	39.88	25.04	29.33	17.92	23.58
Phi-3-vision-3B	38.94	40.26	27.61	26.28	40.33	44.14	23.32	27.18	22.01	26.89
PaliGemma-3B	33.66	33.83	22.61	24.96	38.81	45.36	37.34	37.48	28.30	27.20
InternVL-Chat-1.5-24B	38.78	47.85	34.95	30.40	48.86	40.33	33.48	34.91	27.83	31.92
GPT4o	36.80	48.35	31.57	41.56	37.29	48.25	28.04	36.77	23.58	29.40

Table 17: 2024-*3D tabletop scene* part 1

	how many		what		what attribute		where		where attribute	
	DP	SP	DP	SP	DP	SP	DP	SP	DP	SP
InstructBLIP-7B	26.30	0.00	17.36	0.00	28.79	0.00	26.31	1.05	25.45	1.81
InstructBLIP-13B	20.49	0.00	15.35	0.16	29.89	0.00	21.23	0.75	21.35	1.15
Qwen-VL	30.28	6.88	20.00	5.74	33.49	14.55	20.33	14.05	20.85	11.82
Qwen-VL-Chat	34.86	32.72	19.84	20.47	36.46	35.84	18.24	18.68	16.09	20.53
LLaVA-7B	21.87	21.87	21.24	24.65	29.58	28.17	24.96	20.48	24.96	19.87
LLaVA-13B	29.66	34.25	23.88	24.50	34.90	32.55	22.87	19.43	21.02	19.54
GLM-4v	48.93	48.78	39.69	38.45	42.88	43.51	24.51	22.12	19.54	19.38
CogVLM2-19B	37.77	44.50	29.77	30.23	44.29	46.17	21.23	24.07	21.02	20.36
Idefics2-8B	42.05	41.13	25.89	24.50	40.85	38.97	29.75	28.85	27.09	29.06
Phi-3-vision-3B	40.98	42.05	25.89	24.65	37.25	38.50	20.03	20.63	18.88	23.65
PaliGemma-3B	37.46	30.58	22.95	26.36	38.03	42.25	33.78	32.14	26.27	26.44
InternVL-Chat-1.5-24B	43.12	49.08	39.84	38.14	50.23	45.54	34.08	33.03	33.17	26.93
GPT4o	32.26	54.43	27.29	40.93	36.62	52.27	22.12	32.59	19.38	26.93

Table 18: 2024-*3D tabletop scene* part 2

	what distance		where distance		what attribute distance		what size		where size		what attribute size	
	DP	SP	DP	SP	DP	SP	DP	SP	DP	SP	DP	SP
InstructBLIP-7B	17.26	0.00	31.94	0.00	31.71	0.00	13.04	0.00	27.99	1.78	28.17	0.00
InstructBLIP-13B	16.92	0.17	24.50	0.62	29.20	0.00	9.33	0.15	20.06	1.94	28.76	0.00
Qwen-VL	17.78	8.03	23.88	16.59	28.17	11.50	16.59	11.41	21.68	17.80	28.47	17.55
Qwen-VL-Chat	19.66	18.63	17.36	21.09	31.86	31.12	20.30	22.22	19.90	26.21	31.71	35.40
LLaVA-7B	22.91	23.42	22.17	22.17	26.11	25.22	17.04	19.85	22.98	24.60	26.25	22.86
LLaVA-13B	23.93	25.64	19.38	17.05	32.60	30.24	23.85	24.59	21.84	25.24	34.81	30.53
GLM-4v	35.90	36.75	25.89	25.27	34.96	40.27	55.26	55.26	32.36	31.39	48.08	49.41
CogVLM2-19B	42.39	44.79	26.51	32.25	50.29	52.51	54.52	54.07	29.77	32.20	52.65	55.16
Idefics2-8B	31.79	30.26	22.79	24.50	42.18	43.81	34.96	33.48	37.22	36.57	53.98	51.62
Phi-3-vision-3B	30.43	28.72	15.66	19.38	31.27	34.22	28.59	28.74	27.35	30.10	30.38	34.07
PaliGemma-3B	17.44	21.37	25.27	18.29	30.38	31.56	16.00	19.70	31.88	33.33	38.94	41.59
InternVL-Chat-1.5-24B	44.96	26.84	33.95	41.86	46.76	48.53	50.37	51.85	43.85	43.53	58.55	62.68
GPT4o	38.97	57.61	19.84	38.91	42.04	54.72	32.59	54.07	28.80	43.04	50.59	66.52

Table 19: 2024-Real images with *Scene Graph*

	what attribute		what object		what relation	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**
InstructBLIP-7B	42.48	0.00	39.97	0.32	47.25	0.81
InstructBLIP-13B	46.98	0.47	50.55	0.16	38.51	1.94
Qwen-VL	43.10	1.24	46.76	6.48	38.19	26.05
Qwen-VL-Chat	50.54	45.27	47.08	50.87	61.49	61.00
LLaVA-7B	47.44	36.59	54.03	48.97	26.54	54.69
LLaVA-13B	41.71	34.26	54.19	49.61	57.12	63.43
GLM-4v	42.17	40.78	56.56	54.03	44.66	37.86
CogVLM2-19B	45.12	46.51	65.72	65.40	37.54	32.85
Idefics2-8B	38.29	36.12	48.97	53.55	45.95	50.16
Phi-3-vision-3B	45.58	44.03	54.98	60.98	39.16	43.37
PaliGemma-3B	48.84	49.15	74.57	74.88	51.94	52.10
InternVL-Chat-1.5-24B	47.75	42.64	65.24	55.29	58.25	58.09
GPT4o	39.69	41.86	51.50	52.76	64.89	66.99

F.3 A breakdown of Table 15

Table 20: 2024-*3D tabletop scene*

	what attribute move		what attribute rotate		what move		what rotate		where move		where rotate	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**	**DP**	**SP**
VIDEO-CHATGPT-7B	21.54	13.33	10.12	7.65	5.88	7.11	12.41	12.17	21.39	19.40	5.82	5.56
VIDEO-LLAVA-7B	24.36	32.05	20.49	22.22	17.65	18.38	18.25	19.22	16.92	17.66	19.84	23.28
VIDEO-LLAMA-2-7B	22.31	15.38	24.20	12.84	13.73	11.52	13.14	9.25	17.41	8.46	21.96	6.08
VIDEO-LLAMA-2-13B	26.41	15.90	21.98	15.80	13.97	9.56	13.14	12.17	11.69	8.46	16.67	12.17
CHAT-UNIVI-7B	16.67	14.62	16.54	14.07	13.73	15.93	17.03	15.33	16.42	14.68	18.52	12.17
CHAT-UNIVI-13B	24.62	23.08	19.51	15.56	12.75	9.80	14.84	17.52	18.41	7.71	16.93	18.52
INTERNVL-CHAT-1.5-24B	35.38	35.90	36.30	36.79	16.67	12.99	18.00	15.82	13.43	16.17	22.22	23.81
GPT4O	31.54	37.69	39.01	53.33	22.06	17.89	26.52	33.82	15.92	26.62	26.72	37.83

Table 21: 2024-Real videos with *Scene Graph*

	what action		what object		what relation	
	DP	**SP**	**DP**	**SP**	**DP**	**SP**
VIDEO-CHATGPT-7B	18.40	19.47	13.44	12.66	14.84	9.49
VIDEO-LLAVA-7B	23.47	37.33	16.28	27.13	25.30	39.17
VIDEO-LLAMA-2-7B	21.87	14.13	28.94	24.81	31.14	22.14
VIDEO-LLAMA-2-13B	25.60	16.27	26.87	17.83	17.03	11.19
CHAT-UNIVI-7B	40.00	28.53	33.07	18.60	36.25	32.85
CHAT-UNIVI-13B	30.67	20.00	27.13	19.90	24.09	21.90
INTERNVL-CHAT-1.5-24B	63.73	41.07	47.29	28.94	51.09	44.04
GPT4O	57.33	59.73	41.86	39.79	74.45	75.18

G Details of Experiments on Query Results Approximation Algorithms

To experiment with different query results approximation approaches, we first conduct extensive experiments to evaluate a set of representative models against a subset of tasks for each task generator. Then, we build an Oracle database with the obtained evaluation results, referred to as TASKVERSE-DB, and study different query results approximation methods with this Oracle database to verify their effectiveness. We will release the TASKVERSE-DB for future studies of query results approximation or model performance prediction.

G.1 Experiment details

Setup. For image question answering tasks, We select 6 representative open-sourced large multi-modal language models (MLMs) from 3 model families: INSTRUCTBLIP-7B and INSTRUCTBLIP-13B from INSTRUCTBLIP [6], QWEN-VL and QWEN-VL-CHAT from QWEN-VL [4], and LLAVA-7B and LLAVA-13B from LLAVA [28]. For video question answering tasks, We select 7 representative open-sourced Large Video Language Models from 5 model families: VIDEO-LLAMA-2-7B and VIDEO-LLAMA-2-13B from VIDEO-LLAMA-2 [40], VIDEO-CHATGPT-7B from VIDEO-CHATGPT [30], CHAT-UNIVI-7B and CHAT-UNIVI-13B from CHAT-UNIVI [20], VIDEO-LLAVA-7B from VIDEO-LLAVA [27], and VIDEOCHAT2-7B from VIDEOCHAT2 [26]. We evaluate the models against a subset of tasks whose statistics can be found in Table 22. Since we generate 15 task instances for each task and involve multiple models, these lead to a total number of 24,240,780 <model, task instance> pairs in evaluation. We evaluate the query results approximation methods on a series of query instances for each type of query. These query instances cover all the subsets of tasks and models we evaluate, leading to a set of 1137 query instances in total (741 for ImageQA and 396 for VideoQA). We set the budget to 2,000 task evaluations.

Table 22: **Statistics of evaluated tasks.** For each task, we generate 15 task instances for evaluation.

	Scenerio	Task generator	# of tasks
ImageQA	***2D sticker image***	how many	17,238
		what	12,740
		where	12,740
		what attribute	12,740
		where attribute	12,740
	3D tabletop scene	how many	17,238
		what	12,740
		where	12,740
		what attribute	12,740
		where attribute	12,740
		what size	10,304
		what attribute size	7,840
		where size	10,304
		what distance	6,160
		what attribute distance	6,000
		where distance	6,160
	real image w *Scene Graph*	what object	10,000
		what attribute	10,000
		what relation	10,000
Total number of tasks: 144,966			
VideoQA	***3D tabletop scene***	what rotate video	2,464
		what attribute rotate video	7,840
		where rotate video	2,464
		what distance video	4,928
		what attribute distance video	15,680
		where distance video	4,928
	Real video w *Scene Graph*	what object video	10,000
		what action video	10,000
		what relation video	10,000
Total number of tasks: 106,608			

Evaluation metrics. To evaluate the query results approximation methods, we adopt different evaluation metrics for different types of queries. For Top-K queries, we report the Mean Rank and the Hit Rate: Mean Rank is the average of the ground truth rank of the K items returned by the query results approximation method, so a lower Mean Rank indicates the returned items are actually ranked higher and the query results approximation method is better; Hit Rate measures the percentage of the K returned items are actual Top-K items, so the higher is the better. For the Threshold query and its variants (Model Comparison and Model Debugging query), we can treat them as a binary classification problem and adopt the Prediction, Recall, and F1-score as evaluation metrics.

G.2 Experiments on approximations under different budgets.

To evaluate the performance of approximation algorithms under different budgets, we conducted an experiment using QWEN-VL-CHAT as the target model on 2D how-many tasks. We tested three query approximation algorithms on four types of queries: Top-K query, Threshold query, Model comparison query, and Model debugging query. The experiments were performed under budgets of $1,000$, $2,000$, and $3,000$. The results of the experiment can be found in Table 23, 24, 25, and 26.

The results demonstrate that the *Active* approximation algorithm consistently outperforms the *Random* and *Fitting* algorithms across all query types and budget levels. In particular, for the Model Compare query, *Active* achieves better results with a 2,000 budget than baselines with larger budgets. Also, we can see the performance increase rapidly with more budget, indicating that users could have more accurate results when using a larger budget

Table 23: The performance of Top-K query results approximation algorithms with different budgets.

Budget	Random		Fitting		Active	
	MR	HR (%)	MR	HR (%)	MR	HR (%)
1,000	137.1	0.0	143.3	10.0	44.3	20.0
2,000	116.6	0.0	121.8	0.0	32.2	20.0
3,000	110.3	10.0	121.4	10.0	21.4	20.0

Table 24: The performance of Threshold query results approximation algorithms with different budgets.

Budget	Random			Fitting			Active		
	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
1,000	42.61	31.82	36.43	48.48	10.39	17.11	45.0	11.69	18.56
2,000	43.90	35.06	38.99	43.44	34.42	38.41	43.44	34.42	38.41
3,000	45.38	38.31	41.55	45.89	43.51	44.67	50.93	71.43	59.46

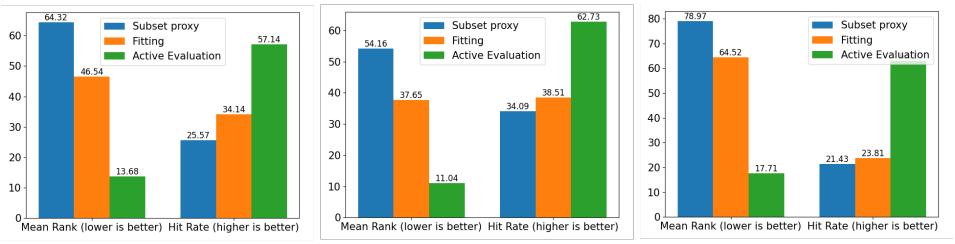
Table 25: The performance of Model comparison query results approximation algorithms with different budgets.

Budget	Random			Fitting			Active		
	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
1,000	100.0	5.86	11.08	88.34	6.73	12.51	61.22	28.71	39.09
2,000	100.0	11.37	20.42	62.88	31.82	42.26	75.18	41.44	53.43
3,000	100.0	17.41	29.66	69.74	43.19	53.35	82.81	52.30	64.11

Table 26: The performance of Model debugging query results approximation algorithms with different budgets.

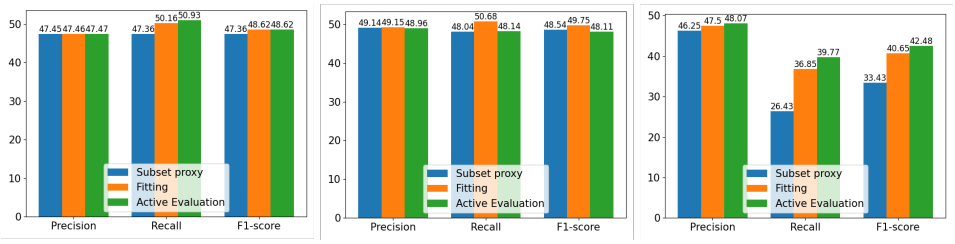
Budget	Random			Fitting			Active		
	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
1,000	100.0	6.34	11.92	100.0	6.34	11.92	100.0	6.93	12.96
2,000	100.0	13.50	23.79	97.18	13.58	23.83	100.0	15.0	26.09
3,000	100.0	18.82	31.68	95.29	19.13	31.87	100.0	22.01	36.08

G.3 Query results approximation experiments in ImageQA



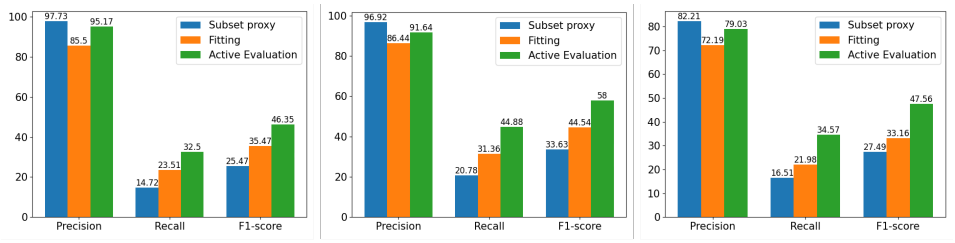
(a) 2D sticker image (b) 3D tabletop scene (c) Scene Graph w real images

Figure 10: **Top-K Query.** These three bar graphs display the performance of three query approximation methods in Top-K Query, measured by Mean Rank and Hit Rate.



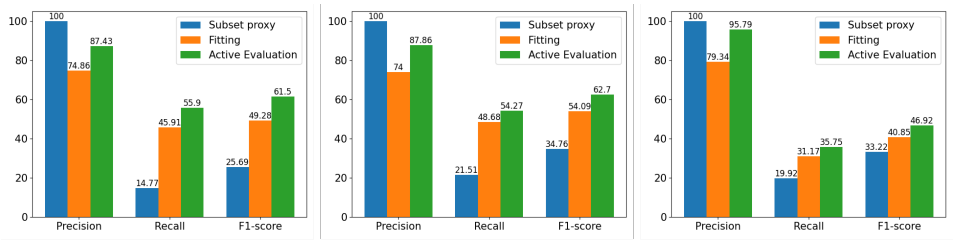
(a) 2D sticker image (b) 3D tabletop scene (c) Scene Graph w real images

Figure 11: **Threshold Query.** These three bar graphs display the performance of three query approximation methods in Threshold Query, measured by Precision, Recall, and F1-score.



(a) 2D sticker image (b) 3D tabletop scene (c) Scene Graph w real images

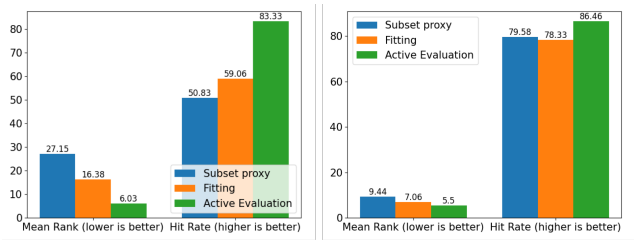
Figure 12: **Model Debugging Query.** These three bar graphs display the performance of three query approximation methods in Model Debugging Query, measured by Precision, Recall, and F1-score.



(a) 2D sticker image (b) 3D tabletop scene (c) Scene Graph w real images

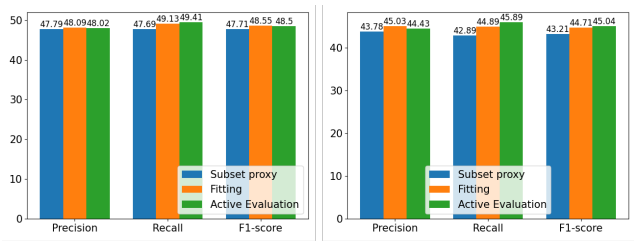
Figure 13: **Model Comparison Query.** These three bar graphs display the performance of three query approximation methods in Model Comparison Query, measured by Precision, Recall, and F1-score.

G.4 Query results approximation experiments in VideoQA



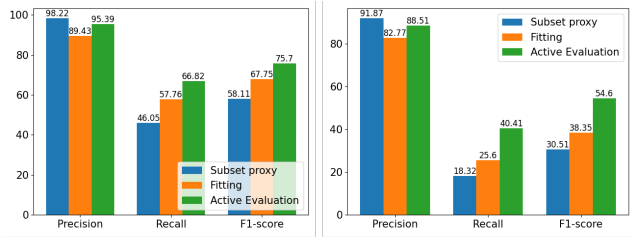
(a) 3D tabletop video (b) Scene Graph w real video

Figure 14: **Top-K Query in VideoQA.** These three bar graphs display the performance of three query approximation methods in Top-K Query, measured by Mean Rank and Hit Rate.



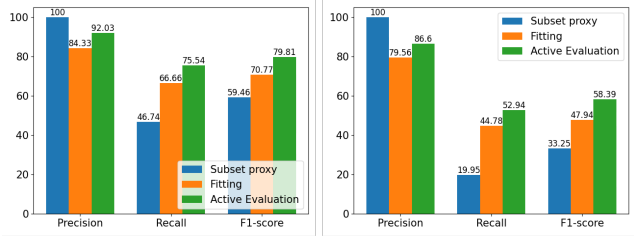
(a) 3D tabletop video (b) Scene Graph w real video

Figure 15: **Threshold Query in VideoQA.** These three bar graphs display the performance of three query approximation methods in Threshold Query, measured by Precision, Recall, and F1-score.



(a) 3D tabletop video (b) Scene Graph w real video

Figure 16: **Model Debugging Query in VideoQA.** These three bar graphs display the performance of three query approximation methods in Model Debugging Query, measured by Precision, Recall, and F1-score.



(a) 3D tabletop video (b) Scene Graph w real video

Figure 17: **Model Comparison Query in VideoQA.** These three bar graphs display the performance of three query approximation methods in Model Comparison Query, measured by Precision, Recall, and F1-score.

H Details of Analysis and Case Study

H.1 What task metadata are models good or bad at?

To obtain a more finegrained understanding of models' skill sets, we also leverage our interface to examine the top and bottom task metadata related to models' best and worst skills. For example, as QWEN-VL-CHAT performs the best on relation understanding across models and skills, we identify the top 20 relations where QWEN-VL-CHAT achieves the highest accuracies (Figure 18) and find that they are mostly actions. Similarly, on VideoQA tasks related to attribute understanding, we are also able to find the attribute values VIDEOCHAT2-7B is the best at and learn that they are mostly associated with color instead of shape or material (Figure 19). On the other hand, we learn that INSTRUCTBLIP-13B does terribly on spatial understanding especially when the object's absolute position is in the back, followed by front right or left (Figure 20); and among the actions VIDEO-LLAMA-2-13B performs the worst on, most involve "putting" or "throwing" something (Figure 21).

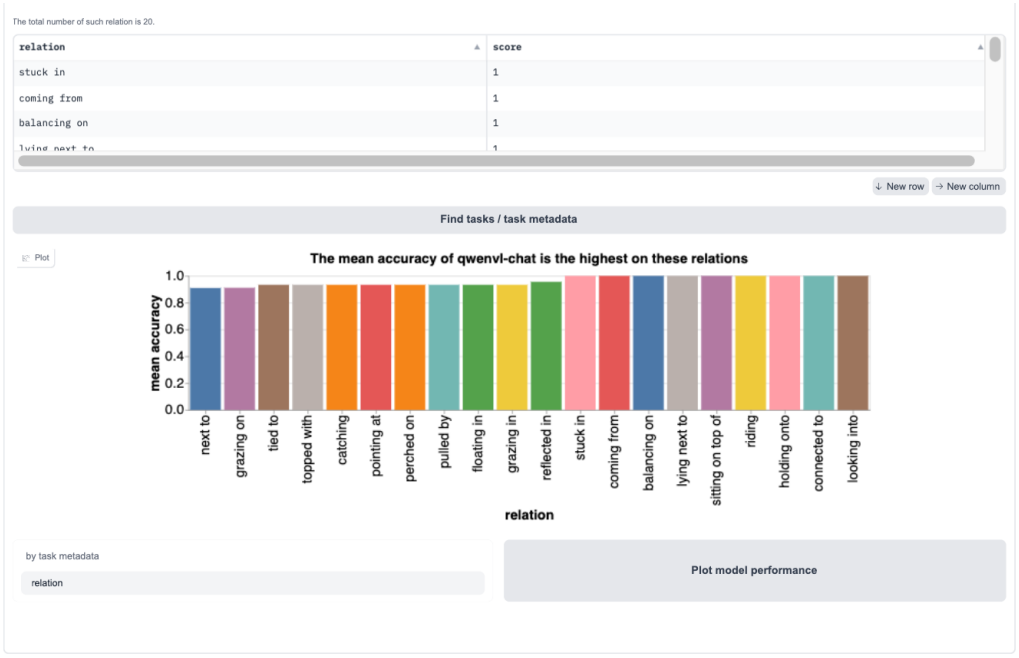


Figure 18: ImageQA: Best relations

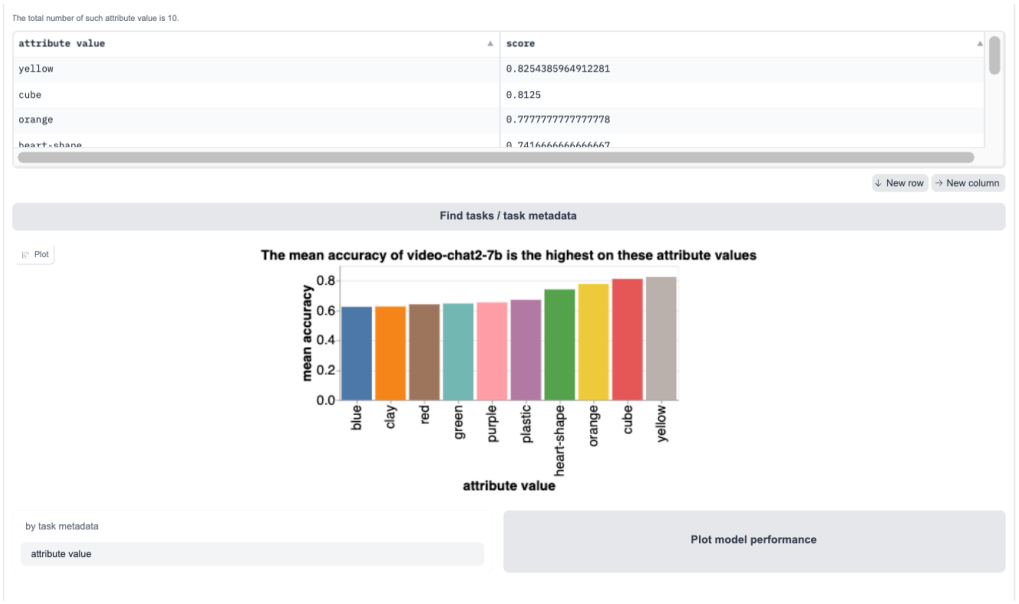


Figure 19: VideoQA: Best attributes

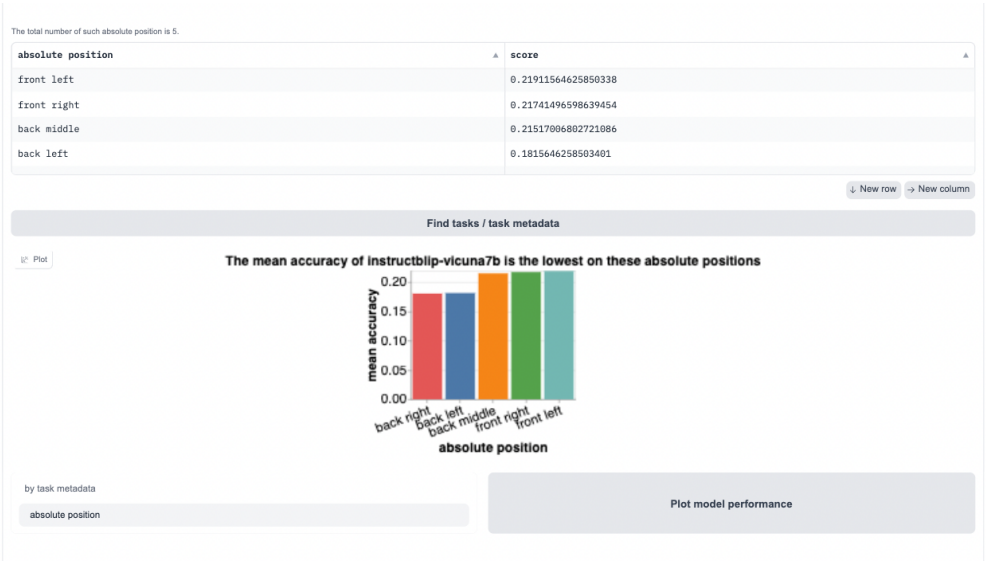


Figure 20: ImageQA: Worst positions

Figure 21: VideoQA: Worst actions

H.2 How do small models compare against large models? (continued)

As discussed in the main paper, we observe that large multi-modal models collectively perform better than smaller models on ImageQA tasks (Figure **??**). Nevertheless, this finding might not always hold for individual models. Through t-tests with pairs of small and large models from the same source, we find one exception: INSTRUCTBLIP-7B ($\mu = 0.63$) significantly outperforms INSTRUCTBLIP-13B ($\mu = 0.49$) on relation understanding (with p-value = 0) (Figure **??**).

Further, upon a closer look with our interface, we identify a few relations where INSTRUCTBLIP-7B outperforms INSTRUCTBLIP-13B by a large margin e.g. 50% (Figure 22). Similarly, we also retrieve a few actions and objects where VIDEO-LLAMA-2-7B performs much better e.g. by 20% than VIDEO-LLAMA-2-13B (Figures 23 and 24).

Figure 22: INSTRUCTBLIP-7B vs. INSTRUCTBLIP-13B relations

Figure 23: VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B actions

Figure 24: VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B objects

H.3 Do TASKVERSE yield results similar to existing benchmarks?

To check whether our TASKVERSE reflects model performance similarly to an existing benchmark, we conducted a case study testing six open-source models on both the well-known TallyQA Counting benchmark [2] (we selected 10,000 simple questions and 10,000 complex from the whole set) and 2D how-many and 3D how-many tasks in TASKVERSE-RANDOM. (Table 27), the results demonstrate a notable correlation. For instance, the LLAVA-13B is the best-performing model in both TallyQA and how-many tasks in TASKVERSE-RANDOM. The Spearman ranking coefficient for the correlation between the 2D how-many tasks and TallyQA is 0.714 (p-value = 0.111), while for the 3D how-many tasks, it is 0.543 (p-value = 0.266). These results indicate positive correlations of model performance between our tasks and existing ones, validating that TASKVERSE can effectively reflect model performance in a manner similar to existing benchmark.

Table 27: Models performance on TallyQA Counting benchmark and 2D how-many and 3D how-many in our TASKVERSE-RANDOM

Model	TallyQA	2D How Many	3D How Many
LLAVA-7B	35.90	42.00	38.67
LLAVA-13B	**38.33**	**49.33**	**46.67**
QWEN-VL	18.79	30.67	32.33
QWEN-VL-CHAT	32.07	39.67	45.00
INSTRUCTBLIP-7B	29.92	23.67	32.67
INSTRUCTBLIP-13B	33.22	26.67	32.00

I Task Generator Cards

WhatGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 2D sticker image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the object in the <absolute pos> part of the image?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the object in the bottom middle part of the image?
      - **A**: folding chair
  - **Partition 2**.
    - **Template**.
      - **Q**: What is the object <reference pos> the <reference category>?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the object to the left of the telephone?
      - **A**: table lamp
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhereGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 2D sticker image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the <target category> in the image?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the apple in the image?
      - **A**: back left
  - **Partition 2**.
    - **Template**.
      - **Q**: Where is the <target category> with respect to the <reference category>?
      - **A**: <reference position>
    - **Example**
      - **Q**: Where is the vacuum cleaner with respect to the backpack?
      - **A**: left
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatAttributeGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 2D sticker image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what attribute".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <attribute type> of the object in the <absolute position> part of the image?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the material of the object in the middle part of the image?
      - **A**: plastic
  - **Partition 2**.
    - **Template**.
      - **Q**: What is the <attribute type> of the object to the left of the <reference category>?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the color of the object to the left of the silverware?
      - **A**: gold
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhereAttributeGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 2D sticker image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "where attribute".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the <attribute value> object in the image?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the white object in the image?
      - **A**: top right
  - **Partition 2**.
    - **Template**.
      - **Q**: Where is the <attribute value> object with respect to the <reference category>?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the gray object with respect to the lollipop?
      - **A**: top
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

HowManyGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 2D sticker image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "how many".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **count** `integer`. The total number of the target objects in the image.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: How many <attribute value> objects are there in the image?
      - **A**: <count>
    - **Example**
      - **Q**: How many blue objects are there in the image?
      - **A**: 2
  - **Partition 2**.
    - **Template**.
      - **Q**: How many <target category> are there in the image?
      - **A**: <count>
    - **Example**
      - **Q**: How many tables are there in the image?
      - **A**: 4
  - **Partition 3**.
    - **Template**.
      - **Q**: How many <attribute value> <target category> are there in the image?
      - **A**: <count>
    - **Example**
      - **Q**: How many pink beverages are there in the image?
      - **A**: 2
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

What3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the object in the <absolute pos> part of the image?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the object in the front right part of the image?
      - **A**: scale
  - **Partition 2**.
    - **Template**.
      - **Q**: What is the object <reference pos> the <reference category>?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the object to the right of the mobile computer?
      - **A**: bucket
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

Where3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "where".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the <target category> in the image?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the vacuum cleaner in the image?
      - **A**: back left
  - **Partition 2**.
    - **Template**.
      - **Q**: Where is the <target category> with respect to the <reference category>?
      - **A**: <reference position>
    - **Example**
      - **Q**: Where is the vacuum cleaner with respect to the wine glass?
      - **A**: left
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatAttribute3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what attribute".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <attribute type> of the object in the <absolute position> part of the image?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the color of the object in the back left part of the image?
      - **A**: red
  - **Partition 2**.
    - **Template**.
      - **Q**: What is the <attribute type> of the object to the left of the <reference category>?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the material of the object behind the plate?
      - **A**: wood
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhereAttribute3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "where attribute".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the <attribute value> object in the image?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the wood object in the image?
      - **A**: front right
  - **Partition 2**.
    - **Template**.
      - **Q**: Where is the <attribute value> object with respect to the <reference category>?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the white object with respect to the trophy?
      - **A**: left
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

HowMany3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "how many".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **count** `integer`. The total number of the target objects in the image.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: How many <attribute value> objects are there in the image?
      - **A**: <count>
    - **Example**
      - **Q**: How many blue objects are there in the image?
      - **A**: 6
  - **Partition 2**.
    - **Template**.
      - **Q**: How many <target category> are there in the image?
      - **A**: <count>
    - **Example**
      - **Q**: How many plates are there in the image?
      - **A**: 5
  - **Partition 3**.
    - **Template**.
      - **Q**: How many <attribute value> <target category> are there in the image?
      - **A**: <count>
    - **Example**
      - **Q**: How many black furnitures are there in the image?
      - **A**: 4
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatDistance3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what distance".
  - **distance type**: `string`. The type of the distance between target object and the reference object, indicates whether it pertains to the "farthest" or "closest" distance.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the object that is <distance type> from the <reference category>?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the object that is farthest from the optical instrument?
      - **A**: juice
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhereDistance3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "where distance".
  - **distance type**: `string`. The type of the distance between target object and the reference object, indicates whether it pertains to the "farthest" or "closest" distance.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the object that is <distance type> from the <reference category> in the image?
      - **A**: <reference position>
    - **Example**
      - **Q**: Where is the object that is farthest from the bread in the image?
      - **A**: middle
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatAttributeDistance3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what attribute distance".
  - **distance type**: `string`. The type of the distance between target object and the reference object, indicates whether it pertains to the "farthest" or "closest" distance.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <attribute type> of the object that is <distance type> to the <target category>?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the color of the object that is closest to the statue?
      - **A**: beige
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatSize3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what size".
  - **size**: `string`. The type of the size of the target object, indicates whether it pertains to the "largest" or "smallest" in all the objects.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <size> object in the image?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the smallest object in the image?
      - **A**: spatula
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhereSize3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "where size".
  - **size**: `string`. The type of the size of the target object, indicates whether it pertains to the "largest" or "smallest" in all the objects.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
  - **target-reference order**: `string`. Define the target object goes first or not in the question. It is related to grammar
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the <size> object in the image?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the largest object in the image?
      - **A**: middle
  - **Partition 2**.
    - **Template**
      - **Q**: Where is the <size> object in the image with respect to the <reference category>?
      - **A**: <reference position>
    - **Example**
      - **Q**: Where is the smallest object in the image with respect to the car?
      - **A**: middle
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatAttributeSize3DGridTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what attribute size".
  - **size**: `string`. The type of the size of the target object, indicates whether it pertains to the "largest" or "smallest" in all the objects.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <attribute type> of the <size> object in the image?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the color of the smallest object in the image?
      - **A**: black
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatMovementVideoGridTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop video
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what move video".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
  - **moving direction**: `string`. The moving direction of the target object, can be either 'left', 'right', 'up', or 'down'.
  - **are other objects moving**: `string`. Indicates that other objects in the video are moving or not, can be "Yes" or "No". If it is "Yes" moving, it should not be in the same direction of the target object's moving direction.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the object that is moving <moving direction> in the video?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the object that is moving left in the video?
      - **A**: serving tray
  - **Partition 2**.
    - **Template**
      - **Q**: What is the moving object in the video?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the moving object in the video?
      - **A**: barrel
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhereMovementVideoGridTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop video
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "where move video".
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
  - **moving direction**: `string`. The moving direction of the target object, can be either 'left', 'right', 'up', or 'down'.
  - **are other objects moving**: `string`. Indicates that other objects in the video are moving or not, can be "Yes" or "No". If it is "Yes" moving, it should not be in the same direction of the target object's moving direction.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the object that is moving down located in the video?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the object that is moving down located in the video?
      - **A**: back right
  - **Partition 2**.
    - **Template**
      - **Q**: Where is the moving object located in the video?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the moving object located in the video?
      - **A**: back right
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatAttributeMovementVideoGridTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop video
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what attribute move video".
  - **size**: `string`. The type of the size of the target object, indicates whether it pertains to the "largest" or "smallest" in all the objects.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <attribute type> of the object that is moving <moving direction> in the video?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the color of the object that is moving left in the video?
      - **A**: black
  - **Partition 2**.
    - **Template**
      - **Q**: Where is the <attribute type> of the moving object in the video?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the color of the moving object in the video?
      - **A**: white
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatRotationVideoGridTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop video
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what rotate video".
  - **size**: `string`. The type of the size of the target object, indicates whether it pertains to the "largest" or "smallest" in all the objects.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <size> object in the image?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the smallest object in the image?
      - **A**: spatula
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhereRotationVideoGridTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop video
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "where rotate video".
  - **size**: `string`. The type of the size of the target object, indicates whether it pertains to the "largest" or "smallest" in all the objects.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **reference category**: `string`. The category name of the object that is used to reference the target object.
  - **reference position**: `string`. The relative position of the target object from the reference object.
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
  - **target-reference order**: `string`. Define the target object goes first or not in the question. It is related to grammar
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: Where is the <size> object in the image?
      - **A**: <absolute position>
    - **Example**
      - **Q**: Where is the largest object in the image?
      - **A**: middle
  - **Partition 2**.
    - **Template**
      - **Q**: Where is the <size> object in the image with respect to the <reference category>?
      - **A**: <reference position>
    - **Example**
      - **Q**: Where is the smallest object in the image with respect to the car?
      - **A**: middle
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatAttributeRotationVideoGridTaaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop video
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what attribute rotate video".
  - **size**: `string`. The type of the size of the target object, indicates whether it pertains to the "largest" or "smallest" in all the objects.
  - **grid number**: `integer`. The number of diagonal grids of the image, N indicates there are $N \times N$ grids in the image. Support $\{2, 3\}$.
  - **target category**: `string`. The category name of the target object.
  - **absolute position**: `string`. The absolute position of the target object in the grid. It is a number ranging from 0 to 3 (grid number = 2) or 0 to 8 (grid number = 3).
  - **attribute type**: `string`. The type of attributes of the target object, currently include: `color`, `material`, and `shape`.
  - **attribute value**: `string`. The value of the attributes of the target object.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <attribute type> of the <size> object in the image?
      - **A**: <attribute value>
    - **Example**
      - **Q**: What is the color of the smallest object in the image?
      - **A**: black
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatObjectSceneGraphTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what object".
  - **object** : `string`. The target object node of the question.
  - **subgraph** : `string`. The subgraph with the target object node as its root, used to reference the target object node.
  - **scene graph id** : `string`. The identifier of the scene graph.
  - **answers**: `list`. A list of object nodes in the scene graph that share the same subgraph structure, except the target object node and itself.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <object and its attributes in the subgraph> that <obj reference(other reference objects, attributes, and relations in the subgraph)>?
      - **A**: <target category>
    - **Example**
      - **Q**: What is the flat object that is on the brown and wood table?
      - **A**: paper
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatAttributeSceneGraphTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what attribute".
  - **attribute type** : `string`. The type of the target attribute.
  - **attribute** : `string`. The target attribute node of the question.
  - **subgraph** : `string`. The subgraph with the target attribute node as its root.
  - **scene graph id** : `string`. The identifier of the scene graph.
  - **answers**: `list`. A list of attribute nodes in the scene graph that share the same subgraph structure, except the target attribute node and itself.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the <attribute type> of the <target attribute's corresponding object and object's other attributes in the subgraph> that <obj reference(other reference objects, attributes, and relations in the subgraph)>?
      - **A**: <attribute>
    - **Example**
      - **Q**: What is the material of the smooth object that is to the right of the yellow container?
      - **A**: plastic
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatRelationSceneGraphTaskGenerator

- **Basic Information**.
  - **Task Type**. ImageQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what relation".
  - **relation**: `string`. The target relation edge between source object node and target object node
  - **source object**: `string`. The source object node of the question.
  - **target object** : `string`. The target object node of the question.
  - **source subgraph** : `string`. The subgraph with the source object node as its root.
  - **target subgraph** : `string`. The subgraph with the target object node as its root.
  - **scene graph id** : `string`. The identifier of the scene graph.
  - **answers**: `list`. A list of relation edges in the scene graph that connect the same source subgraph and target subgraph.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the relation from the <source object's attributes in the source subgraph> object, which <source obj reference(other reference objects, attributes, and relations in the source subgraph)>, to the <target object's attributes in the source subgraph> object, which <target obj reference(other reference objects, attributes, and relations in the target subgraph)>?
      - **A**: <relation>
    - **Example**
      - **Q**: What is the relation from the standing object, which the colorful and long snowboard is to the right of, to the blue and long object, which is to the left of the patterned skis?
      - **A**: holding
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatObjectVideoSceneGraphTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what object video".
  - **object** : `string`. The target object the person in the video interacts with.
  - **relation** : `string`. The relation between the person and the target object it interacts with.
  - **reference action** : `string`. The reference action to locate the moment when a person is interacting with the target object.
  - **reference type** : `string`. The target object of the relation between the person and the target object it interacts with, can be "spatial" or "contact"
  - **temporal reference type** : `string`. Type of the temporal reference between the reference action and the moment when a person is interacting with the target object. Can be "before", "while", or "after"
  - **video scene graph id** : `string`. The identifier of the video scene graph.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the object that the person is <reference> <temporal reference type> the person <reference action>?
      - **A**: <object>
    - **Example**
      - **Q**: What is the object that the person is behind after the person watching something in a mirror?
      - **A**: floor
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatRelationVideoSceneGraphTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what relation video".
  - **object** : `string`. The object the person in the video interacts by the target relation.
  - **relation** : `string`. The target relation between the person and the target object it interacts with.
  - **reference action** : `string`. The reference action to locate the moment when a person is interacting with the object.
  - **reference type** : `string`. The type of the target relation between the person and the object it interacts with, can be "spatial" or "contact"
  - **temporal reference type** : `string`. Type of the temporal reference between the reference action and the moment when a person is interacting with the object. Can be "before", "while", or "after"
  - **video scene graph id** : `string`. The identifier of the video scene graph.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What is the spatial relation of the person to the <object> while the person <reference action>.
      - **A**: <attribute>
    - **Example**
      - **Q**: What is the spatial relation of the person to the closet while the person closing a closet?
      - **A**: behind
  - **Partition 2**.
    - **Template**
      - **Q**: What is the person doing to the <object> before the person <reference action>?
      - **A**: <attribute>
    - **Example**
      - **Q**: What is the person doing to the blanket before the person putting a phone somewhere?
      - **A**: touching
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

WhatActionVideoSceneGraphTaskGenerator

- **Basic Information**.
  - **Task Type**. VideoQA
  - **Question Type**. what object
  - **Answer Type**. object category
  - **Image Type**. 3D tabletop image
  - **The model capability to evaluate**. object recognition with / without reference
- **Source Data**.
  - rendering images of objects from Objaverse
  - Annotations regarding object category, attribute, and shape
- **Task Plan Schema**.
  - **question type**: `string`. The question type of these tasks will be "what action video".
  - **action** : `string`. The target action that the person in the video performs.
  - **reference action** : `string`. The reference action to locate the moment when a person is performing the target action.
  - **temporal reference type** : `string`. Type of the temporal reference between the reference action and the moment when a person is performing the target action. Can be "before", "while", or "after"
  - **video scene graph id** : `string`. The identifier of the video scene graph.
- **Partitions**.
  - **Partition 1**.
    - **Template**
      - **Q**: What action is the person doing while <reference action>?
      - **A**: <action>
    - **Example**
      - **Q**: What action is the person doing while laughing at something?
      - **A**: sitting at a table
- **Limitations**: The current setup is primarily designed for stationary objects and may not effectively assess dynamic scenarios or human actions, such as interactions with objects or motion-based tasks.
- **Recommendations**: A task generator includes compositional and contextual challenges that require deeper reasoning about object relation and recognition.

References

[1] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019.

[2] Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions. In *AAAI Conference on Artificial Intelligence*, 2018.

[3] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European conference on computer systems*, pages 29–42, 2013.

[4] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

[5] Blender Online Community. Blender - a 3d modelling and rendering package, 2018.

[6] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

[8] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

[9] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. *arXiv preprint arXiv:2404.12390*, 2024.

[10] Xiaoyu Ge and Panos K Chrysanthis. Efficient prefdiv algorithms for effective top-k result diversification. In *EDBT*, pages 335–346, 2020.

[11] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[12] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224. IEEE, 2001.

[13] Jiawei Han, Jian Pei, and Hanghang Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[14] Dong He, Maureen Daum, Walter Cai, and Magdalena Balazinska. Deepeverest: Accelerating declarative top-k queries for deep neural network interpretation. *Proc. VLDB Endow.*, 15(1):98–111, 2021.

[15] Dong He, Jieyu Zhang, Maureen Daum, Alexander Ratner, and Magdalena Balazinska. Masksearch: Querying image masks at scale. *arXiv preprint arXiv:2305.02375*, 2023.

[16] Kohei Hirata, Daichi Amagata, Sumio Fujita, and Takahiro Hara. Solving diversity-aware maximum inner product search efficiently and effectively. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 198–207, 2022.

[17] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[18] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020.

[19] Yu Jiang, Guoliang Li, Jianhua Feng, and Wen-Syan Li. String similarity joins: An experimental evaluation. *Proc. VLDB Endow.*, 7(8):625–636, apr 2014.

[20] Peng Jin, Ryuichi Takanobu, Caiwan Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. *arXiv preprint arXiv:2311.08046*, 2023.

[21] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[22] Daniel Kang, Peter Bailis, and Matei Zaharia. Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046*, 2018.

[23] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[24] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench-2: Benchmarking multimodal large language models. *arXiv preprint arXiv:2311.17092*, 2023.

[25] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.

[26] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. Mvbench: A comprehensive multi-modal video understanding benchmark, 2023.

[27] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[28] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[29] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023.

[30] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv:2306.05424*, 2023.

[31] Willi Mann, Nikolaus Augsten, and Panagiotis Bouros. An empirical evaluation of set similarity join techniques. *Proc. VLDB Endow.*, 9(9):636–647, may 2016.

[32] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[33] Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Proceedings of the 35th Conference on Neural Information Processing Systems Track on Datasets and Benchmarks*, December 2021.

[34] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350, 2017.

[35] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 510–526. Springer, 2016.

[36] Jacob VanderPlas, Brian Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, Arvind Satyanarayan, Eitan Lees, Ilia Timofeev, Ben Welsh, and Scott Sievert. Altair: Interactive statistical visualizations for python. *Journal of Open Source Software*, 3(32):1057, 2018.

[37] Manasi Vartak, Joana M F. da Trindade, Samuel Madden, and Matei Zaharia. Mistique: A system to store and query model intermediates for model diagnosis. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1285–1300, 2018.

[38] Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *Proceedings. 20th international conference on data engineering*, pages 79–90. IEEE, 2004.

[39] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR*, 2024.

[40] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.