

Towards Grounded Visual Understanding in Vision–Language Models

Jieyu Zhang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2026

Reading Committee:
Ranjay Krishna, Co-Chair
Alexander J. Ratner, Co-Chair
Ali Farhadi
Luke Zettlemoyer

Program Authorized to Offer Degree:
Computer Science and Engineering

© Copyright 2026

Jieyu Zhang

University of Washington

Abstract

Towards Grounded Visual Understanding in Vision–Language Models

Jieyu Zhang

Co-chairs of the Supervisory Committee:

Assistant Professor Ranjay Krishna
Computer Science and Engineering

Affiliate Assistant Professor Alexander J. Ratner
Computer Science and Engineering

Vision-language models have made rapid progress, but strong performance on standard benchmarks does not necessarily indicate grounded visual understanding: the ability to connect language to the actual objects, attributes, relations, and dynamics present in visual input. This thesis studies how to build vision-language models with stronger grounded visual understanding, with contributions spanning evaluation, supervision, representation, and reasoning. First, this thesis shows that existing benchmarks can overestimate grounded understanding because models may exploit dataset artifacts rather than visual evidence. To address this limitation, it develops more reliable and fine-grained evaluation tools, including benchmarks designed to reduce annotation bias and a programmatic benchmark generation engine that enables query-driven analysis of model capabilities across a broad range of image and video tasks. Second, this thesis studies how richer supervision can improve grounding. It introduces a scalable pipeline for generating vision-language instruction data from structured visual representations, covering objects, attributes, relations, segmentation, and depth. The resulting large-scale instruction data substantially improves performance on diverse visual understanding benchmarks. Third, this thesis investigates visual representation learning for grounding, with a focus on tokenizations that better reflect underlying visual structure. In video, it develops object- and trajectory-centric tokenization methods that replace standard patch-based representations with more grounded

visual units, improving efficiency and downstream performance while better preserving object identity over time. Finally, this thesis shows how grounding can serve as a foundation for multimodal reasoning. It presents open vision-language models that support pointing, tracking, and spatially precise reasoning over images and videos, enabled by new grounded datasets and training strategies. Together, these contributions advance a unified view of grounded visual understanding: models should not only describe visual content, but also localize, track, and reason over it in ways that are faithful to the depicted world. This thesis argues that such grounding is essential for making vision-language models more reliable, interpretable, and useful in real-world settings.

Acknowledgements

The past six years of my PhD journey have been an incredible and truly once-in-a-lifetime experience. Looking back on this journey, I can hardly believe how fortunate I have been to work with my two advisors: Ranjay and Alex. In every sense, they are the best advisors I could have ever hoped for. Their brilliance, kindness, patience, and unwavering support have shaped not only this dissertation, but also the researcher and person I have become along the way. I feel deeply grateful to have shared this journey with them.

I am deeply grateful to Ranjay for his support, patience, and constant encouragement throughout my PhD. At a pivotal moment in my journey, his willingness to welcome, trust, and invest in me made a profound difference in the path that followed, and it is something I will always be thankful for. He entrusted me with opportunities that profoundly shaped my growth, including the freedom to lead major projects, mentor students, and collaborate with remarkable people from around the world. His trust gave me the confidence to take ownership of ambitious ideas and to grow into a more independent and creative researcher. At the same time, he was always incredibly sharp in his feedback, with an ability to quickly see the heart of a problem, identify what truly mattered, and push an idea toward greater clarity and ambition. His mentorship extended far beyond research alone: he taught me how to write, how to present, how to communicate ideas effectively, and how to carry myself as a member of the vision community that I have grown to love so much. More than anyone, he introduced me to the world of computer vision and made me feel excited to be part of it. One of the most important ways he shaped my thinking was by teaching me to approach research from a cognitive science perspective, and to connect model building to deeper questions about cognition and intelligence. That way of thinking has become one of the most valuable mindsets I carry with me in research. Just as importantly, he gave me the space to make mistakes, learn from them, and develop my own creativity and independence as a researcher. One memory that stays with me is that, in the middle of my PhD,

he encouraged me to pursue a certain direction that I did not fully appreciate at the time. By the end of my PhD, I found myself returning to that very direction and recognizing how promising it truly was. In hindsight, I wish I had understood his vision earlier, and that feeling has only deepened my appreciation for his insight, generosity, and belief in me.

I am deeply grateful to Alex, who first opened the door to this journey for me. His early belief in me is something I will always carry with gratitude, because it gave me the opportunity to grow, explore, and become the researcher I am today. Throughout my PhD, he was a supportive mentor whose thoughtful and critical feedback constantly challenged me to think more deeply and more rigorously about my work. He also brought me into the machine learning community and helped me feel that I belonged in it, which meant a great deal to me as I grew into a researcher. From him, I learned how central data is to machine learning research, and that lesson has profoundly shaped the way I think about research. He taught me that good research does not begin only with models, but with understanding the data itself: what it captures, what it misses, what assumptions it carries, and how it ultimately defines the questions we are able to ask and answer. More broadly, he taught me how to think from a learning perspective: to ground ideas in evidence, to ask precise questions, and to build research on strong empirical foundations. That way of thinking has deeply influenced my work throughout the years and remains one of the most important things I carry with me from my PhD. I will always be thankful for his support, his guidance, and the confidence he placed in me from the very beginning.

I would like to sincerely thank my PhD committee—Ali, Luke, and Zaid—for their time, thoughtful feedback, and generous support throughout the development of this thesis. Their insightful questions and suggestions consistently pushed me to think more deeply and carefully about my work. Our discussions not only helped shape this thesis, but also influenced how I think about the future directions of my research. I am truly grateful for their guidance, encouragement, and generosity throughout this process.

I am also grateful to my internship mentors, who welcomed me into their teams and gave me the opportunity to grow in new environments beyond my home institution. Their guidance, generosity, and encouragement broadened my perspective on research and helped me become a more confident and mature researcher. Through working with them, I learned not only new technical ideas and ways of approaching problems, but also how different research cultures can inspire new ways of thinking. I am thankful for the

trust they placed in me, for the time they invested in mentoring me, and for the opportunities they gave me to learn from them so closely. Thank you, Lei, Yujing, Chi, Le, Zeyuan, Ran, Juan Carlos, and Caiming.

I have been incredibly fortunate to work with so many wonderful mentors beyond my thesis committee and internships, as well as with collaborators throughout my PhD. I am grateful to Ludwig, Pang Wei, Chao, Manling, Jieyu, Qingyun, Shanghang, Tianyi, Aniruddha, and Bharath for your generosity, insight, and intellectual guidance, which broadened the way I think about research and introduced me to new ideas and perspectives. I am equally thankful to the many student collaborators and friends I worked with along the way, including Cheng-yu, Jeffrey, Chenhao, Mahtab, Yue, Shaokun, Hejie, Yuanqi, Yanqiao, Bohan, Haonan, Yuchen, Linxin, Chengkai, Zhengyu, Weikai, Ziqi, Linxing, Weijia, and many others. Working alongside them made this journey not only intellectually rewarding, but also deeply joyful, and I will always treasure the ideas, friendship, and shared excitement we experienced together. I learned so much from our conversations, our brainstorming sessions, and even the challenges we worked through together. Many of the most meaningful experiences of my PhD came from these collaborations, and I am sincerely grateful to everyone who worked with me, challenged me, and made this journey richer and more joyful.

I would like to sincerely thank the UW CSE staff, Joe, Elise, and others for all of their help throughout my graduate studies. Their prompt responses, patience, and willingness to assist with countless administrative matters made a real difference over the years. I am deeply grateful for their support and for the important role they played in making this journey possible.

I would also like to thank my girlfriend, Zixian, for her companionship and support throughout this journey. We have shared countless memorable moments together—working on projects side by side, interning at the same place multiple times, and traveling across the country for both fun and conferences. These moments reminded me that life during a PhD is about more than working every minute, and they brought balance, joy, and warmth to a journey that might otherwise have felt consumed by work. They are memories I will always treasure.

Finally, I want to thank my family for their love, support, and belief in me throughout all these years. Their encouragement has been a constant source of strength, especially during the moments when this journey felt uncertain or difficult. Even when they may not have fully seen every detail of what my work involved, they always stood behind me with patience, care, and unwavering faith. Their love gave me the foundation

that made everything else possible.

As my final year approached and I began to truly realize that I would soon graduate, I also came to understand more deeply than ever that this PhD has been the best job in the world for me. Because of UW, because of this group, and because of the extraordinary privilege of working with my two advisors, research became not just work, but a source of joy, meaning, and belonging. These years have shown me how rare and beautiful it is to spend each day thinking, learning, building, and discovering in a community that I care about so deeply. That is why it is so hard to say goodbye in any sense. I am not only leaving behind a degree, but also a group, a home, and a way of life that has shaped me profoundly and that I will always treasure.

Contents

1	Introduction	29
1.1	Background and Motivation	29
1.2	Grounded Visual Understanding in Vision-Language Models	30
1.2.1	What Would Improved Grounding Enable?	32
1.3	Challenges in Achieving Grounded Visual Understanding	33
1.3.1	Inadequate Evaluation of Grounding	34
1.3.2	Weak or Indirect Supervision for Grounding	34
1.3.3	Visual Representations Are Not Always Grounding-Friendly	34
1.3.4	Temporal and Dynamic Grounding Is Harder than Static Grounding	35
1.3.5	Grounded Multimodal Reasoning Remains an Open Challenge	35
1.4	Thesis Statement and Overview	35
1.5	Chapter Contributions	36
1.5.1	Evaluating Grounded Visual Understanding	36
1.5.2	Improving Grounded Visual Understanding with Rich Visual Instructions	37
1.5.3	Realizing Grounded Visual Understanding through Grounded Visual Tokenization	37
1.5.4	Unlocking Grounded Multimodal Reasoning	38
2	Evaluating Grounded Visual Understanding	39
2.1	Introduction	39
2.2	Related Work	40
2.3	SugarCrep: Fixing Hackable Compositionality Benchmarks	41

2.3.1	Biases in Current Benchmarks	41
2.4	SugarCrepe: Evaluations	47
2.4.1	Reducing Dataset Biases	47
2.4.2	Re-evaluating VL Compositionality	49
2.4.3	Limitation and future work	53
2.4.4	Societal impact	54
2.5	Task-Me-Anything: A Benchmark Generation Engine	55
2.5.1	Taxonomy and Task Generators	55
2.5.2	Query-Centric Evaluation	57
2.6	Task-Me-Anything: Analysis of MLM Capabilities	62
2.6.1	Fine-Grained Capability Profiling	63
2.6.2	Validation	72
2.7	Conclusion	74
3	Improving Grounded Visual Understanding with Rich Visual Instructions	77
3.1	Introduction	77
3.2	Related Work	79
3.3	ProVision: Rich Visual Instruction Data for Grounded Understanding	81
3.3.1	Generating vision-centric instructions programmatically	81
3.3.2	Generating scene graph for any image	82
3.3.3	PROVISION-10M Dataset	83
3.4	Experiments	84
3.4.1	Instruction Tuning	85
3.4.2	Pre-training vs. Instruction Tuning	88
3.5	Conclusion	90
4	Realizing Grounded Visual Understanding through Grounded Visual Tokenization	91
4.1	Introduction	91
4.2	Related Work	94

4.3	Grounded Visual Tokenization	95
4.3.1	Grounded Visual Tokenization with External Vision Models	95
4.3.2	End-to-End Grounded Visual Tokenization	100
4.4	Experiments	103
4.4.1	Training a Video Encoder from Scratch	104
4.4.2	Adapting Pretrained Features	110
4.4.3	Vision-Language Models	111
4.4.4	Ablation Studies	115
4.5	Conclusion	119
5	Unlocking Grounded Multimodal Reasoning in Vision-Language Models	121
5.1	Introduction	121
5.2	Related Work	123
5.3	Molmo 2: Grounded Multimodal Reasoning in Practice	124
5.3.1	Data	124
5.3.2	Training	127
5.4	Evaluation	131
5.4.1	Overall results	131
5.4.2	Grounding results	132
5.4.3	Ablations and specialized models	134
5.5	Conclusion	136
6	Conclusion	137
A	Appendix: SugarCrepe	171
A.1	Implementation details	171
A.1.1	Hardware information	171
A.1.2	Dataset sources	171
A.1.3	Software configuration	172
A.2	Vision-language compositionality benchmarks	172

A.2.1	Image-to-text formulation	172
A.2.2	Text-to-image formulation	173
A.3	SUGARCREPE	174
A.3.1	Taxonomy	174
A.3.2	Hard negative generation procedure and templates	174
A.3.3	Adversarial refinement	176
A.3.4	Dataset construction cost	176
A.3.5	Dataset information	176
A.4	Detailed evaluation results	177
A.4.1	Full evaluation results on existing benchmarks	177
A.4.2	SUGARCREPE human evaluation	177
A.4.3	Additional NEGCLIP results	178
B	Appendix: Task-Me-Anything	183
B.1	Details of Task Generation	183
B.1.1	Key concepts	183
B.1.2	The generation process	184
B.2	Details of Fine-grained User Query and Query Approximation Algorithms	187
B.2.1	Fine-grained user query	187
B.2.2	Query execution	188
B.2.3	Efficient Query Approximation Algorithms	190
B.3	Details of TASK-ME-ANYTHING 1.0	192
B.3.1	Source data	192
B.3.2	Task generators for different scenarios	193
B.3.3	TASK-ME-ANYTHING-UI	196
B.4	Details of Model and Human Performance on Random Task Instances	200
B.4.1	Raw results of Figure 2.10	200
B.4.2	A breakdown of Table B.6	202
B.4.3	A breakdown of Table B.7	205

B.5	Details of Experiments on Query Results Approximation Algorithms	207
B.5.1	Experiment details	207
B.5.2	Experiments on approximations under different budgets.	209
B.5.3	Query results approximation experiments in ImageQA	211
B.5.4	Query results approximation experiments in VideoQA	213
B.6	Details of Analysis and Case Study	215
B.6.1	What task metadata are models good or bad at?	215
B.6.2	How do small models compare against large models? (continued)	218
B.6.3	Do TASK-ME-ANYTHING yield results similar to existing benchmarks?	220
C	Appendix: ProVision	221
C.1	Instruction Data Generator Statistics	221
C.2	Example Generated Instructions	221
C.3	Results with Model-Generated Scene Graphs	221
D	Appendix: TrajViT and TrajTok	225
D.1	Segmenter Training Details	225
D.1.1	Dataset Construction	225
D.1.2	Training Configuration	226
D.2	More Qualitative Examples of the Segmenter	226
D.3	Quantitative Evaluation of the Segmenter	228
D.4	Training Details for TrajViT2	229
D.5	Training Details for TrajAdapter	230
D.6	Training Details for TrajVLM	230
E	Appendix: Molmo2	233
E.1	Model Architecture and Training Details	233
E.2	Image Benchmark Results	233
E.3	NLP Benchmark Results	233
E.4	Additional Video Counting and Pointing Results	233

E.5	Tracking Dataset Statistics	235
E.6	Pre-training Ablation	235
E.7	Long Context Training	236
E.8	Image-Only Ablation	236
E.9	Additional Video Model Ablations	236

List of Figures

2.1	Top row: We define <i>Vera score gap</i> as the score difference between the positive and hard negative texts: $Vera(T^p) - Vera(T^n)$. The entire Vera score gap distribution lies on the positive spectrum, indicating that the template-generated hard negative texts usually have low plausibility. Bottom row: Similarly, <i>Grammar score gap</i> is defined by: $Grammar(T^p) - Grammar(T^n)$. On grammar score, we also find that the distribution largely rests on the positive side, suggesting that most hard negative texts in existing benchmarks exhibit grammatical errors.	43
2.2	Blind commonsense Vera model and Grammar model outperform state-of-the-art CLIP models on nearly <i>all</i> existing benchmarks by exploiting the nonsensical and non-fluent artifacts. This suggests that existing benchmarks are hackable and ineffective in measuring compositionality.	43
2.3	Example prompt (black) and actual hard negative (green) generated from ChatGPT.	45
2.4	We compare the Vera (top row) and Grammar (bottom row) score gap distributions between ARO+CREPE (leftmost column), SUGARCREPE without adversarial refinement (middle), and SUGARCREPE (rightmost). Top row: We see that Vera score gap distribution shifts from the positive spectrum to more centered around zero from ARO+CREPE to SUGARCREPE without refinement. After adversarial refinement, we ensure the score gap distribution is centered around zero on SUGARCREPE. Bottom row: Similarly, from ARO+CREPE to SUGARCREPE, we see the Grammar score gap distribution shifts from the positive spectrum to centered around zero.	49

2.5	We plot pretrained vision-language models' zero-shot top-1 accuracy on ImageNet versus their retrieval recall@1 on SUGARCREPE, where r is the Pearson correlation coefficient. This plot suggests that models' ImageNet zero-shot accuracy positively correlates with their compositionality.	53
2.6	We present the key components in TASK-ME-ANYTHING. The top part illustrates the task generation process with an example video synthesized with 3D objects and their annotations, and the task generator for generating questions about rotating objects' attributes. The bottom part depicts the model evaluation process, which selects the relevant tasks based on the user's query and their budget and performs either full evaluation or results approximation to answer the query.	56
2.7	The statistics of generatable tasks of each task generator and example image/video in TASK-ME-ANYTHING. We each task generator with high-level perceptual skills and this collection of task generators can collectively generate over 750M VQA tasks.	59
2.8	We adopt two distinct prompts, the detailed prompt and the succinct prompt, in our evaluation to assess models' sensitivity to different prompts.	61
2.9	TASK-ME-ANYTHING-UI Interface.	63
2.10	Model performance on the TASK-ME-ANYTHING-RANDOM, a random subset of tasks from TASK-ME-ANYTHING.	64
2.11	The absolute difference of model performance in different prompts in TASK-ME-ANYTHING-RANDOM.	65
2.12	Image and VideoQA, high-level skills, all models. We plot models' performance on Image and VideoQA tasks across all skills. We learn that models are relatively good at object and attribute recognition in both Image and VideoQA and relation understanding in ImageQA but still struggle at others.	66
2.13	Image and VideoQA, high-level skills, open-source vs. proprietary best models. We plot the performance of the best open-source and proprietary model for each skill on ImageQA and VideoQA tasks.	67

2.14	ImageQA, fine-grained skills, all models. We also analyze models’ performance on ImageQA tasks across fine-grained skills and find that models are good at recognizing plants, understanding mood, and comprehending spatial relations between real-world objects on average despite differences in individual models.	67
2.15	VideoQA, fine-grained object and attribute skills, all models. We present models’ performance on VideoQA tasks across fine-grained skills and find that, on average, models are good at recognizing vehicles and understanding materials in videos.	68
2.16	VideoQA, fine-grained relation skills, all models. On VideoQA tasks, we find that models are better at understanding spatial relations than interactional ones on average.	68
2.17	Skill comparison: small vs. large models on ImageQA.	69
2.18	Skill comparison: small vs. large models on VideoQA.	69
2.19	Skill comparison: INSTRUCTBLIP-7B vs. INSTRUCTBLIP-13B.	69
2.20	Skill comparison: VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B.	69
2.21	Skill comparison: CHAT-UNIVI-7B vs. CHAT-UNIVI-13B.	70
2.22	We present models’ performance for each skill across visual inputs.	70
2.23	Answering Q1-Q3 with GPT4O performance on randomly generated task instances relating to coarse-grained object/relation/attribute categories.	71
2.24	Correlations of TASK-ME-ANYTHING with other popular MLM benchmarks.	74
2.25	Correlations of synthetic data with real data in TASK-ME-ANYTHING.	74
3.1	Overview of PROVISION system. It can generate a scene graph for any image, enabling programs to synthesize instruction data procedurally from the scene graph. If the scene graph exists, it directly generates instruction data based on it.	79
3.2	We visualize the instruction data generation process and generated examples for both single and multiple image scenarios. Single-image instruction data can be categorized into six dimensions, aiming to improve the model’s ability in retrieving and understanding the annotations. We divide multiple-image instruction data into three categories that help the model understand the relation between different images from the perspective of the relation between their scene graph.	80

3.3	Our scene graph generation pipeline. For each input image, it will generate five annotations, including object, depth, segmentation, attribute, and relation, collectively forming an augmented scene graph.	82
4.1	(a) Traditional video tokenization splits a video into space-time patches, introducing large number of redundant tokens. (b) We propose to represent a video via panoptic sub-object trajectory (TrajViT), which significantly reduces redundancy but relies on slow, non-differentiable pipelines. (c) We further propose TrajTok, an end-to-end differentiable trajectory tokenizer that learns to implicitly propose trajectory tokens, offering low token counts, efficiency and adaptability to downstream objectives.	91
4.2	Overview of TrajViT. Given a video, we first panoptically extract the trajectories for all objects. Our trajectory encoder converts these dynamic object trajectories into fixed sized embeddings, which serve as the input to the transformer encoder.	95
4.3	Our parallel trajectory generation pipeline. We use key frame detection to break a video into subclips. We segment and track objects in each clip in parallel and finally merge objects between clips. This paradigm captures objects that emerge over time while reducing overall tracking latency.	96
4.4	Visualizations of generated trajectories.	96
4.5	Architecture of trajectory encoder. we employ a two-branch design that encodes a trajectory’s appearance and temporal position separately. At each frame, we represent the appearance of a segment by mask pooling its feature, and represent its position by bounding box coordinates. Both features are then aggregated across frames via perceiver resampler and added together to form the trajectory feature.	98
4.6	Training with downstream understanding tasks reshapes the segmentation granularity. We visualize the trajectory masks produced by our segmenter when trained with only segmentation supervision versus jointly with segmentation and CLIP objectives. The CLIP objective reshapes the segmentation granularity, producing finer foreground object masks while merging background regions.	100

4.7	Overview of the TrajTok architecture. TrajTok comprises a <i>trajectory segmenter</i> and a <i>trajectory encoder</i> . The segmenter proposes trajectory masks for all objects in an image or video within a single forward pass. The encoder then aggregates raw video pixels or encoded visual features (parameterized by f in the figure) according to these masks to produce trajectory tokens. The number of tokens per trajectory can be flexibly adjusted based on the available compute budget.	101
4.8	TrajTok is a versatile module applicable across pretraining, feature adaptation, and finetuning stages. We demonstrate its use in three scenarios: TrajViT2, which trains a visual encoder from scratch; TrajAdapter, which adapts pretrained features for downstream tasks; and TrajVLM, which uses TrajTok as a connector in LLaVA-style large vision–language models.	104
4.9	Comparison of inference frame number scaling in ActivityNet video-to-text retrieval task. Scaling with our tokenization paradigm obtains a better trade-off than baselines in terms of efficiency and accuracy.	108
4.10	Scaling with video training data. TrajViT2 exhibits stronger scaling behavior than TrajViT and sustains a consistent performance margin over ViT3D at every data scale.	108
4.11	Test time FLOPs comparison under different frame numbers.	109
4.12	Incorporating image data. TrajViT benefits more from adding image data into pretraining as it requires no architectural modifications.	110
4.13	Accuracy and inference FLOPs at MovieChat long video benchmarks with input frame scaling. The VideoLLM with TrajViT as video encoder scales significantly better than the one with ViT3D in both accuracy and efficiency.	113
4.14	VideoQA results for TrajTok applying to large vision-language model. VLM with TrajTok as connector (TrajVLM) notably outperforms patch pooling baseline (PatchVLM) in long-video benchmarks, while the performance is mixed for short-video benchmark.	113
4.15	Trajectory masks produced by our segmenter vs. by TrajViT pipeline. While our segmenter produces coarser masks and may miss very small objects, it demonstrates strong semantic grouping ability that is sufficient for downstream understanding tasks.	118

5.1	Molmo2 is trained on the largest fully open video-centric multimodal corpus to date, including nine new datasets for dense video captioning, long-form and long-video QA, and open-vocabulary pointing and tracking over images, multi-images, and videos. Molmo2 accepts single images, image sets, and videos as input and can produce both free-form language and grounded outputs such as spatio-temporal points, object tracks, and grounded chain-of-thoughts that localize objects and events over time. Across diverse video-language and grounding benchmarks, Molmo2 matches or surpasses prior open models, approaches proprietary systems, and remains fully open.	124
5.2	Molmo2 follows the standard design of connecting a vision encoder and a language model to process video inputs.	128
5.3	Attention mask for a <i>packed</i> sequence with two examples. The first contains two QA pairs for one image. Frame tokens (dark pink) have forward attention, while masking blocks cross-attention between different examples (lower-left empty block) and between distinct QA pairs within the same example (upper empty block).	129
A.1	Taxonomy of hard negatives considered in SUGARCREPE.	175
A.2	Example prompt templates (black) and outputs (green) from ChatGPT for REPLACE hard negatives.	180
A.3	Example prompt templates (black) and outputs (green) from ChatGPT for SWAP hard negatives.	181
A.4	Example prompt templates (black) and outputs (green) from ChatGPT for ADD hard negatives.	181
B.1	An illustration of core concepts and the task generation process.	186
B.2	An illustration of the query execution process.	189
B.3	Summary of task generator card sections and suggested prompts for each.	192
B.4	TASK-ME-ANYTHING-UI Interface.	197
B.5	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.	211
B.6	Threshold Query. These three bar graphs display the performance of three query approximation methods in Threshold Query, measured by Precision, Recall, and F1-score.	211

B.7	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.	211
B.8	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.	212
B.9	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.	213
B.10	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.	213
B.11	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.	213
B.12	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.	214
B.13	ImageQA: Best relations	215
B.14	VideoQA: Best attributes	216
B.15	ImageQA: Worst positions	216
B.16	VideoQA: Worst actions	217
B.17	INSTRUCTBLIP-7B vs. INSTRUCTBLIP-13B relations	218
B.18	VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B actions	219
B.19	VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B objects	219
D.1	Qualitative Examples of the trajectory masks produced by our segmenter.	227

List of Tables

2.1	Existing compositionality benchmarks rely on procedurally-generated hard negatives which often do not make logical sense or are not fluent due to grammatical errors.	42
2.2	We report the number of hard negative captions of all types in SUGARCREPE.	47
2.3	We present example positive texts and their hard negatives in ARO+CREPE (generated using existing procedures) and SUGARCREPE (generated with ChatGPT). SUGARCREPE brings significant improvements in commonsense and fluency.	48
2.4	We compare the commonsense and grammar scores on hard negatives in ARO+CREPE and SUGARCREPE. We report both their respective average scores and the ratio where SUGARCREPE has higher score than ARO+CREPE in pairwise comparison. Overall, SUGARCREPE has hard negatives with better commonsense and grammar.	48
2.5	Re-evaluating hard negative augmented training shows that the method’s improvements on existing benchmarks (ARO+CREPE) are hugely overestimated, particularly when the test hard negative type matches the one used in training, which can be attributed to overfitting the artifacts. Color notations: Gains compared to standard CLIP (finetuned / from scratch) > 10% .	51
2.6	Our evaluation of pretrained CLIP models on SUGARCREPE shows that they demonstrate compositionality on some hard negatives but are far from human performance on others, especially on SWAP hard negatives or ones perturbing attributes and relations (also illustrated in Figure 2.5: lower overall performance on SWAP, and lower performances on attributes/relations compared to objects). We additionally evaluate recently introduced GPT-4V. While it demonstrates strong results, there is still gap to human-level performance.	52

2.7	Answering Q1-Q3 with Top-K query regarding individual objects/relations/attributes. We also present the GPT4O performance drop (Δ Perf. (%)) on task instances involving found task elements as ground truth answers compared to random task instances, and show that performance drops by a large margin.	72
2.8	The performance of query results approximation algorithms. Top-K query uses Mean Rank (MR, lower is better) and Hit Rate (HR, higher is better) as metrics, while other queries use Precision (P), Recall (R), and F1-score (F1).	73
3.1	Results of instruction tuning LLaVA-1.5-7B with VG-S , i.e. , single-image instruction data generated from the Visual Genome images and scene graphs. *: data ratio = number of our data added / size of LLaVA-1.5 instruction data.	86
3.2	Results of instruction tuning Mantis-SigLIP-8B with VG-M , i.e. , multi-image instruction data generated from the Visual Genome images and scene graphs. *: data ratio = number of our data added / size of Mantis instruction data.	88
3.3	Comparison of augmenting the base data recipe with our data (both DC-S and VG-S) in pre-training, fine-tuning, or both. The first row represents the baseline model’s performance, which uses only the base recipes for both stages without our data. The results demonstrate that augmenting the base data recipes with our data in either stage (pre-training or fine-tuning) improves the model’s average performance across 11 benchmarks, with augmentation in both stages generally achieving the highest average performance. When augmented with 1.5 million samples from VG-S , our model shows an average improvement of approximately 1.4% over the baseline across all benchmarks. Bold values indicate the highest score for each benchmark.	89
3.4	Impact of dataset augmentation scale on model performance. This table compares the model’s performance across the same 11 benchmarks, with varying scales of data augmentation during pre-training: no augmentation, 0.75 million samples, and 1.5 million samples from VG-S	90
4.1	Zero-shot video & image retrieval performance (R@5) . TrajViT2 consistently surpasses all baselines by a large margin.	107

4.2	Action & image classification (attentive probe, top-1). IN-1K stands for ImageNet-1K. TrajViT2 improves upon TrajViT and outperforms all baselines in most of the benchmarks.	107
4.3	Probing results on different video backbones. Top-1 accuracy (%) on Kinetics-400 (K400) and Something-Something-V2 (SSv2) using various probing strategies. TrajAdapter consistently improves the downstream probing accuracy of pretrained backbone features, and performance further increases with more tokens per trajectory.	110
4.4	Comparison of ViT3D and TrajViT on VideoLLM QA tasks. We report results on NextQA (WUPS score in open-ended subsets, accuracy in multi-choice subsets), TempCompass (accuracy for four subsets of multi-choice, yes/no, caption matching, and captioning), VideoMME (accuracy), MovieChat (accuracy and GPT score), ActivityNetQA (accuracy and GPT score), and MLVU (accuracy). The VideoLLM that uses our video encoder consistently outperforms the one with ViT3D across all video QA benchmarks.	112
4.5	Individual impact of segmentation and tracking. We can see that removing segmentation (tracking-only) or tracking (segmentation-only) significantly hurts model performance. . . .	117
4.6	Ablation study of both trajectory generation pipeline and trajectory encoder architecture.	117
4.7	Ablations of segmenter design.	118
4.8	Ablations of trajectory encoder design.	118
5.1	Video benchmark results for a range of proprietary APIs, open-weight baselines, video-specialized models, and our Molmo2 family across video understanding, captioning, and counting benchmarks. The result of the best-performing open-weight model is in bold	131
5.2	Video counting and pointing results. Molmo2 scores highest on BURST-VC and Molmo2-VP and second highest on Molmo2-VC’s close accuracy, slightly behind Gemini 2.5 Pro. . . .	133
5.3	Video object tracking results. Molmo2 outperforms API-based and open models across all benchmarks. F1 scores are omitted for VOS models that only generate a single mask. . . .	134
5.4	Video ablations. For ablations (a)(b)(c) we train models on only video data; ablation (d) has models with only video captions.	135
5.5	Counting and pointing ablations. BVC represents Burst-VideoCount accuracy; and MVC and MVP are Molmo2-VideoCount accuracy and Molmo2-VideoPoint F1 on the validation sets.	135

5.6	Tracking ablations. We report average metrics across the five tracking benchmarks (the valid-u split for MeViS). HOTA measures association accuracy.	136
A.1	Summary on vision-language compositionality benchmarks. SUGARCREPE considers image-to-text formulation to enable larger scale evaluation set. In addition, SUGARCREPE considers a wide range of hard negative types. SHUFFLE and NEGATE are omitted as they introduce inevitable biases discussed in Sec. 2.3.1.	174
A.2	Blind models (i.e. , Vera and Grammar model) outperform all 17 existing pretrained CLIP models on nearly all existing benchmark tasks. This implies that current benchmarks fail to faithfully measure a model’s vision-language compositionality.	178
A.3	Human evaluation results on the comparisons between hard negatives in ARO+CREPE and SUGARCREPE. We report the counts (out of 100 sampled examples) that the human user considers better or tie, w.r.t. both commonsense and grammatical correctness.	178
A.4	Model performances on SUGARCREPE when trained with hard negatives generated through similar procedure as how SUGARCREPE is created.	179
B.1	<i>2D sticker image</i>	194
B.2	<i>3D tabletop scene</i> with images	195
B.3	<i>3D tabletop scene</i> with videos	196
B.4	Real images with <i>Scene Graph</i>	196
B.5	Real videos with <i>Scene Graph</i>	196
B.6	TASK-ME-ANYTHING-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.	200
B.7	TASK-ME-ANYTHING-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.	201
B.8	random- <i>2D sticker image</i>	202
B.9	random- <i>3D tabletop scene</i> part 1	203

B.10	random-3D tabletop scene part 2	203
B.11	random-Real images with <i>Scene Graph</i>	204
B.12	random-3D tabletop scene	205
B.13	random-Real videos with <i>Scene Graph</i>	206
B.14	Statistics of evaluated tasks. For each task, we generate 15 task instances for evaluation. . .	208
B.15	The performance of Top-K query results approximation algorithms with different budgets. .	209
B.16	The performance of Threshold query results approximation algorithms with different budgets.	210
B.17	The performance of Model comparison query results approximation algorithms with different budgets.	210
B.18	The performance of Model debugging query results approximation algorithms with different budgets.	210
B.19	Models performance on TallyQA Counting benchmark and 2D how-many and 3D how-many in our TASK-ME-ANYTHING-RANDOM	220
C.1	Statistic of the number of templates each generator can create and the corresponding vocabulary size for each generator.	222
C.2	Example QA for Single-Image Generator.	223
C.3	Example QA for Multi-Image Generator.	223
C.4	Raw results of Figure 3.4a, Results of instruction tuning LLaVA-1.5-7B with DC-S , i.e. , single-image instruction data generated from DataComp images with our automatic scene graph generation pipeline.	224
C.5	Raw results of Figure 3.4b, Results of instruction tuning Mantis-SigLIP-8B with DC-M , i.e. , multi-image instruction data generated from DataComp images with our automatic scene graph generation pipeline.	224
D.1	Hyperparameters used for segmenter pretraining.	228
D.2	Evaluation of a scaled-up version of our segmenter on the ViPEntitySeg benchmark. This model is <i>not</i> used in any main-paper experiments; it serves only as an isolated study of segmentation quality under increased capacity.	229

D.3	Key hyperparameters used for training TrajVLM. Values reflect the shared configuration across all VLM experiments.	231
E.1	Model and training hyper-parameters, Molmo2-7B is a version of Molmo2 with the OLMo 2 language model.	234
E.2	Image benchmark results for a range of proprietary APIs, open-weight baselines, and our Molmo2 family across image understanding and counting benchmarks. The result of the best-performing open-weight model is in bold	235
E.3	Results on selective NLP benchmarks, including MMLU for general knowledge QA, GSM8K for math, ARC-C for reasoning, and MBPP+ for coding tasks.	235
E.4	Molmo2-VideoCount accuracy by query category.	236
E.5	Molmo2-VideoCount accuracy by object count.	237
E.6	Molmo2-VideoPoint full results.	238
E.7	Distribution of tracking dataset for Molmo2-VideoTrack (train) and Molmo2-Track (benchmark). We report the number of unique video clips, unique tracks, total queries, and average number of objects per query (Avg # Obj/Q) for each dataset. Type indicates video category; Ann. indicates original tracking annotation format (Segm: segmentation masks, Bbox: bounding boxes).	239
E.8	Pre-training ablations. Columns show the average of our 12 video benchmarks, using validation sets for EgoSchema, PerceptionText, and MLVU, video captioning F1, the average of the 11 image benchmarks using validation sets for InfoQA, DocQA, ChartQA, VQA v2, and AI2D, and the average score in PointBench.	239
E.9	Molmo2-4B with long context post-training. Post-training our model with a longer context length helps improve performance on benchmarks with long videos (>3 minutes).	239
E.10	Image Specialization. Scores on the 11 image benchmark average from. Molmo2 outperforms the specialized version showing positive video to image transfer.	240
E.11	Additional model ablation. Increasing pool size and removing time tokens hurts the model performance.	240

Chapter 1

Introduction

1.1 Background and Motivation

The past several years have witnessed remarkable progress in vision-language models (VLMs)—systems that jointly process visual and textual information. In this thesis, we use the term VLM broadly to encompass two prominent families of models that have shaped the field.

Contrastive vision-language models. One influential family of VLMs, exemplified by CLIP [Radford et al., 2021], learns a joint embedding space for images and text through contrastive pretraining on large-scale image-text pairs scraped from the web. By training a vision encoder and a text encoder to align matching image-caption pairs while pushing apart mismatched ones, CLIP and related models [Jia et al., 2021; Li et al., 2022; Tschannen et al., 2023] enabled powerful zero-shot image classification, image-text retrieval, and open-vocabulary recognition. These contrastive VLMs demonstrated that web-scale pretraining could produce visual representations with remarkable generalization, and they quickly became the foundation for a wide range of downstream applications.

Multimodal large language models. The now dominant family of VLMs takes the form of *multimodal large language models* (MLLMs): systems that connect a vision encoder to a large language model (LLM) through a learned connector module, enabling the model to process visual inputs alongside text in an autoregressive, conversational manner. Pioneered by models such as Flamingo [Alayrac et al., 2022], BLIP-2 [Li et al., 2023c], and LLaVA [Liu et al., 2024a], and further advanced by InstructBLIP [Dai et al., 2024],

Qwen-VL [Wang et al., 2024a], and proprietary systems like GPT-4V [OpenAI, 2023] and Gemini [Team, 2024a], these models can perform image captioning, visual question answering, open-ended multimodal dialogue, and complex reasoning tasks. Their capabilities are typically unlocked through a combination of large-scale vision-language pretraining and visual instruction tuning—a process in which models are fine-tuned on curated datasets of image-text instructions to follow diverse user queries grounded in visual content. MLLMs are the primary focus of this thesis.

Despite the impressive capabilities of both contrastive VLMs and MLLMs, strong aggregate performance on standard benchmarks does not necessarily imply that these models truly *understand* visual content. A contrastive model like CLIP may retrieve the correct caption for an image while being unable to distinguish it from a minimally altered hard negative that swaps two words [Yuksekgonul et al., 2023; Thrush et al., 2022; Zhao et al., 2022]. An MLLM asked “How many fingers are shown?” may confidently answer “five” even when the image clearly depicts six fingers, relying on the strong prior that hands have five fingers rather than actually counting what is visible. It may also generate fluent image descriptions that sound plausible but are unfaithful to what is actually depicted.

These failures point to a fundamental gap: current VLMs often achieve high performance through shallow image-text correlations rather than through *grounded visual understanding*—the ability to connect language to specific visual entities, their attributes, their relationships, and their dynamics over time. Closing this gap is essential for building reliable multimodal systems that can be trusted in real-world applications, from autonomous driving and robotics to medical imaging and assistive technologies.

1.2 Grounded Visual Understanding in Vision-Language Models

We define **grounded visual understanding** as a model’s ability to form precise, faithful connections between linguistic concepts and their corresponding visual evidence—going beyond surface-level image-text matching to establish fine-grained correspondences between what is said and what is seen.

Grounded visual understanding encompasses several key dimensions:

- **Entity grounding:** The ability to identify and localize specific objects or entities referenced in language within the visual input. For example, given “the red cup on the left,” a grounded model should attend

to the correct object rather than relying on generic scene statistics.

- **Attribute grounding:** The ability to bind visual attributes—such as color, shape, size, material, and texture—to the correct entities. A grounded model should distinguish “a large red ball” from “a small blue ball” by connecting each attribute to its referent.
- **Relational grounding:** The ability to understand spatial, semantic, and functional relationships between entities. This includes spatial relations (“the book is on top of the shelf”), actions (“the person is riding a bicycle”), and comparative relations (“the taller building”).
- **Compositional grounding:** The ability to understand how atoms—objects, attributes, and relations—compose to form complex meanings. Compositional grounding requires sensitivity to word order and structure: “a horse eating grass” must be distinguished from “grass eating a horse” [Ma et al., 2022; Yuksekgonul et al., 2023].
- **Temporal grounding:** The ability to ground language in the temporal dimension of video, identifying *when* events occur, tracking objects over time, and understanding the temporal ordering of actions [Krishna et al., 2017a].
- **Interactive grounding:** The ability to express grounding through spatial outputs such as points, bounding boxes, or tracks—enabling the model to “show” what it is referring to rather than merely describing it in language [Deitke et al., 2025; Yuan et al., 2024].

It is important to distinguish grounded visual understanding from *shallow image-text matching*. A model that relies on co-occurrence statistics (e.g., “beach” co-occurring with “sand” and “ocean”) may perform well on coarse retrieval tasks without ever grounding “sand” to the sandy region of the image. Grounded understanding, by contrast, requires the model to form explicit, verifiable connections between linguistic and visual content—connections that remain correct even when the visual scene is manipulated or the language is compositionally rearranged.

1.2.1 What Would Improved Grounding Enable?

Achieving reliable grounded visual understanding would unlock a wide range of capabilities and applications that current VLMs cannot support.

Scientific discovery and ecological research. Grounded VLMs could transform how scientists analyze visual data. Wildlife biologists who study animal behavior currently spend thousands of hours manually reviewing camera trap footage to identify species, count individuals, and annotate behaviors. A grounded model could automatically point to each animal in a frame, track individuals across a video sequence, and describe their interactions—“the juvenile elephant approaches the water hole while two adults stand guard nearby.” Similarly, marine biologists studying coral reef health could use grounded models to localize and track individual coral colonies across time-lapse imagery, identifying regions of bleaching or recovery. In astronomy, grounded understanding could help identify and track transient events in telescope surveys. The common thread is that scientific imaging produces vast quantities of visual data where the bottleneck is not collection but *precise, spatially grounded analysis*—exactly the capability that improved grounding would provide.

Medical imaging and clinical decision support. In radiology, a grounded VLM could point to the specific region of a scan that supports its finding, rather than simply generating a text report. A clinician could ask “Where is the lesion?” and receive a spatially precise response overlaid on the image. This would make AI-assisted diagnosis more transparent and trustworthy, allowing doctors to verify the model’s reasoning rather than treating it as a black box. Grounded models could also track disease progression across longitudinal scans by localizing and comparing specific anatomical structures over time.

Embodied agents and robotics. Grounded understanding is a prerequisite for embodied AI systems that must act on their perceptions. A robot asked to “pick up the red cup next to the laptop” must ground “red cup” and “next to the laptop” to specific objects in its visual field. Without reliable entity and relational grounding, such instructions become ambiguous or dangerous. A household robot that can track objects as they are moved, remember where items were placed, and reason about spatial relationships would be far more useful than one that merely recognizes object categories. Improved grounding would enable more robust robotic

manipulation, navigation, and human-robot interaction in unstructured environments.

Video understanding at scale. Many real-world applications require understanding long-form video: surveillance systems that must locate specific incidents across days of footage, sports analysts who need to find particular plays and track player movements, educators reviewing classroom recordings to identify moments of student engagement, or meeting tools that retrieve specific discussion points along with the visual context in which they occurred. Grounded models that can point to and track objects and events across time would transform these applications from keyword-based search to spatially and temporally precise retrieval.

Accessibility and assistive technology. For visually impaired users, current image description systems provide generic captions that often omit critical spatial details. A grounded VLM could provide descriptions that are spatially faithful—“the door is on your left, about three steps ahead, with the handle at waist height”—enabling more independent navigation and interaction with the physical world. In educational settings, grounded models could describe diagrams, charts, and figures with precise reference to their spatial layout, making STEM content more accessible.

Verifiable and interpretable AI. Grounding provides a pathway to *verifiable* multimodal AI. When a model can point to the visual evidence supporting its answer, users and auditors can verify its reasoning. This is fundamentally different from a model that produces a correct answer through opaque internal mechanisms. Grounded outputs—such as spatial points, object tracks, or highlighted regions—make model behavior interpretable and auditable, which is essential for deploying VLMs in high-stakes domains such as autonomous driving, legal analysis of video evidence, and content moderation.

1.3 Challenges in Achieving Grounded Visual Understanding

Despite the importance of grounded visual understanding, achieving it in practice faces several interconnected challenges.

1.3.1 Inadequate Evaluation of Grounding

Most widely used benchmarks for vision-language models evaluate coarse capabilities—overall captioning quality, aggregate VQA accuracy, or holistic retrieval performance [Yue et al., 2024; Liu et al., 2023c; Fu et al., 2024a]. These benchmarks often fail to probe whether models achieve their scores through genuine grounding or through shortcut exploitation. Worse, some compositionality benchmarks that specifically aim to test grounding contain systematic biases: the hard negatives they employ are distinguishable from correct captions through surface-level textual cues alone, allowing blind models that never see the image to outperform vision-language models [Hsieh et al., 2024]. Without reliable evaluation tools, it is difficult to measure progress toward grounded understanding or to identify the specific dimensions where models fall short.

1.3.2 Weak or Indirect Supervision for Grounding

The training data used for vision-language models typically provides only sparse supervision over visual content. Web-crawled image-text pairs offer broad coverage but shallow alignment—a caption like “beautiful sunset at the beach” provides no supervision over specific objects, their locations, attributes, or relationships [Gadre et al., 2024; Sharma et al., 2018]. Even curated visual instruction datasets, while richer, tend to annotate only a small fraction of each image’s visual content, leaving most objects, attributes, and relations unsupervised [Liu et al., 2024a; Dai et al., 2024]. This sparsity means that models are rarely trained to attend exhaustively to the visual world, contributing to their tendency to produce confident but ungrounded predictions.

1.3.3 Visual Representations Are Not Always Grounding-Friendly

The dominant approach to visual representation in modern VLMs divides images and videos into fixed-size patches processed by a vision transformer [Dosovitskiy et al., 2021]. This uniform spatial tokenization is agnostic to the semantic structure of the scene: it fragments coherent objects across multiple tokens, groups semantically unrelated pixels into single tokens, and treats all spatial regions equally regardless of their importance [Zheng et al., 2025]. Such representations place the burden on downstream transformer layers to reassemble these fragmented patches into coherent perceptual units—a task that is both computationally

expensive and often imperfect. More grounding-friendly representations would organize visual tokens around the natural structure of the scene: objects, their parts, and their trajectories over time.

1.3.4 Temporal and Dynamic Grounding Is Harder than Static Grounding

Extending grounded understanding from images to video introduces additional challenges. Videos contain temporal dynamics—objects move, interactions unfold, and scenes change over time. Patch-based tokenization applied to video produces token counts that scale linearly with frame count rather than with semantic complexity, creating severe efficiency bottlenecks for long videos [Arnab et al., 2021]. Moreover, maintaining consistent grounding of an entity across time—tracking the same object as it moves, is occluded, and reappears—requires representations that are robust to appearance changes and camera motion [Ravi et al., 2025].

1.3.5 Grounded Multimodal Reasoning Remains an Open Challenge

Even when individual grounding capabilities are present, integrating them into a coherent multimodal reasoning system remains difficult. A truly grounded VLM should not only understand what is in a scene but also be able to *point* to specific objects, *track* them over time, and *reason* about spatial and temporal relationships in a way that is verifiable and precise. Current open-source models lag behind proprietary systems in these grounding capabilities, and even the best proprietary models struggle with complex grounding tasks such as counting objects by pointing or tracking objects through occlusions [Ahmad et al., 2025; Munasinghe et al., 2025].

1.4 Thesis Statement and Overview

Thesis statement. This thesis argues that grounded visual understanding—the ability to form precise, faithful connections between language and visual content—is the key bottleneck limiting the reliability and capability of current vision-language models. We demonstrate that progress on grounded understanding requires advances along four complementary dimensions: *evaluation*, to reliably measure grounding capabilities; *supervision*, to provide richer training signals that encourage grounding; *representation*, to align

visual tokens with the semantic structure of scenes; and *reasoning*, to integrate grounding into a complete multimodal system.

Research agenda. We pursue these four dimensions through four interconnected research contributions, each forming a chapter of this thesis:

1. **Evaluation** (Chapter 2): We develop reliable diagnostic tools for measuring grounded visual understanding, addressing both the trustworthiness of existing benchmarks and the need for fine-grained, task-specific evaluation.
2. **Supervision** (Chapter 3): We introduce methods for generating exhaustive visual instruction data that provides dense supervision over objects, attributes, and relations, using scene graphs as a structured intermediary.
3. **Representation** (Chapter 4): We propose grounded visual tokenization—organizing visual tokens around object trajectories rather than fixed spatial patches—to build representations that are inherently more aligned with scene structure.
4. **Reasoning** (Chapter 5): We present a fully open vision-language model with first-class grounding capabilities, demonstrating that grounded understanding enables advanced multimodal reasoning including spatial pointing, temporal tracking, and grounded chain-of-thought.

Together, these contributions trace a path from *measuring* grounded visual understanding to *improving* it through data and representations, and finally to *deploying* it in a complete reasoning system.

1.5 Chapter Contributions

1.5.1 Evaluating Grounded Visual Understanding

Chapter 2 addresses the evaluation dimension through two complementary works.

SugarCrepe [Hsieh et al., 2024] reveals that widely used compositionality benchmarks for vision-language models contain systematic biases that inflate performance estimates. We show that a text-only model that never sees the image can outperform state-of-the-art VLMs on these benchmarks, because the

hard negative captions are distinguishable from real captions through surface-level cues alone. We introduce SUGARCREPE, a new benchmark that leverages large language models and adversarial refinement to produce hard negatives free of these artifacts. Re-evaluation on SUGARCREPE reveals that previously reported improvements from compositional data augmentation are substantially overestimated.

Task-Me-Anything [Zhang et al., 2024c] is a programmatic benchmark generation engine that enables fine-grained, query-driven evaluation of VLM capabilities. It maintains an extensible taxonomy of visual concepts with over 750 million possible task instances spanning image and video modalities. Users can pose queries such as “Which model is best at recognizing object shapes?” or “What are the top-10 tasks that GPT-4o struggles with?” and receive targeted evaluation results within a computational budget. Our analyses reveal that open-source VLMs excel at object and attribute recognition but lack spatial, temporal, and counting capabilities essential for grounded understanding.

1.5.2 Improving Grounded Visual Understanding with Rich Visual Instructions

Chapter 3 addresses the supervision dimension.

ProVision [Zhang et al., 2024d] presents a system for programmatically generating exhaustive vision-language instruction data using scene graphs as a structured intermediary between pixels and language. We implement 24 single-image and 14 multi-image instruction data generators that cover objects, attributes, relations, segmentation, and depth. To scale beyond manually annotated scene graphs, we also develop an automatic scene graph generation pipeline using state-of-the-art models for object detection [Cheng et al., 2024b], segmentation [Ravi et al., 2025], and depth estimation [Yang et al., 2024]. The resulting PROVISION-10M dataset contains over 10 million instruction examples. Experiments show that this data yields significant improvements—at least 7% on CVBench and 8 percentage points on Mantis-Eval—and proves effective for both pre-training and instruction tuning.

1.5.3 Realizing Grounded Visual Understanding through Grounded Visual Tokenization

Chapter 4 addresses the representation dimension through two systems that progressively realize grounded visual tokenization.

TrajViT [Zheng et al., 2025] introduces a new paradigm of grounded visual tokenization, replacing

the standard practice of dividing videos into fixed space-time patches with tokenization organized around panoptic sub-object trajectories. By extracting trajectories using an external segmentation-and-tracking pipeline and encoding each trajectory as a fixed-size token, TrajViT reduces token counts by $10\times$ while improving performance across retrieval, classification, and video question answering benchmarks. The approach is robust to camera motion and lighting changes, and scales efficiently to long videos.

TrajTok [Zheng et al., 2026] makes this process fully end-to-end and differentiable, replacing the external pipeline with a lightweight learned segmenter that adapts its tokenization granularity to downstream objectives. TrajTok achieves improvements of +4.8% on Kinetics-400 and +4.1% on Something-Something V2 over standard patch-based tokenization, exhibits stronger scaling behavior with increasing data, and proves versatile as both a feature adapter for pretrained encoders and a connector in vision-language models.

1.5.4 Unlocking Grounded Multimodal Reasoning

Chapter 5 addresses the reasoning dimension.

Molmo 2 [Clark et al., 2025] is a family of fully open vision-language models that make grounding a first-class capability. Supporting single images, multi-image contexts, and video, Molmo 2 can not only describe visual content but also *point* to specific objects, *track* them across video frames, and *reason* with spatial and temporal precision. A core contribution is a suite of 9 novel datasets—including dense video captions, open-vocabulary video pointing and tracking annotations, and long-form QA data—all collected without distilling from proprietary models. Training innovations include token weighting for balancing diverse tasks, sequence packing, and message-tree scheduling for training efficiency. Molmo 2 achieves state-of-the-art results on video grounding benchmarks while remaining competitive on standard video understanding tasks, demonstrating that grounded visual understanding is the foundation for advanced multimodal reasoning.

Chapter 2

Evaluating Grounded Visual Understanding

2.1 Introduction

A central premise of this thesis is that vision-language models must achieve *grounded* visual understanding—connecting language to actual visual content rather than exploiting superficial statistical patterns. Before we can improve grounding, however, we must first be able to measure it reliably. This chapter addresses the evaluation challenge by asking two complementary questions: *Are our current benchmarks trustworthy?* and *Can we evaluate models at a finer granularity than existing benchmarks allow?*

Compositionality—the ability to understand how atoms such as objects, attributes, and relations combine to form meaning—is a cornerstone of visual understanding [Cresswell, 1973; Janssen and Partee, 1997; Hupkes et al., 2020]. A wave of recent benchmarks, including Winoground [Thrush et al., 2022], ARO [Yuksekgonul et al., 2023], and CREPE [Ma et al., 2022], have attempted to measure whether vision-language models exhibit this capability by testing whether they can distinguish a correct image–caption pair from a “hard negative” in which words are swapped, replaced, or shuffled. In Section 2.3, we reveal a critical flaw shared by these benchmarks: a *blind* model that never sees the image can outperform state-of-the-art vision-language models, because the hard negatives are systematically distinguishable from real captions through surface-level textual cues alone. We trace this vulnerability to two biases—implausibility and disfluency of the generated negatives—and introduce SUGARCREPE, a benchmark that leverages large language models and adversarial refinement to produce hard negatives free of these artifacts. Re-evaluation

on SUGARCREPE shows that previously reported improvements from compositional data augmentation are substantially overestimated, underscoring the need for more reliable diagnostic tools.

Reliable benchmarks, however, are only part of the picture. Even a perfect compositionality benchmark probes a narrow slice of visual understanding; practitioners also need to diagnose model capabilities across a much broader space of perception tasks—counting, spatial reasoning, attribute recognition, temporal understanding, and more—at a granularity that static benchmarks cannot provide. Starting from Section 2.5, we present TASK-ME-ANYTHING, a programmatic benchmark generation engine that addresses this gap. Given a user query such as “Which model is best at recognising shapes of rotating objects?”, TASK-ME-ANYTHING selects the relevant portion of a structured taxonomy of visual concepts, instantiates task-specific generators to produce evaluation instances, and returns fine-grained performance profiles. Its generation space spans over 750 million possible task instances across image and video modalities, and it supports efficient on-budget approximation so that even large-scale queries remain tractable.

Together, SUGARCREPE and TASK-ME-ANYTHING establish the diagnostic framework for this thesis: the former ensures that benchmark measurements are not inflated by dataset artifacts, while the latter enables systematic, query-driven identification of *where* and *how* current models fall short. The subsequent chapters build on these findings to close the identified gaps—through richer training supervision (Chapter 3), more aligned visual representations (Chapter 4), and a fully grounded multimodal reasoning system (Chapter 5).

2.2 Related Work

This chapter draws on three interrelated lines of research: evaluating and debiasing compositionality benchmarks for vision-language models, and programmatic benchmark generation for broad-coverage model evaluation.

Evaluating vision-language compositionality. Recent benchmarks have uncovered that vision-language models exhibit surprisingly weak compositional understanding despite strong aggregate performance on downstream tasks [Radford et al., 2021; Li et al., 2022; Singh et al., 2022; Alayrac et al., 2022]. Models struggle to distinguish texts containing the same words in different orders [Thrush et al., 2022], to bind objects to their attributes, and to understand inter-object relationships [Zhao et al., 2022; Yuksekgonul

et al., 2023; Ray et al., 2023]. To improve these capabilities, several methods introduce compositional hard negatives during training [Yuksekgonul et al., 2023; Doveh et al., 2023; Cascante-Bonilla et al., 2023; Singh et al., 2023]. However, as we show in this chapter, the benchmarks used to evaluate these methods contain exploitable biases, making it unclear whether reported gains reflect genuine compositional understanding or mere artifact exploitation.

Debiasing datasets for faithful evaluation. Biased evaluation datasets can substantially overestimate model capabilities [Gururangan et al., 2018]. To address this, adversarial filtering methods iteratively train classifiers to identify and remove “easy” instances whose labels can be predicted from surface cues alone [Zellers et al., 2018; Sakaguchi et al., 2021; Le Bras et al., 2020; Reif and Schwartz, 2023]. We build on these ideas—in particular the lightweight AFLite procedure [Sakaguchi et al., 2021; Le Bras et al., 2020]—to construct SUGARCREPE, a compositionality benchmark in which the identified textual biases are maximally suppressed.

Programmatic and large-scale benchmark generation. Static benchmarks, however carefully curated, cover a fixed set of skills. An alternative paradigm generates evaluation instances *programmatically* from structured scene representations. GQA [Hudson and Manning, 2019] and AGQA [Grunde-McLaughlin et al., 2021; Gandhi et al., 2022] pioneered this approach for visual question answering using scene graphs [Krishna et al., 2017b; Ji et al., 2020]. More recently, efforts have extended programmatic generation to broader model evaluations, but most focus on a single modality or a narrow task family and do not support user-driven, query-centric evaluation. TASK-ME-ANYTHING generalises this paradigm: it combines a spatio-temporal scene-graph taxonomy with pluggable task generators and on-budget approximation methods, enabling fine-grained, on-demand evaluation of multimodal language models across hundreds of perception dimensions.

2.3 SugarCrepe: Fixing Hackable Compositionality Benchmarks

2.3.1 Biases in Current Benchmarks

A majority of existing compositionality benchmarks for vision-language models formulate the evaluation task as image-to-text retrieval [Zhao et al., 2022; Yuksekgonul et al., 2023; Ma et al., 2022]. We focus on

Dataset	Nonsensical Hard Negatives	Non-fluent Hard Negatives
CREPE	Olives and grape inside a plate. Ground in a basket on the flowers. A hair wearing a necklace, with her lady on a table.	A door with panes not in a room; the door has windows. Right has word another word. There is a words. A shelf with books in something. There is no background.
ARO	The grass is eating the horse. A gray bathtub is looking at a white cat. Green ball with a remarkable chair behind a blue scene.	At brown cat a in looking a gray dog sitting is and white bathtub. Scene with remarkable a ball blue a green behind chair. Books the looking at people are.
VL-CheckList	Sheep is hardwood. Empty zebras. The bush speaking in the garden.	An man fishing a food from a wrapper using a paw at a open. It heaving at a city. An grouping subduing at a room access.

Table 2.1: Existing compositionality benchmarks rely on procedurally-generated hard negatives which often do not make logical sense or are not fluent due to grammatical errors.

these benchmarks and discuss others [Thrush et al., 2022; Ray et al., 2023] in Appendix A.2. Given an image, the model is probed to select text that correctly describes the image from a pool of candidates. Unlike standard retrieval tasks where the negative (incorrect) candidates differ a lot from the *positive* (correct) text, compositionality benchmarks intentionally design *hard negative* texts that differ minimally from the positive text, in order to test whether the model understands the fine-grained atomic concepts that compose the scene.

Existing hard negative generation process introduces undesirable biases. Existing benchmarks generate hard negative texts through rule-based programmatic procedures [Zhao et al., 2022; Yuksekgonul et al., 2023; Ma et al., 2022], which produce hard negatives by replacing a word of specific type (an object, attribute, or relation) in the original text, by swapping two words, or by shuffling the word order. We find that such procedures introduce unintentional biases in the generated hard negatives (see Table 2.1); specifically, we observe two major types of undesirable artifacts: (1) *nonsensical* artifacts, and (2) *non-fluent* artifacts. In order to quantitatively measure these biases, we utilize Vera [Liu et al., 2023a], a plausibility estimation model, to characterize the nonsensical bias. Specifically, we define $\text{Vera}(T)$ to be the plausibility score of a caption T , where a higher score suggests more sensical the caption is. Similarly, to capture the non-fluent bias, we leverage a grammar-check model [Morris et al., 2020] that assigns high scores, $\text{Grammar}(T)$, to more grammatically correct texts. In Figure 2.1, we find that Vera and the grammar model assign higher scores to positive texts, suggesting that many hard negatives are nonsensical and not fluent.

Dataset biases render current compositionality benchmarks ineffective. Given the heavily-skewed score gaps, we show that blind models (i.e., Vera and the grammar model) that simply select the higher-scoring texts as positives and admittedly do not possess any vision-language compositionality, can achieve

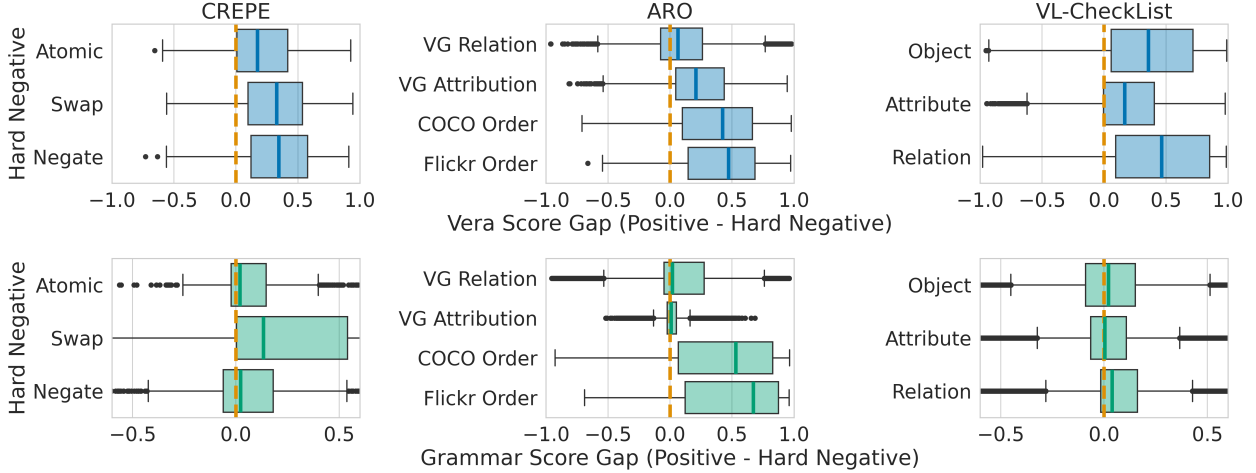


Figure 2.1: Top row: We define *Vera score gap* as the score difference between the positive and hard negative texts: $Vera(T^p) - Vera(T^n)$. The entire Vera score gap distribution lies on the positive spectrum, indicating that the template-generated hard negative texts usually have low plausibility. Bottom row: Similarly, *Grammar score gap* is defined by: $Grammar(T^p) - Grammar(T^n)$. On grammar score, we also find that the distribution largely rests on the positive side, suggesting that most hard negative texts in existing benchmarks exhibit grammatical errors.

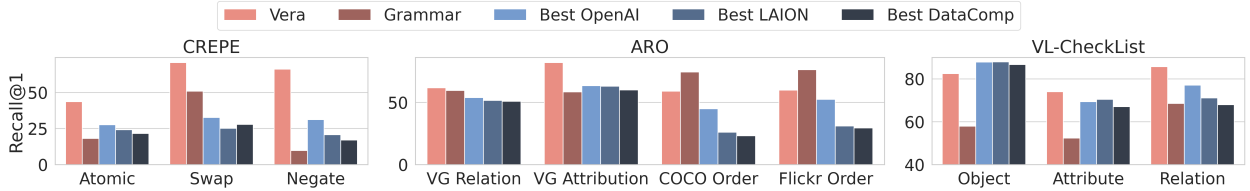


Figure 2.2: Blind commonsense Vera model and Grammar model outperform state-of-the-art CLIP models on nearly *all* existing benchmarks by exploiting the nonsensical and non-fluent artifacts. This suggests that existing benchmarks are hackable and ineffective in measuring compositionality.

state-of-the-art performances on existing benchmarks. We compare the the blind models against 17 pre-trained CLIP models from three sources: OpenAI’s in-house WebImageText dataset [Radford et al., 2021], LAION [Schuhmann et al., 2022], and Datacomp [Gadre et al., 2024]. We plot the performances of the blind models and the best-performing CLIP models from each category (Figure 2.2). Blind models achieves state-of-the-art performances on 9 out of 10 existing benchmark tasks. We provide full evaluation results in Appendix A.4.1.

We introduce SUGARCREPE, a new benchmark for faithful evaluation of vision-language models’ compositionality based on the image-text pairs of COCO [Lin et al., 2014]. SUGARCREPE presents two key contributions over existing benchmarks: (1) it drastically reduces the two identified dataset biases (Sec. 2.3.1),

and (2) it covers a broad range of fine-grained types of hard negatives (Sec. 2.3.1). We present a summary comparison on compositionality benchmarks in Appendix A.2.

SUGARCREPE generation workflow alleviates dataset biases. The generation procedure of SUGARCREPE consists of three main stages, centered around creating sensical and fluent hard negatives that close the distributional gaps to the positive texts, and ensuring a balanced distribution on the score gaps to make the final dataset robust to the identified biases.

Stage 1: Generate sensical and fluent hard negatives with a large language model. Observing the capability of modern large language models in generating fluent and plausible texts, we leverage ChatGPT [OpenAI, 2022] to generate hard negative texts where we explicitly instruct it to avoid commonsense (logical) and fluency (grammatical) errors. To guide ChatGPT in re-writing a given positive text into its hard negative counterparts, we provide few-shot demonstrations written by the authors and leverage its in-context learning ability [Brown et al., 2020] to generalize to unseen texts. Figure 2.3 shows an example demonstration used and an actual hard negative generated. We detail all the prompt templates in Appendix A.3.2. Table 2.3 shows the comparisons between hard negatives generated from ChatGPT in SUGARCREPE and that from existing benchmarks.

Stage 2: Filter false negatives with human validation. A generated text is considered a valid hard negative only if it incorrectly describes the corresponding image. For example, given an image with a positive caption “a man and a child sitting on a sofa”, a compositional change that replaces “child” with “girl” may still result in a correct caption. To ensure the validity of the hard negatives in SUGARCREPE, we filter out *false* negatives by manually examining the generated hard negatives and their corresponding images.

Stage 3: De-bias dataset with adversarial refinement. While ChatGPT yields more sensical and fluent text, there is no guarantee that the bias between positive and negative texts is negligible. Following dataset de-biasing work [Zellers et al., 2018; Sakaguchi et al., 2021; Le Bras et al., 2020], we develop an adversarial refinement mechanism that maximally reduces the undesirably exploitable artifacts in SUGARCREPE. Specifically, our goal is to ensure that performance improvements on SUGARCREPE cannot be achieved by exploiting the identified nonsensical and non-fluent biases. To accomplish this, we characterize the biases again with the commonsense and grammar models [Liu et al., 2023a; Morris et al., 2020], and subsample the dataset to ensure symmetric score gap distributions on both the positive and negative sides, as shown in

```

Given an input sentence describing a scene, your task
is to:
1. Locate the noun words in the sentence.
2. Randomly pick one noun word.
3. Replace the selected noun word with a new noun word
to make a new sentence.

The new sentence must meet the following three
requirements:
1. The new sentence must be describing a scene that is
as different as possible from the original scene.
2. The new sentence must be fluent and grammatically
correct.
3. The new sentence must make logical sense.

Here are some examples:

Original sentence: A man is in a kitchen making pizzas.
Nouns: ["man", "kitchen", "pizzas"]
Selected noun: man
New noun: woman
New sentence: A woman is in a kitchen making pizzas.

Original sentence: a woman seated on wall and birds
besides her
Nouns: ['woman', 'wall', 'birds']
Selected noun: wall
New noun: bench
New sentence: A woman seated on a bench and birds
besides her.

```

Figure 2.3: Example prompt (black) and actual hard negative (green) generated from ChatGPT.

Algorithm 1 Adversarial Refinement

Require: Text-only model M_1 and M_2 ; Number of grids K ; A set of candidates $\mathcal{D} = \{I_i, T_i^p, T_i^n\}_{i \in [N]}$, where I_i , T_i^p , and T_i^n are i -th image, positive caption, and negative caption.

Ensure: A subset $\bar{\mathcal{D}} \subset \mathcal{D}$

- 1: Calculate the model score gap for each candidate $g_i^{(1)} = M_1(T_i^p) - M_1(T_i^n)$ and $g_i^{(2)} = M_2(T_i^p) - M_2(T_i^n)$
 - 2: Split the 2D space $[-1, 1] \times [-1, 1]$ to $K \times K$ equal-size grids.
 - 3: Place each candidate to a grid based on the score gaps $g_i^{(1)}$ and $g_i^{(2)}$.
 - 4: Initialize $\bar{\mathcal{D}} = \{\}$
 - 5: **for** each pair of grid (G_j, G_j^*) symmetric about the original point $(0, 0)$ **do**
 - 6: **if** $|G_j| > |G_j^*|$ **then**
 - 7: Sample $|G_j^*|$ candidates from G_j and put them to $\bar{\mathcal{D}}$.
 - 8: Put candidates in G_j^* to $\bar{\mathcal{D}}$.
 - 9: **else**
 - 10: Sample $|G_j|$ candidates from G_j^* and put them to $\bar{\mathcal{D}}$.
 - 11: Put candidates in G_j to $\bar{\mathcal{D}}$.
-

Figure 2.4. We note the symmetry around zero implies that the commonsense and grammar scores can no longer be used to infer the ground truth positive texts. We provide the adversarial refinement algorithm in Algorithm 1.

SUGARCREPE covers a broad range of hard negative types. To test different aspects of vision-language models’ compositional understanding, we follow CREPE [Ma et al., 2022] to consider various *forms* of hard negatives, and follow VL-CheckList [Zhao et al., 2022] and ARO [Yuksekgonul et al., 2023] to consider different fine-grained *categories* of the atomic concepts. In total, SUGARCREPE covers 7 fine-grained types of hard negatives, as shown in Table 2.2. We introduce the dataset taxonomy below, starting from the *form* of the hard negatives to its different *finer-grained* variants.

The REPLACE form. Given a positive text describing a scene, we generate a REPLACE hard negative by replacing an atomic concept in the original text with a new concept that makes the text mismatch with the original scene. Based on the type of the atomic concept—object, attribute, or relation—we further categorize REPLACE hard negatives into REPLACE-OBJ, REPLACE-ATT, and REPLACE-REL.

The SWAP form. Different from REPLACE, SWAP does not introduce new concepts in the hard negatives, but a SWAP hard negative is generated by swapping two atomic concepts of the same category in the positive text. We further categorize SWAP into SWAP-OBJ and SWAP-ATT, and omit swapping two relationships since it generally results in nonsensical texts.

The ADD form. Similar to the REPLACE form, but instead of replacing an atomic concept with a new one, we generate an ADD hard negative by adding a new atomic concept to the positive text that makes it mismatch with the original scene. We only further categorize ADD into ADD-OBJ (adding object concept) and ADD-ATT (adding attribute concept), as adding new relationship concepts to the positive texts often make them highly implausible.

Dataset overview. The final evaluation set of SUGARCREPE consists of 7,512 examples, where the numbers for each fine-grained type are listed in Table 2.2. Each example is an image-to-text retrieval task composed of an image, a positive text, and a hard negative. On SUGARCREPE, random chance performance has an average accuracy of 50%. We note that ARO and CREPE additionally consider SHUFFLE (randomly shuffling words in a sentence) and NEGATE (adding negation keywords “no/not” to a sentence) hard negatives. We however omit them in SUGARCREPE as SHUFFLE is very unlikely to be plausible and fluent, and NEGATE

	REPLACE			SWAP		ADD	
	Object	Attribute	Relation	Object	Attribute	Object	Attribute
# negative captions	1,652	788	1,406	246	666	2,062	692

Table 2.2: We report the number of hard negative captions of all types in SUGARCREPE.

introduces irreducible keyword artifacts [Ma et al., 2022].¹

2.4 SugarCrepe: Evaluations

In this section, we qualitatively and quantitatively compare SUGARCREPE to existing benchmarks (Sec. 2.4.1), re-evaluate recent methods proposed to improve compositionality of vision-language models (Sec. 2.4.2), and comprehensively evaluate a wide array of pretrained CLIP models (Sec. 2.4.2).

To systematically and fairly compare SUGARCREPE with existing benchmarks, we normalize the benchmarks by reproducing their data generation workflow using COCO [Lin et al., 2014] as in SUGARCREPE. We utilize source code from CREPE [Ma et al., 2022] to generate REPLACE, SWAP, NEGATE hard negatives and take SHUFFLE hard negatives released in ARO [Yuksekgonul et al., 2023]. We refer to this reproduced dataset as ARO+CREPE. In addition, we standardize the evaluation task as retrieving the correct caption from *two* possible choices, i.e. , a positive text and a hard negative. This normalization sets the positive texts fixed for all benchmarks, including SUGARCREPE.

2.4.1 Reducing Dataset Biases

SUGARCREPE generates more sensical and fluent hard negatives. We validate that SUGARCREPE generates higher quality hard negative texts by leveraging ChatGPT than previous rule-based approaches. Qualitatively, in Table 2.3, we observe that the hard negatives in SUGARCREPE are more sensical and fluent compared to hard negatives in ARO+CREPE. We report human evaluation results in Appendix A.4.2 that show on an average of 35% of examples, hard negatives in SUGARCREPE have *strictly* higher quality than ARO+CREPE in terms of commonsense and fluency. For instance, on SWAP, humans judge that SUGARCREPE wins 68% over ARO+CREPE and ties on 28% of examples in terms of commonsense. Quantitatively, in Table 2.4, we compare the commonsense and grammar scores averaged over the hard negative

¹One can easily infer hard negatives from whether the text contains negation keywords “no/not”.

Hard-Negative Type	Text Type	Commonsense	Fluency
REPLACE	Original	Two adult bears play fight in the water.	A man sitting in front of a laptop computer.
	ARO+CREPE	Two adult bears play fight in the soda.	A man sitting around front of a laptop computer.
	SUGARCREPE	A flock of ducks play fight in the water.	A man standing in front of a laptop computer.
SWAP	Original	A woman standing behind a fence looking at an elephant.	Man swinging tennis racket while group of people watches.
	ARO+CREPE	A fence standing behind a woman looking at an elephant.	Group swinging tennis racket while man of people watches.
	SUGARCREPE	An elephant standing behind a fence looking at a woman.	Group of people swinging tennis racket while man watches.
NEGATE / ADD	Original	A teddy bear next to a stuffed fish.	A red fire hydrant on a city sidewalk.
	ARO+CREPE	A teddy bear next to a stuffed fish. There is no teddy bear.	A red fire not hydrant on a city sidewalk.
	SUGARCREPE	A teddy bear and a stuffed fish and a robot toy.	A red fire hydrant and a trash can on a city sidewalk.

Table 2.3: We present example positive texts and their hard negatives in ARO+CREPE (generated using existing procedures) and SUGARCREPE (generated with ChatGPT). SUGARCREPE brings significant improvements in commonsense and fluency.

Hard-negative Type	Metric	Average Score		Pairwise Better Ratio
		ARO+CREPE	SUGARCREPE	
REPLACE	Commonsense	37.46	50.21	77.71
	Grammar	76.79	88.96	86.85
SWAP	Commonsense	23.09	41.57	78.76
	Grammar	45.67	80.46	87.02
NEGATE / ADD	Commonsense	25.24	50.20	87.24
	Grammar	65.09	90.07	95.03

Table 2.4: We compare the commonsense and grammar scores on hard negatives in ARO+CREPE and SUGARCREPE. We report both their respective average scores and the ratio where SUGARCREPE has higher score than ARO+CREPE in pairwise comparison. Overall, SUGARCREPE has hard negatives with better commonsense and grammar.

texts in both ARO+CREPE and SUGARCREPE. We see SUGARCREPE has much higher average scores than ARO+CREPE. Additionally, pairwise comparisons show that SUGARCREPE has higher commonsense and grammar scores than ARO+CREPE on 86% of examples on average.

SUGARCREPE disentangles the identified exploitable biases. We show that the final SUGARCREPE evaluation set maximally reduces the identified biases that could be exploited undesirably to achieve improvements on a benchmark. Figure 2.4 visualizes the Vera/Grammar score gap distributions. We compare the distributions between ARO+CREPE and SUGARCREPE (before and after adversarial refinement). First, We see that by leveraging ChatGPT, the hard negative texts in SUGARCREPE already have lower biases than ARO+CREPE before adversarial refinement, i.e. , the score gap distribution is more centered around zero. Furthermore, we see that after adversarial refinement, the score gap distributions on the final SUGARCREPE evaluation set are symmetric around zero. This implies that the previously identified artifacts can no longer be exploited to infer the positive texts. As a result, we show that the previous commonsense and grammar attacks

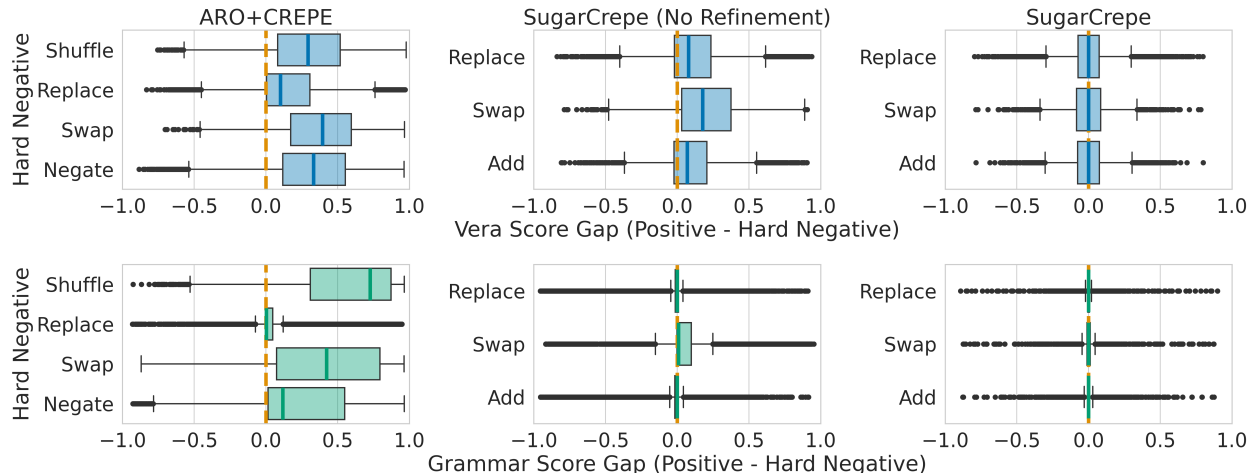


Figure 2.4: We compare the Vera (top row) and Grammar (bottom row) score gap distributions between ARO+CREPE (leftmost column), SUGARCREPE without adversarial refinement (middle), and SUGARCREPE (rightmost). Top row: We see that Vera score gap distribution shifts from the positive spectrum to more centered around zero from ARO+CREPE to SUGARCREPE without refinement. After adversarial refinement, we ensure the score gap distribution is centered around zero on SUGARCREPE. Bottom row: Similarly, from ARO+CREPE to SUGARCREPE, we see the Grammar score gap distribution shifts from the positive spectrum to centered around zero.

that are extremely successful on existing benchmarks do not work on SUGARCREPE. As shown in Table 2.6, these blind models now consistently rank the *last* on SUGARCREPE as compared to other pretrained CLIP models.

2.4.2 Re-evaluating VL Compositionality

Given the vulnerability of existing compositionality benchmarks, it is unclear whether recently proposed methods that show state-of-the-art performances on these benchmarks are indeed effective. Thus, we re-evaluate these methods with SUGARCREPE.

Hard negative augmented training. Specifically, we focus on evaluating one common *data-augmentation* approach considered in [Yuksekgonul et al., 2023; Doveh et al., 2023], where the core idea is to explicitly create hard negatives and train the model to distinguish them. We broadly refer to this training scheme as NEGCLIP following [Yuksekgonul et al., 2023]. We evaluate two NEGCLIP training schemes: finetuning and training from scratch. For finetuning, in addition to taking the model released in [Yuksekgonul et al., 2023], we finetune another three NEGCLIP models (using ViT-B/32 following [Yuksekgonul et al., 2023])

with three respective types of hard negatives (i.e. , REPLACE, SWAP, NEGATE) generated using CREPE’s [Ma et al., 2022] source code. For training from scratch, we use RN50 as the base model and train variants of NEGCLIP by augmenting the training examples with different types of hard negatives. We perform both training and finetuning on COCO [Lin et al., 2014].

Improvements are overestimated due to unintentionally overfitting. In Table 2.5, we first see that NEGCLIP finetuned models show significant improvements on ARO+CREPE, boosting the performance more than 10% compared to standard CLIP finetuning on 11 out of 16 cases (highlighted in green). The lifts are especially large when the hard negative type used in finetuning matches that used in evaluation, where NEGCLIP finetuned models can achieve near human-level performances. For instance, by finetuning with REPLACE hard negatives, NEGCLIP reaches 94% on ARO+CREPE evaluated with REPLACE hard negatives (human performance is 95%). While the results on ARO+CREPE suggest that NEGCLIP is seemingly sufficient in equipping models with strong compositionality, we however see that the improvements brought by NEGCLIP are much smaller on SUGARCREPE. In fact, none of the improvements on SUGARCREPE is larger than 10%, and the best performing NEGCLIP finetuned models still have large gaps to human-level performances, e.g. , best NEGCLIP model lags behind human by 23% on SUGARCREPE’s SWAP hard negatives. Similarly, when trained from scratch, we observe the same trend that NEGCLIP’s improvements are much larger on ARO+CREPE than on SUGARCREPE. The improvements on ARO+CREPE are again most pronounced when the training and testing hard negative type matches.

We attribute the stark contrast in NEGCLIP’s effectiveness on ARO+CREPE and SUGARCREPE to model’s unintentional overfitting: The NEGCLIP models learned to exploit artifacts that can be used to easily distinguish hard negatives from positives on ARO+CREPE, instead of actually improving compositionality. Thus, when evaluated on SUGARCREPE where the artifacts are removed, the improvement from NEGCLIP drastically reduces. These results imply that NEGCLIP’s effectiveness is overestimated on existing benchmarks, and we may still need further innovations to fundamentally improve a model’s compositionality.²

²In Appendix A.4.3, we provide further results on training NEGCLIP with hard negatives filtered with our adversarial refinement mechanism.

Model	Training	Hard Negative Used	ARO+CREPE				SUGARCREPE		
			REPLACE	SWAP	NEGATE	SHUFFLE	REPLACE	SWAP	ADD
Human			95.33	100	99.33	96.00	98.67	99.50	99.00
ViT-B/32	Pretrained	N/A	75.71	71.58	76.89	72.06	80.76	63.27	75.09
	CLIP finetuned	N/A	77.06	68.81	61.19	63.04	84.76	70.83	85.58
	NEGCLIP finetuned	REPLACE	94.51	90.04	85.06	88.15	88.27	74.89	90.16
		SWAP	82.88	94.48	77.57	87.00	85.54	76.21	86.56
		NEGATE	77.24	68.91	99.54	64.28	84.97	70.29	85.84
		Released in	85.72	94.35	83.51	90.45	85.36	75.33	87.29
	CLIP from scratch	N/A	69.93	59.96	55.36	68.78	69.54	60.33	67.63
RN50	NEGCLIP from scratch	REPLACE	89.04	66.51	60.90	75.23	74.32	62.65	72.92
		SWAP	72.33	92.29	64.51	84.84	73.31	68.35	71.93
		NEGATE	70.09	60.29	99.45	69.03	72.74	60.89	70.47
		REP + SW + NEG	86.30	88.60	99.34	82.93	75.26	67.69	73.08

Table 2.5: Re-evaluating hard negative augmented training shows that the method’s improvements on existing benchmarks (ARO+CREPE) are hugely overestimated, particularly when the test hard negative type matches the one used in training, which can be attributed to overfitting the artifacts.

Color notations: Gains compared to standard CLIP (finetuned / from scratch) > 10% .

Comprehensive evaluations on existing pretrained vision-language models. We present four key findings in our evaluation over 17 pretrained CLIP models on SUGARCREPE, with results reported in Table 2.6 and visualized in Figure 2.5.

The best pretrained CLIP models demonstrate some compositional understanding but still have overall large rooms for improvements. Table 2.6 shows that the largest pretrained CLIP models, e.g. , OpenAI’s RN50x64, LAION’s xlm-roberta-large-ViT-H-14, and DataComp’s ViT-L-14, achieve near-human performance on REPLACE-OBJ. However, on REPLACE-OBJ, smaller models pretrained on small datasets still suffer from big drops in performance — 23% and 43% respectively for DataComp’s small and medium models — compared to humans. Additionally, on nearly all other hard negative types, there are clear gaps (larger than 10%) between the best model performances and human performances, showing an overall large room for improvements in current models’ compositionality.

All models struggle at identifying SWAP hard negatives, regardless of their pertaining dataset and model size. Among the three types of hard negatives, SWAP hard negatives present the biggest challenge to the pretrained CLIP models, even though humans can easily tell them apart from the positive captions. We observe in Table 2.6 that all models demonstrate low performance on both SWAP-OBJ and SWAP-ATT hard negatives regardless of their pretraining dataset and model sizes, with the difference from human performance

Source	Model	Data Size	Model Size (M)	REPLACE			SWAP		ADD		Average
				Object	Attribute	Relation	Object	Attribute	Object	Attribute	
	Human			100	99	97	99	100	99	99	99
Text-only model	Vera			49.39	49.62	49.36	49.19	49.40	49.42	49.57	49.42
	Grammar			50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
OpenAI	RN50	400M	102	91.77	80.58	69.99	61.79	68.47	74.54	69.65	73.83
	RN101		120	92.49	83.88	67.07	56.50	65.92	75.46	70.09	73.06
	ViT-B-32		151	90.92	80.08	69.20	61.38	63.96	77.21	68.79	73.08
	ViT-B-32-negclip		151	92.68	85.91	76.46	75.20	75.38	88.80	82.80	82.46
	RN50x4		178	92.68	82.99	67.57	65.04	63.36	79.34	70.09	74.44
	RN50x16		291	93.46	82.11	69.20	63.01	65.77	80.70	75.87	75.73
	ViT-L-14		428	94.07	79.19	65.15	60.16	62.31	78.32	71.53	72.96
	RN50x64		623	94.49	83.50	70.63	61.79	66.67	83.27	73.99	76.33
LAION	roberta-ViT-B-32	2B	212	92.86	84.90	72.40	63.01	71.02	87.34	79.91	78.78
	ViT-H-14		986	96.49	84.77	71.76	67.48	73.12	92.05	85.84	81.64
	ViT-g-14		1367	95.76	85.03	72.40	63.01	71.17	91.51	82.08	80.14
	ViT-bigG-14	5B	2540	96.67	88.07	74.75	62.20	74.92	92.19	84.54	81.91
	xlm-roberta-base-ViT-B-32		366	93.16	84.01	69.20	63.41	67.57	87.78	81.07	78.03
	xlm-roberta-large-ViT-H-14		1193	96.85	86.04	72.05	63.82	72.07	93.11	86.13	81.44
DataComp	small: ViT-B-32	13M	151	56.90	56.85	51.99	50.81	50.00	53.93	60.55	54.43
	medium: ViT-B-32	128M	151	77.00	69.54	57.68	57.72	57.06	66.73	64.88	64.37
	large: ViT-B-16	1B	150	92.68	79.82	63.94	56.10	57.66	84.34	78.61	73.31
	xlarge: ViT-L-14	13B	428	95.52	84.52	69.99	65.04	66.82	91.03	84.97	79.70
OpenAI	GPT-4V			96.31	93.53	90.26	83.13	90.09	91.59	91.76	90.95

Table 2.6: Our evaluation of pretrained CLIP models on SUGARCREPE shows that they demonstrate compositionality on some hard negatives but are far from human performance on others, especially on SWAP hard negatives or ones perturbing attributes and relations (also illustrated in Figure 2.5: lower overall performance on SWAP, and lower performances on attributes/relations compared to objects). We additionally evaluate recently introduced GPT-4V. While it demonstrates strong results, there is still gap to human-level performance.

reaching from 27% to 50%.

Existing models are object-centric, struggling to compose attributes and relations. We find that existing pretrained models are a lot better at composing objects than attributes or relations (Table 2.6). This finding holds for both REPLACE and ADD hard negatives but not the most difficult SWAP negatives, where models perform equally poorly on both SWAP-OBJ and SWAP-ATT. On REPLACE hard negatives, even though most models achieve human-level performance on REPLACE-OBJ, they all suffer from a drop in performance on REPLACE-ATT and REPLACE-REL, where the drop is as large as 15% and 29% respectively. Similarly, on ADD hard negatives, all models except for DataComp’s small:ViT-B-32 experience a decrease in performance from ADD-OBJ to ADD-ATT, with the largest difference reaching 10%.

Models’ performance on SUGARCREPE correlates with their ImageNet zero-shot accuracy. We show in Figure 2.5 that there is a positive correlation between models’ performance on SUGARCREPE and their zero-shot accuracy on ImageNet. This correlation is moderate on SWAP-OBJ and ADD-ATT (Pearson correlation coefficient $r = 0.78$ and $r = 0.75$ respectively) and strong on all other hard negatives ($r > 0.8$).

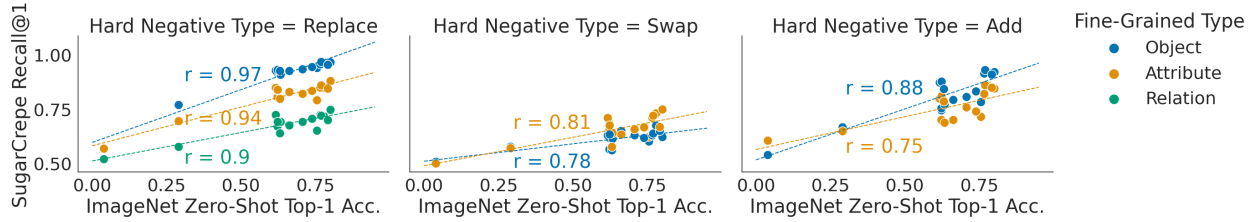


Figure 2.5: We plot pretrained vision-language models’ zero-shot top-1 accuracy on ImageNet versus their retrieval recall@1 on SUGARCREPE, where r is the Pearson correlation coefficient. This plot suggests that models’ ImageNet zero-shot accuracy positively correlates with their compositionality.

2.4.3 Limitation and future work

Scope of the compositionality benchmarks and vision-language models. We focus our scope in this chapter on compositionality benchmarks formulated as image-to-text retrieval task. While this is currently the most prevailing evaluation framework, future research can characterize compositionality evaluation as text-to-image retrieval problem, as in the initial efforts considered by [Ray et al., 2023; Thrush et al., 2022]. More importantly, we hope our work can guide future efforts in creating and ensuring faithful compositionality benchmarks in text-to-image form. In addition, we focus our evaluations on contrastively learned vision-language models [Radford et al., 2021]. Future work should include and characterize the compositionality of modern generative vision-language models [Alayrac et al., 2022; Chen et al., 2022; Li et al., 2023c; Tschannen et al., 2023].

Potential biases imposed by language models. In this chapter, we identify *two* human interpretable dataset biases, the nonsensical and non-fluent biases, which may not cover all dataset artifacts that could possibly be exploited by a model. By leveraging ChatGPT in generating hard negatives, the generated captions may also exhibit hard to detect biases imposed by the language model, e.g., watermarks [Kirchenbauer et al., 2023]. Future work may utilize more sophisticated adversarial filtering techniques that train models to detect and remove spurious dataset artifacts beyond human comprehension [Zellers et al., 2019; Le Bras et al., 2020].

Shifts in language model behavior. Our work leverages ChatGPT to generate hard negatives. However, recent work has pointed out that the underlying model behind these APIs may change, resulting in model behavior shifts [Chen et al., 2023; Liu et al., 2023b]. We discuss how this potential model behavior shift may affect our proposed dataset construction pipeline. Specifically, while there may be variances on the

quality of the generated texts, we note that our employed adversarial refinement mechanism can ensure that the final evaluation set is free of the identified artifacts. In the case when ChatGPT improves and generates higher-quality captions, the refinement mechanism will filter out less examples and we can more efficiently create the final evaluation set. On the other hand, if ChatGPT degrades and shifts towards generating less fluent and plausible captions, the refinement mechanism will filter out more generated examples and we would need to generate more candidates in order to create an evaluation set of the same desired size. As a result, while the efficiency of the proposed dataset construction pipeline depends on quality of the language model used, our pipeline ensures the generated set does not contain the identified biases. In the large language model era, we see these capable models as productive tools one can leverage to efficiently process and create data. We do however deem careful validation mechanisms, such as our manual and automatic filtering technique, necessary to ensure that the ultimate goal is properly achieved.

2.4.4 Societal impact

As vision-language models such as CLIP [Radford et al., 2021] are becoming the foundation models for many downstream applications [Rombach et al., 2022; Ramesh et al., 2022], it is imperative to understand the limitations of these models to avoid misuses and undesirable outcomes [Cho et al., 2022; Bianchi et al., 2022]. Compositionality benchmarks probe a model’s understanding of finer-grained concepts, and hence allow us to identify blind spots [Yuksekgonul et al., 2023; Zhao et al., 2022; Ma et al., 2022] of seemingly powerful models deemed by standard classification and retrieval benchmarks [Deng et al., 2009; Lin et al., 2014]. Our work further alleviates common artifacts in existing compositionality benchmarks that result in overestimation of a model’s capability. We hope our proposed benchmark SUGARCREPE leads to more faithful assessment of a vision-language model’s compositionality, and can hence guide more accurate usages of the models. Nevertheless, we note that strong performances on SUGARCREPE do not imply perfect models. We envision SUGARCREPE being one of the many benchmarks used to comprehensively understand the abilities of vision-language models from various aspects.

2.5 Task-Me-Anything: A Benchmark Generation Engine

Consider a user who wants to know “Which open-sourced MLM is best at recognizing objects even if the object is rotating?”. TASK-ME-ANYTHING provides an interface for the user to pose such questions and provides them with an answer (Figure 2.6). It contains a taxonomy to symbolically represent visual content. A query identifies the relevant portion of the Taxonomy required to answer the query. It also contains task generators that create input-output pairs that test for a specific capability. The Taxonomy subset is used to select the appropriate task generator. We adopt the common input-output format used in existing benchmarks, i.e. , all the task instances in TASK-ME-ANYTHING contain an image/video, a question, and multiple options with one ground truth answer. MLMs will be evaluated on these generated task instances and the results will be returned back to the user. Finally, it also supports queries that ask for, not just the best performing model, but also task instances (“Find top-10 task instances that GPT4O performs the worst”) or taxonomy concepts (“Find the objects that GPT4O’s performance is higher than a threshold”), as well as on-budget results approximation methods for such fine-grained queries. unlike most existing procedural data systems, we design TASK-ME-ANYTHING so that the generation space of tasks can be expanded by adding new source data and/or task generator code. More details in Appendix B.1 and B.2.

2.5.1 Taxonomy and Task Generators

We adopt a spatio-temporal scene graph as a representation of concepts represented in an image or video [Krishna et al., 2017b; Ji et al., 2020]. In a scene graph, objects and their corresponding attributes are nodes and relationships between objects are edges. Scene graphs have already been utilized in programmatic generation of VQA task instances in datasets like GQA [Hudson and Manning, 2019] and AGQA [Grunde-McLaughlin et al., 2021; Gandhi et al., 2022]. For example, the object nodes of the scene graph can be used to create counting tasks, relationships edges can encode relative locations and generate spatial understanding tasks, and attributes can ask about color, material, physical states like rotation, etc. The scene graph representation is generic: it can be extended to incorporate concepts like lightning conditions and ask questions about the light source, illumination, and shadows [Bashkirova et al., 2023]. In fact, we extend traditional scene graphs with 3D object assets from Objaverse [Deitke et al., 2023, 2024], enabling us to ask questions about any objects with available 3D models and their spatial positions, *etc.*.

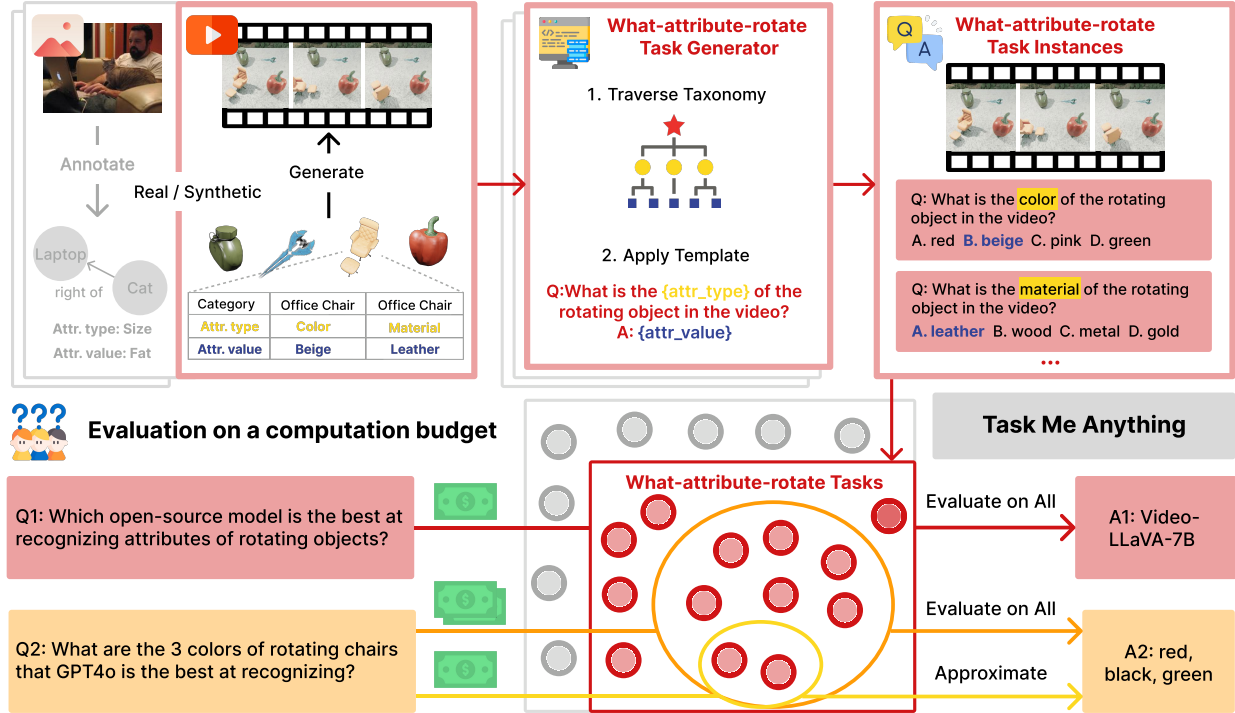


Figure 2.6: We present the key components in TASK-ME-ANYTHING. The top part illustrates the task generation process with an example video synthesized with 3D objects and their annotations, and the task generator for generating questions about rotating objects’ attributes. The bottom part depicts the model evaluation process, which selects the relevant tasks based on the user’s query and their budget and performs either full evaluation or results approximation to answer the query.

A task generator is a Python program that can generate VQA task instances given a subset of the taxonomy. It generates questions using templates of the type: "How many <target object> are there in the image?", where the <target object> can be filled with objects in the scene graph such as "telephone". Also, it programmatically produces the ground truth answer based on the scene graph. It synthesizes incorrect yet plausible options for each question [Zellers et al., 2019]. For the visual input associated with every question, we use the images [Hudson and Manning, 2019] and videos [Grunde-McLaughlin et al., 2021] annotated with scene graphs. However, scene graph data is expensive and therefore, limited. To facilitate diverse user queries, we programmatically generate images/videos from scene graph representations [Cascante-Bonilla et al., 2022; Andreas et al., 2016]. Since image/video generation models can introduce potential errors into our evaluation pipeline, we leave the use of generative models to future work. Instead, we programmatically generate image/video layouts and render them using Blender [Community, 2018] with 3D object models [Deitke

et al., 2023, 2024] via the following two approaches: 1) *2D sticker image* (abbreviated to 2D): Inspired by the SHAPES dataset [Andreas et al., 2016], we position individual 2D rendering images of 3D object models in a grid (either 2x2 or 3x3) to compose an image, which is fast to generate but lack realism, e.g. , plausible object co-occurrences, lighting, shadows, *etc.* are absent. and 2) *3D tabletop scene* (abbreviated to 3D): To overcome the limitations of the 2D approach, we render tabletop scenes to generate images after placing the 3D object assets on the table [Michel et al., 2024]. Similarly, we generate videos and adjust the position and angle of the objects across different key frames to make objects move and rotate. Such rendered images/videos are more realistic since Blender also supports lightning and collision controls.

Concretely, we use the term *task plan* to refer to the ingredients that a task generator requires for task generation, which contain the necessary task metadata and configurations. 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/video, question, options, and ground truth answer tuple that comprises a single evaluation test case and is generated by a task generator with a specific task plan. 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. We refer to this family of task instances that can be generated by a task generator with a specific task plan as *task class* or *task*, a conceptual abstraction of all task instances derived from the same task plan. Finally, each task generator is implemented for a specific type of task, e.g. , the 2D how-many task generator is for generating counting tasks with *2D sticker image* images and has two major functionalities. First, it should define the schema of the task plan it can input and be able to enumerate all the possible task plans given the source data, e.g. , the 3D object models and annotations; Second, it can generate concrete task instances given a valid task plan.

2.5.2 Query-Centric Evaluation

Given the millions of task instances that TASK-ME-ANYTHING can generate, it is computationally infeasible to evaluate even a single model on the entire task space. It would also take too long to be useful for everyday

users. We describe how TASK-ME-ANYTHING supports on-demand task generation and evaluation to address user queries.

Because each task generator supports generating all the task plans without generating the actual task instances and these task plans, once pre-computed, can act as a structured representation of the task space, users can leverage them to identify tasks relevant to their queries and then opt to only generate and evaluate the models on the relevant tasks. For example, image a user query "Which open-source model is the best at recognizing shapes of rotating objects?", the user can leverage the what-attribute-rotate task generator to compute all the task plans, select those related to recognizing shapes and then use them to generate actual task instances to evaluate and compare open-source models. Such a workflow enables *query-centric* task generation and evaluation, avoiding generating and evaluating the entire task space.

Fine-grained user queries. While many user queries can be simply addressed by the aforementioned workflow, we additionally support 4 types of fine-grained user queries for investigations regarding individual tasks and taxonomy concepts:

- ① *Top-K queries* enable users to request the top-K taxonomy concepts or tasks (e.g. , "Return the top-10 colors/tasks that LLAVA-13B struggles with").
- ② *Threshold queries* allows users to query for taxonomy concepts or tasks where model performance surpasses or falls below a given threshold (e.g. , "Find all the object recognition tasks that both LLAVA-NEXT-34B and GPT4O perform below 30% accuracy?").
- ③ *Model comparison queries* identify where one model outperforms another by a specified margin, enabling comparative analysis (e.g. , "Which types of tasks does GPT4O outperform GEMINI-PRO?").
- ④ *Model debugging queries* identify where a model’s performance deviates from its average by one standard deviation, facilitating the ability to uncover models’ inconsistent behavior. (e.g. , What action does VIDEO-LLAMA-2-7B struggle to recognize compared to other actions?).

Addressing fine-grained queries under a budget. These fine-grained user queries might involve a large number of tasks to generate and evaluate to obtain query results. For example, to obtain the top K tasks of a task generator that model performs the worst, we have to evaluate all the possible tasks. To address this, we draw on active learning literature [Karamcheti et al., 2021], to implement 3 efficient query results

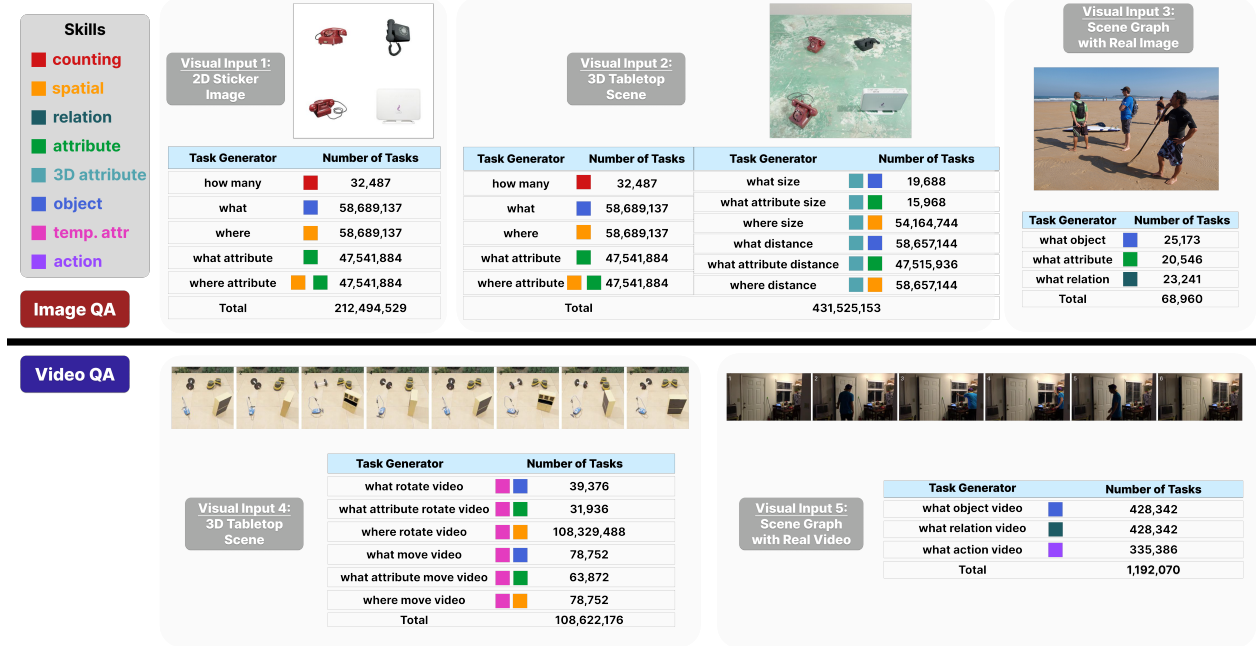


Figure 2.7: The statistics of generatable tasks of each task generator and example image/video in TASK-ME-ANYTHING. We each task generator with high-level perceptual skills and this collection of task generators can collectively generate over 750M VQA tasks.

approximation approaches for these fine-grained user queries:

- ① *Random* randomly samples a subset of task instances from the total possible for that query. MLMs are evaluated on only this subset.
- ② *Fitting* similarly samples a random subset and evaluates MLMs. The results are used to train an efficient function approximator for each MLM. This function approximator learns to predict an MLM’s performance on a task, by featurizing the task plan—never actually generating the task instance itself. While many model choices are applicable, we adopt the Gaussian Process regressor throughout this work since it renders stable performance in preliminary studies. It uses this function to approximate the MLM’s performance on the remaining task space.
- ③ *Active* is similar to *fitting* but iteratively trains each function approximator using active learning. Given a smaller subset, it trains an initial function, which is then used to sample the most task instances that are most likely to lead to improved approximation results. MLMs are evaluated on these samples; the results are used to *re-train* the functions again.

Although TASK-ME-ANYTHING supports many different kinds of reasoning tasks, it currently focuses

on visual perception capabilities. We include 28 different task templates across 5 types of visual inputs: 2D sticker images (2D), 3D tabletop scene images/videos (3D), and real images/videos with manually-annotated scene graphs. In total, it can generate over 750 million possible VQA task instances (see Figure 2.7 for a breakdown). We draw image scene graphs from Visual Genome [Krishna et al., 2017b], and video spatio-temporal scene graphs from Action Genome [Ji et al., 2020]. We also include GQA [Hudson and Manning, 2019] and AGQA [Grunde-McLaughlin et al., 2021] for their real VQA instances. For 2D and 3D scenes, we select 1,996 high-quality 3D objects across 337 categories from Objaverse-LVIS, the subset of Objaverse 1.0 [Deitke et al., 2023] that has been annotated with LVIS [Gupta et al., 2019] categories. Each 3D object was manually annotated with attributes such as color, material, shape, and visible angles. More details can be found in Appendix B.3.

These 28 different task generators provide a comprehensive way to evaluate visual understanding capability including object recognition, attribute recognition, relation recognition, localization, spatial reasoning, temporal reasoning, action recognition, *etc.* (Figure 2.7). With this diversity of potential questions, TASK-ME-ANYTHING supports the evaluation at varying desired levels of granularity

For model users, TASK-ME-ANYTHING can help decide which model to use for their needs, and for model developers, it can identify the weaknesses of models to improve. For example, a model user wanting to find the best model for distinguishing different breeds of dogs can query: “What are the top 3 models for distinguishing dogs?” Similarly, a model developer might query: “Find the spatial reasoning capabilities that all models lack?” to identify some general issues in current architecture. Or they might also query: “Which types of materials do LLAVA underperform on?” and then add the corresponding data into training to enhance LLAVA’s material recognition performance.

This system is not only versatile but also scalable. By adding new task generators, assets like 3D object models, and software like Blender, DALL-E, etc., we can continuously expand its taxonomy. Updating a taxonomy of underlying capabilities is more scalable than collecting sufficient data for the rapid growth in use-cases for MLMs.

In this work, we extensively evaluate 13 open-source MLMs over 1M task instances and 18 open-source/proprietary MLMs over 8,400 task instances, both generated by TASK-ME-ANYTHING, for validating TASK-ME-ANYTHING and analyses.

Model evaluation protocol. We adopt the accuracy of the model on a task to capture the model’s performance. However, one task can contain numerous concrete task instances. In practice, we randomly generate n task instances for a task and then use the model’s accuracy on the n task instances as a proxy of the model’s accuracy on the task. For prompts used to evaluate the models, to fairly evaluate the model’s performance and enhance the robustness of the results, we use two versions of prompts: a succinct prompt and a detailed prompt. The succinct version simply adds ‘Select from the following choices’ between the question and the options [Fu et al., 2024c], while the detailed prompt includes more instructions such as: ‘Based on the image/video’, and also enclose the options within parentheses (e.g., ‘(A) camera (B) telephone’) and ends the prompt with ‘Best Option: (’ to guide the model to output the option only [Li et al., 2024b]. The exact prompt template can be found in Figure 2.8." For option extraction, we match the model output to three types of option representations: 1) option identifier, e.g. , "(A)", 2) option name, e.g. , "camera", and 3) option identifier and name, e.g. , "(A) camera" in order to increase the recall of the option extraction.

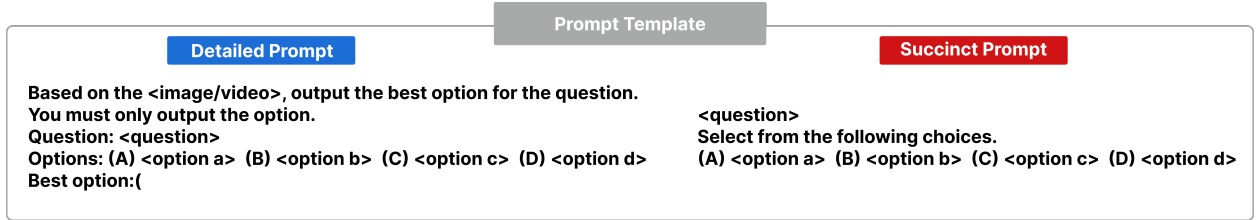


Figure 2.8: We adopt two distinct prompts, the detailed prompt and the succinct prompt, in our evaluation to assess models’ sensitivity to different prompts.

TASK-ME-ANYTHING-RANDOM: A random set of tasks. To offer an overview of the task space of the current TASK-ME-ANYTHING, we create a random subset of 100 tasks from each task generator. For each task, we randomly generate 3 task instances, resulting in 5,700 ImageQA task instances and 2,700 VideoQA task instances. We refer to this random set as TASK-ME-ANYTHING-RANDOM, which we release as a benchmark. We evaluate 18 open-source/proprietary MLMs on this set using both detailed prompt and succinct prompt.

TASK-ME-ANYTHING-DB: A database of model evaluation results. We also randomly select over 100K tasks across all the task generators and generate 15 task instances for each task, leading to over 1M task instances in total. Then we evaluate 13 open-source MLM models on the generated task instances

using detailed prompt, leading to a total number of 24,240,780 <model, task instance> evaluation pairs. We refer to this set of evaluation results as TASK-ME-ANYTHING-DB, which we use to study the query results approximation methods and release for future study of model performance prediction.

TASK-ME-ANYTHING-UI: A graphical interface for model performance investigation. TASK-ME-ANYTHING allows users to query for tasks that most resemble their application. As such, TASK-ME-ANYTHING doesn't have to be limited to a static leaderboard commonly seen with most other benchmarks. Instead, we make TASK-ME-ANYTHING's findings accessible through an interactive graphical user interface. Our interface allows users to specify their needs without writing any code. They can select parts of the taxonomy that best represent their application. We use the evaluation results in TASK-ME-ANYTHING-DB obtained our explorations to build a simple example interface: TASK-ME-ANYTHING-UI³. It consists of four tabs: the **overall** tab reports model performance across a dozen MLM across different subsets of TASK-ME-ANYTHING's taxonomy; the **task embedding** tab visualizes different task instances in a 2D space and allows users to observe model behavior across similar tasks; the **surprisingness** tab highlights tasks where a model achieves surprisingly better or worse performance compared to similar tasks; and the **query** interface supports users to conduct query-centric investigation of models' capabilities or limitations using the four types of fine-grained user queries mentioned above (Figure 2.9). More details can be found in the Appendix B.3.3.

2.6 Task-Me-Anything: Analysis of MLM Capabilities

We use TASK-ME-ANYTHING to conduct multiple analyses to highlight its different use cases, while simultaneously drawing insights about today's MLMs (More details in Appendix B.6). Specifically, we evaluated 18 MLMs on TASK-ME-ANYTHING-RANDOM for Query 1 and 4 and reused the evaluation results of TASK-ME-ANYTHING-DB for Query 2, 3, 5, and 6. Finally, we leverage TASK-ME-ANYTHING to provide an in-depth analysis on GPT4O as Query 7.

³<https://huggingface.co/spaces/zixianma/TaskMeAnything-UI>

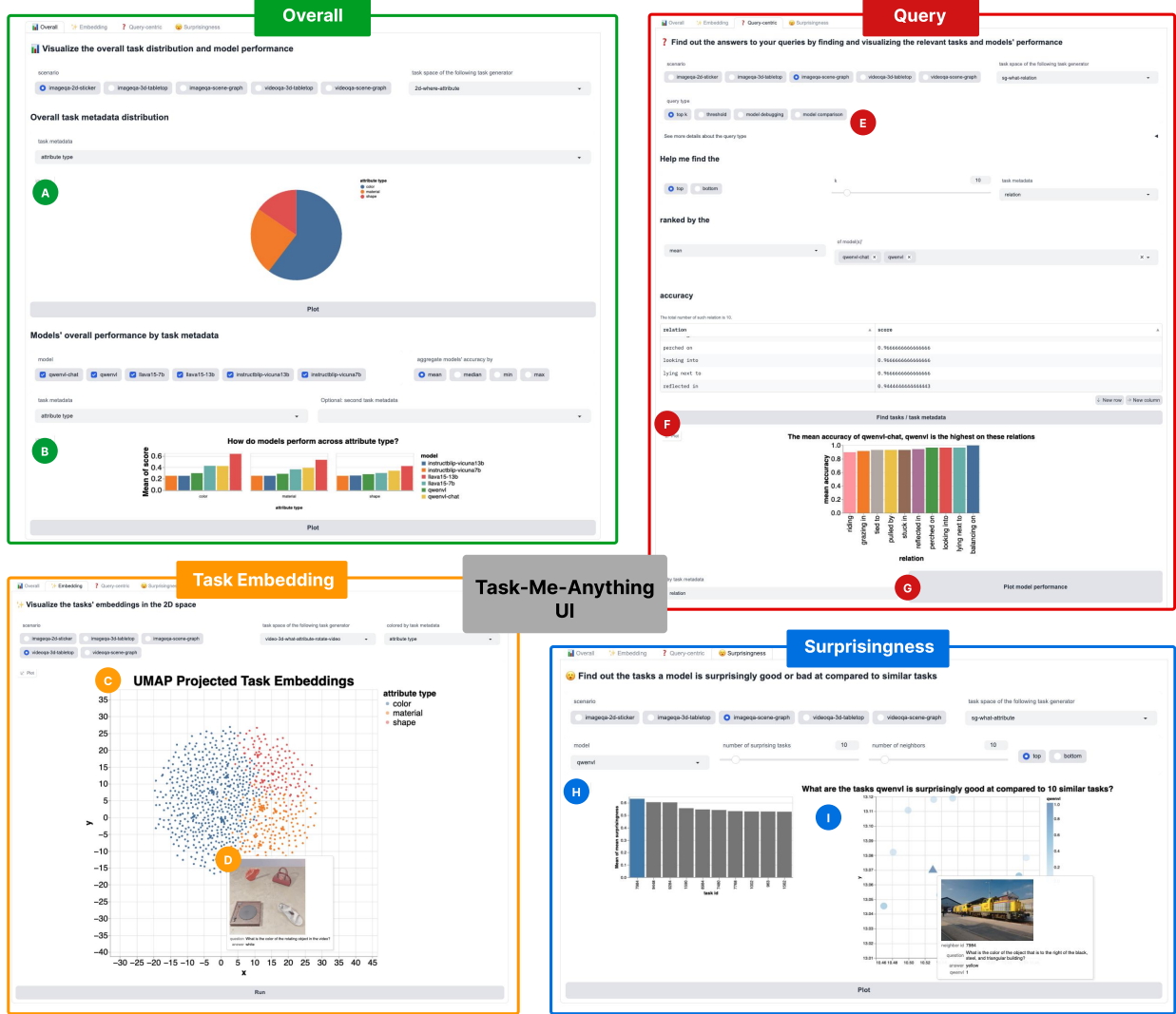


Figure 2.9: TASK-ME-ANYTHING-UI Interface.

2.6.1 Fine-Grained Capability Profiling

Query 1: How do models perform over a random subset of all possible questions? We evaluated 18 MLMs on the TASK-ME-ANYTHING-RANDOM test set (Figure 2.10) to gain an overview of model performance. (Details in Appendix B.4).

For ImageQA tasks, the latest open-sourced models, such as INTERNVL-CHAT-1.5-24B and LLAVA-NEXT-34B, surprisingly outperform popular proprietary models, achieving state-of-the-art performance as shown in recent benchmarking results [Contributors, 2023].

For VideoQA tasks, in addition to natural VideoQA models, we evaluated larger or proprietary ImageQA models, such as GPT4V, by combining four frames of a video into a single picture. We observed that natural VideoQA models, such as VIDEO-LLAVA-7B, still have a significant performance gap compared to large ImageQA models, even though ImageQA models are not trained on video and only use a combined image rather than the full video.

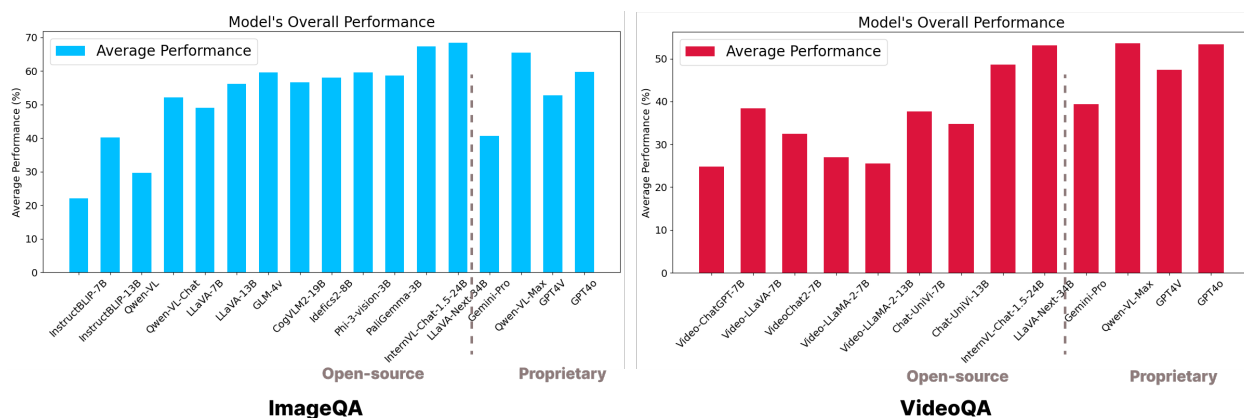


Figure 2.10: Model performance on the TASK-ME-ANYTHING-RANDOM, a random subset of tasks from TASK-ME-ANYTHING.

Query 2: Are current models sensitive to prompts? Our evaluation indicates that current models are still prompt-sensitive (Figure 2.11). We tested our model on two versions of prompts: a succinct version that only describes the question and a detailed version that includes more instructions.

First, we observed that MLMs are still highly prompt-sensitive, and using different prompts might result in over 40% performance differences.

While detailed prompts generally lead to better results, certain models, such as GPT4V, perform notably better with more succinct prompts. In contrast, models like INSTRUCTBLIP-7B and QWEN-VL exhibit a marked improvement with detailed prompts compared to succinct ones. (Details in Appendix B.4).

Query 3: What skills are MLMs best and worst at? We analyze performance across different perceptual capabilities to answer: what skills are all models good or bad at? We conduct this study for both ImageQA and VideoQA tasks respectively. We find that no specific skill appears to be the best or worst across (both image and video) models (Figure 2.12). We see that all models struggle in spatial reasoning, counting

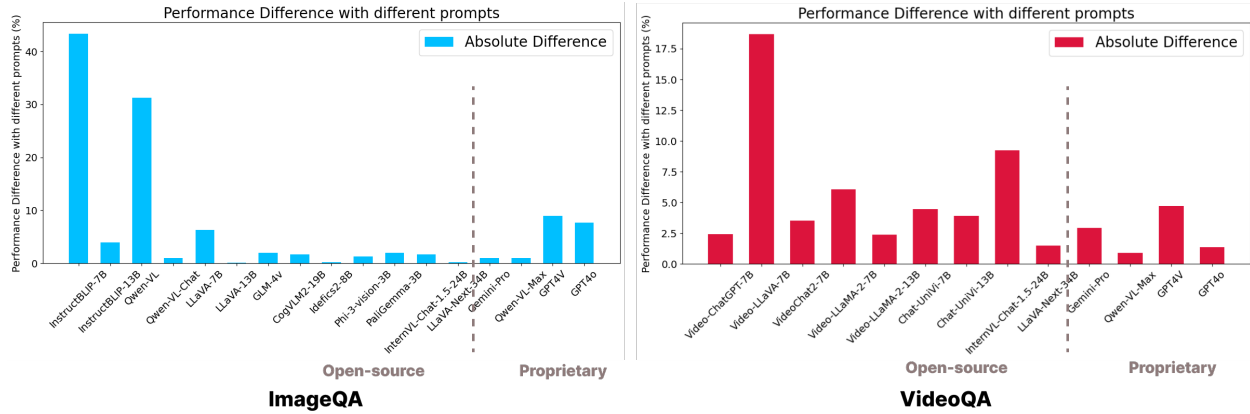


Figure 2.11: The absolute difference of model performance in different prompts in TASK-ME-ANYTHING-RANDOM.

objects, and 3D attribute understanding on ImageQA tasks, and object recognition, temporal understanding on VideoQA tasks. They perform well on object, attribute, and other relationship recognition instances. Surprisingly, we find that most MLMs perform the best at relationship understanding between objects, scoring high if not perfectly on interactional relations such as “riding”, “looking into”, “lying next to” etc. On the other hand, these models struggle the most in spatial reasoning in synthetic images, performing poorly especially on questions that ask about objects in the “middle”, “bottom” or “back” (for 3D images) part of the image. Nevertheless, some models behave differently. For example, LLaVA-13B is worst at recognizing 3D attributes, failing at identifying the “smallest” or “closest” 3D objects correctly. Meanwhile, LLaVA-7B is best at object recognition and worst at relation understanding, struggling to understand simple actions such as “touching” that other models perform well on.

Further, TASK-ME-ANYTHING also enables us to conduct analyses of models’ fine-grained skills such as recognizing a specific type of object, attribute, or relation. For example, on ImageQA tasks, we find that on average models are better at recognizing plants, understanding mood and comprehending spatial relations between real-world objects (Figure 2.14). Nevertheless, some models might showcase different strengths: LLaVA-13B is better at recognizing animals (Figure 2.14 (a)), and INSTRUCTBLIP-7B is better at understanding emotional relationships (Figure 2.14 (c)). On the other hand, for VideoQA tasks, we learn that models are better at recognizing vehicles, material and understanding spatial relationships (Figures 2.15 and 2.16).

Query 4: What is the best MLM for each specific skill? LLAVA-13B stood out as the strongest model on ImageQA tasks, achieving the best performance on all skills except for relation understanding; and VIDEO-LLAVA-7B is the overall winner on VideoQA tasks, scoring the highest on action understanding and second or third elsewhere. Specifically, we find that LLAVA-13B performs consistently better than other multi-modal models on all skills except for relation understanding, where QWEN-VL-CHAT performs better (Figure 2.12 (a)). On VideoQA tasks, in addition to VIDEO-LLAVA-7B, CHAT-UNIVI-7B is also relatively well-rounded, positioning in the top 3 models across all skills except for Attribute understanding (Figure 2.12 (b)). On the other hand, while VIDEOCHAT2-7B specializes in object, attribute, and temporal attribute understanding, it falls short on Action and Relation reasoning (Figure 2.12 (b)).

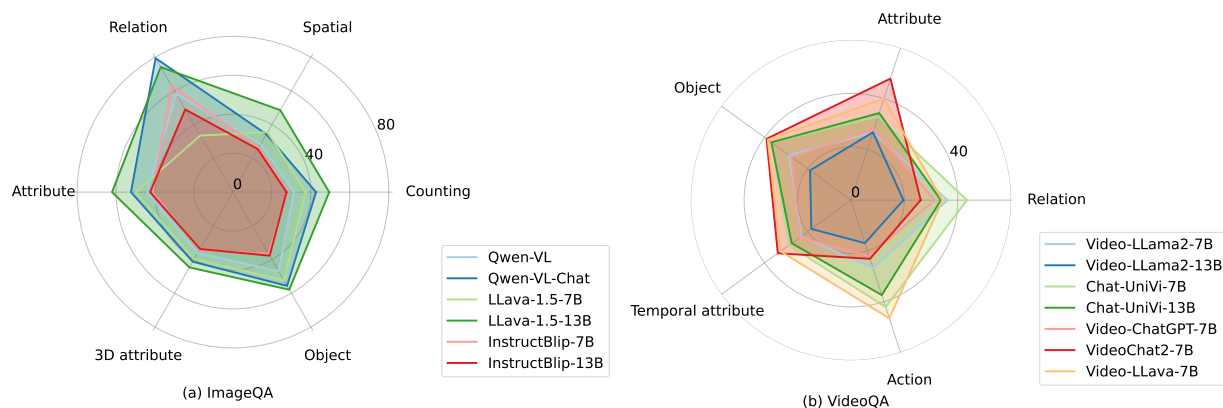


Figure 2.12: Image and VideoQA, high-level skills, all models. We plot models’ performance on Image and VideoQA tasks across all skills. We learn that models are relatively good at object and attribute recognition in both Image and VideoQA and relation understanding in ImageQA but still struggle at others.

Query 5: How does the best open-source model compare against the best proprietary model across skills? Moreover, we find that on ImageQA tasks, the best open-source model (LLAVA-NEXT-34B on object recognition, LLAVA-13B on relation understanding and INTERNVL-CHAT-1.5-24B else where) is on par with if not better than the best proprietary model (GPT4O on attribute recognition, GPT4V on counting and QWEN-VL-CHAT else where) for most skills (Figure 2.13). Notably, the best open-source model outperforms the best proprietary one on spatial reasoning by around 8% and 3D attribute by 7%. On VideoQA tasks, the best open-source model INTERNVL-CHAT-1.5-24B surpasses the best proprietary one QWEN-VL-MAX on object and action recognition but lags behind proprietary models by 5-10% on attribute,

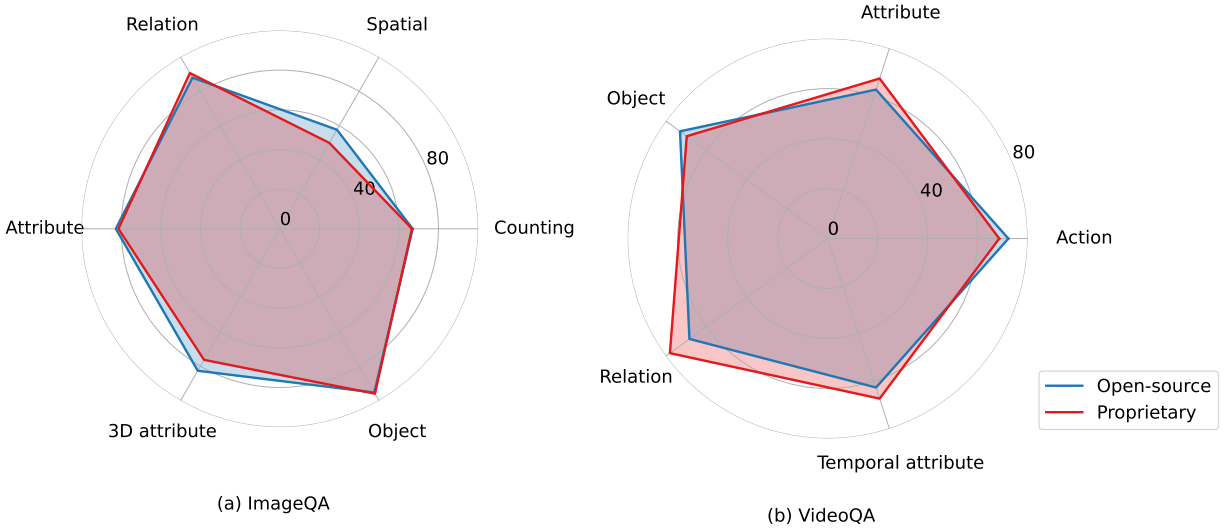


Figure 2.13: Image and VideoQA, high-level skills, open-source vs. proprietary best models. We plot the performance of the best open-source and proprietary model for each skill on ImageQA and VideoQA tasks.

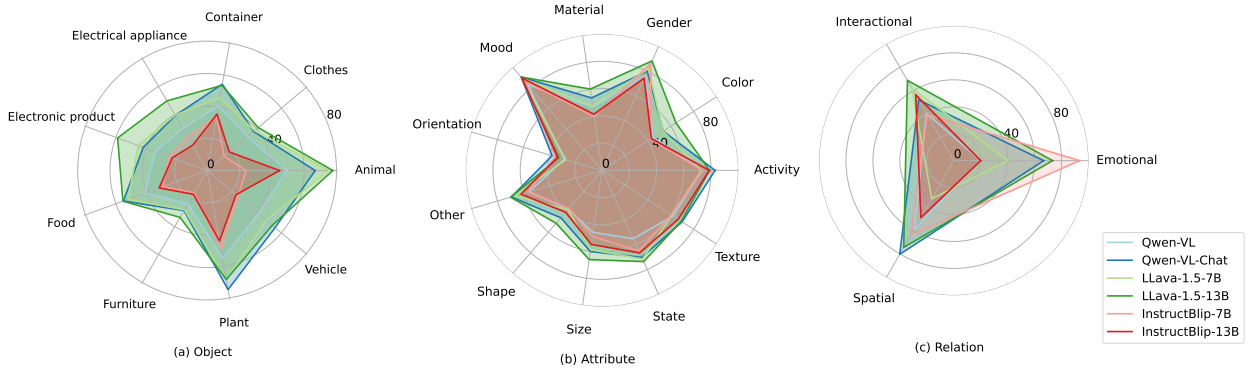


Figure 2.14: ImageQA, fine-grained skills, all models. We also analyze models’ performance on ImageQA tasks across fine-grained skills and find that models are good at recognizing plants, understanding mood, and comprehending spatial relations between real-world objects on average despite differences in individual models.

temporal attribute and relation understanding.

Query 6: How do small models compare against large models? We are also interested in the relative performance of small versus large models with the same skills. On ImageQA tasks, for example, we observe that large multi-modal models collectively perform better than smaller models on ImageQA tasks (Figure 2.17). 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 ($\mu = 0.63$)

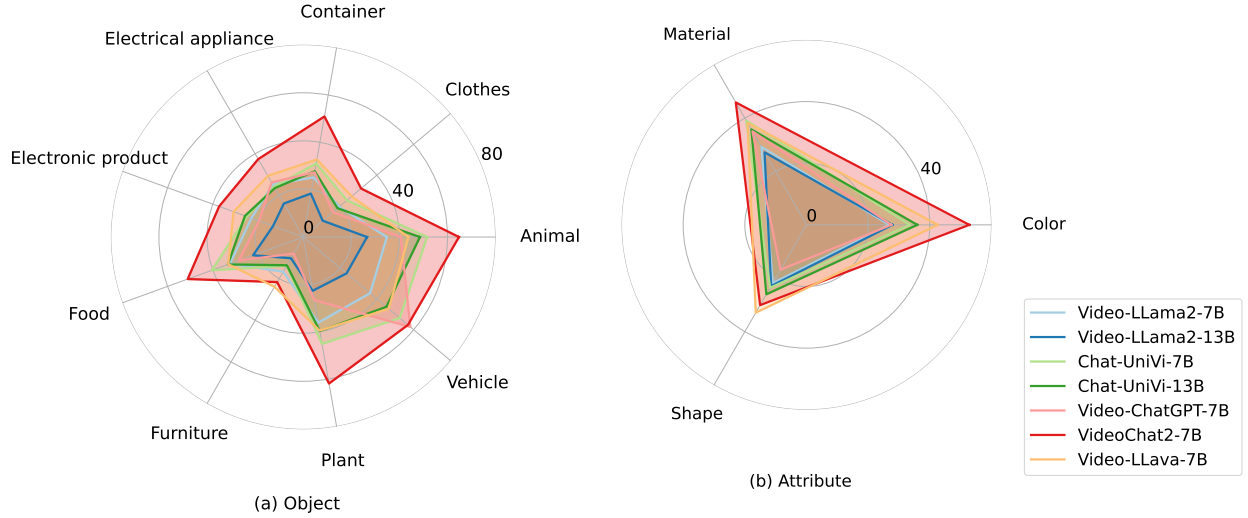


Figure 2.15: VideoQA, fine-grained object and attribute skills, all models. We present models’ performance on VideoQA tasks across fine-grained skills and find that, on average, models are good at recognizing vehicles and understanding materials in videos.

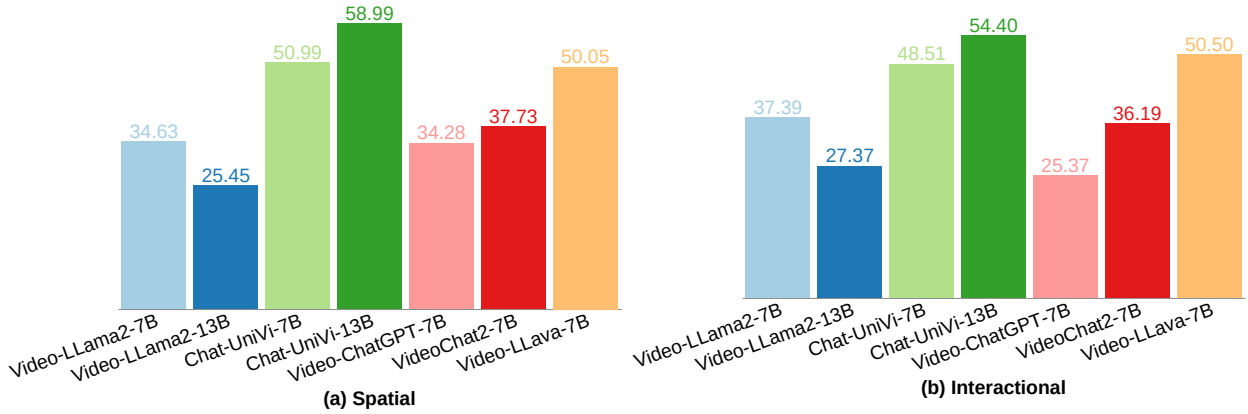


Figure 2.16: VideoQA, fine-grained relation skills, all models. On VideoQA tasks, we find that models are better at understanding spatial relations than interactional ones on average.

significantly outperforms INSTRUCTBLIP-13B ($\mu = 0.49$) on relation understanding with $p\text{-value} < 1e - 5$ (Figure 2.19).

On VideoQA tasks, interestingly, we find that small models beat larger models on VideoQA tasks on average (Figure 2.18). We hypothesize that this is because we included some strong small video models in our evaluation. For example, we see that VIDEO-LLAMA-2-7B achieves a higher score than VIDEO-LLAMA-2-13B in all skills with $p\text{-value} < 3e - 5$ (Figure 2.20), and CHAT-UNIVI-7B outperforms CHAT-UNIVI-13B on action and relation understanding with $p\text{-value} < 1e - 5$ (Figure 2.21).



Figure 2.17: Skill comparison: small vs. large models on ImageQA.



Figure 2.18: Skill comparison: small vs. large models on VideoQA.



Figure 2.19: Skill comparison: INSTRUCTBLIP-7B vs. INSTRUCTBLIP-13B.



Figure 2.20: Skill comparison: VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B.

Query 7: Are models’ strengths and weaknesses consistent across visual inputs? Further, we are curious if the models’ strong and weak skills are consistent across visual inputs. To this end, we look at models’ performance across visual inputs for object, attribute, spatial understanding, and counting as these skills involve tasks in multiple visual inputs such as 2D and 3D. We find that for the same skill, the rankings of models remain largely consistent across visual inputs (Figure 2.22). We observe strong correlations (with

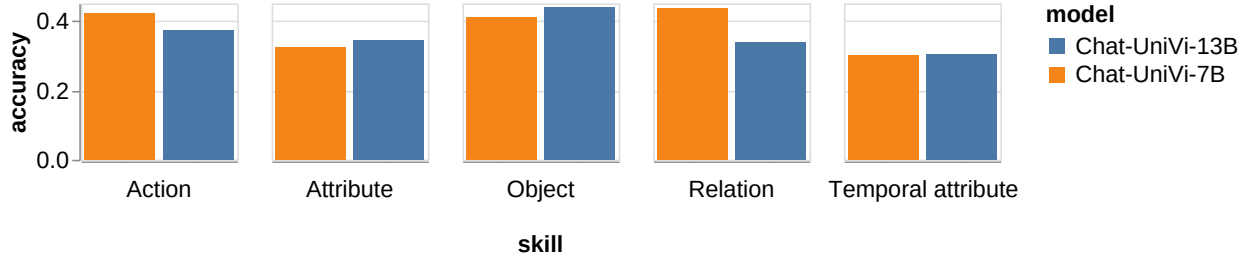


Figure 2.21: Skill comparison: CHAT-UNIVI-7B vs. CHAT-UNIVI-13B.

Spearman coefficients of 0.77-0.94) between models’ accuracy scores for different visual inputs in the same skill with only one exception: the video models’ performance on object understanding in 3D tabletop tasks is only weakly correlated (coefficient = 0.64) with their performance in scene graph tasks. This finding suggests our definition of skills is orthogonal to visual inputs and enables us to find models’ inherent strengths and weaknesses.

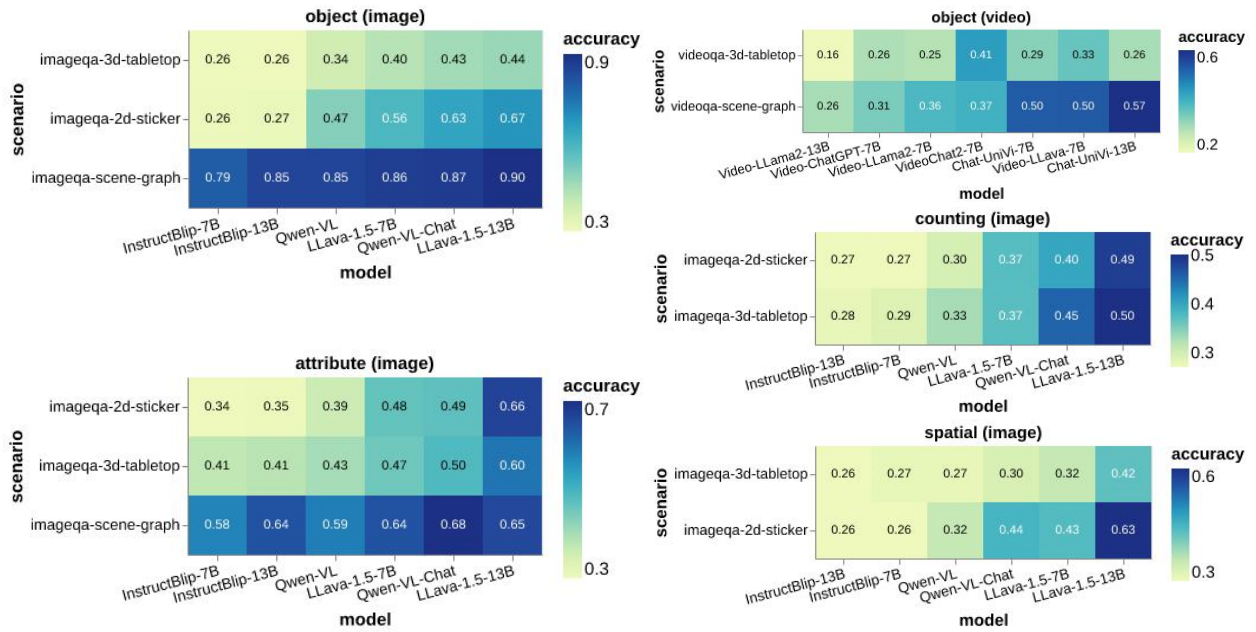


Figure 2.22: We present models’ performance for each skill across visual inputs.

Query 8: What is today’s popular proprietary model, GPT4O, bad at? Finally, we investigate GPT4O, today’s popular proprietary model: what *objects* are GPT4O bad at recognizing when rotating/moving? what *relations* are GPT4O bad at understanding? and what *attributes* of objects are GPT4O bad at recognizing?

To answer these questions, we first identify task generators for each question that can generate relevant tasks to evaluate, based on which we provide both the object/relation/attribute categories and individuals that GPT4O are bad at. Note that these are just example questions, and many more of this type can be addressed by TASK-ME-ANYTHING.

Answering with object/relation/attribute categories. First, we answer these questions by comparing GPT4O’s performance across different coarse-grained object/relation/attribute categories and their average, as shown in Figure 2.23. We can see that 1) GPT4O does not perform well in recognizing "interactional" relations in images and "spatial" relations in videos, 2) recognizing rotating/moving "furniture", "food", and "plant" is more challenging for GPT4O than other object categories such as animal and vehicle, and 3) GPT4O is worse at recognizing "color" than other attributes.

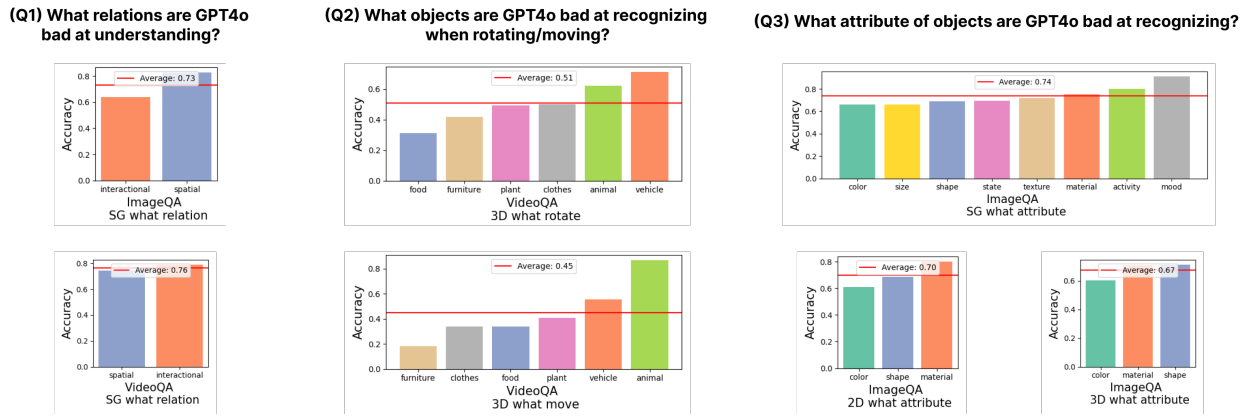


Figure 2.23: Answering Q1-Q3 with GPT4O performance on randomly generated task instances relating to coarse-grained object/relation/attribute categories.

Answering with individual objects/relations/attributes. To pinpoint the specific objects/relations/attributes that GPT4O can’t do well, we convert each question to a Top-K query regarding individual objects/relation-s/attributes, and employ our *Active* method for query results approximation with a budget of GPT4O calls. We found that GPT4O’s performance drops by a large margin (−5% to −50%) on the Top-5 objects/relation-s/attributes founded by TASK-ME-ANYTHING, indicating they remain challenging for GPT4O (Table 2.7). This example use case of TASK-ME-ANYTHING demonstrates how to leverage the system for locating the model weakness regarding fine-grained concepts.

Question	Task generator	Top-K objects/relations/attributes	Δ Perf. (%)
what objects are GPT4O bad at recognizing when rotating/moving?	VideoQA 3D what rotate	<i>fermentation product, hamper, tool, computer keyboard, mathematical instrument</i>	-21.67
	VideoQA 3D what move	<i>towel, bathtub, furniture, air conditioner, desk</i>	-19.33
what relations are GPT4O bad at understanding?	ImageQA SG what relation	<i>taller than, exiting, pushing, pushed by, between</i>	-51.05
	VideoQA SG what relation	<i>beneath, covered by, carrying, above, standing on</i>	-16.66
what attributes are GPT4O bad at recognizing?	ImageQA 2D what attribute	<i>purple, brown, red, gray, beige</i>	-5.33
	ImageQA 3D what attribute	<i>stone, rubber, textile, leather, plastic</i>	-10.67
	ImageQA SG what attribute	<i>crooked, power, lower, steep, glowing</i>	-45.45

Table 2.7: Answering Q1-Q3 with Top-K query regarding individual objects/relations/attributes. We also present the GPT4O performance drop (Δ Perf. (%)) on task instances involving found task elements as ground truth answers compared to random task instances, and show that performance drops by a large margin.

Which models should I use? When considering which models to use for your tasks, proprietary models generally offer superior performance due to their extensive training on diverse datasets and robust architectures. For tasks requiring high accuracy in object recognition, relation understanding, or attribute identification, GPT4O is currently among the best options. Its ability to handle complex queries and produce refined outputs makes it a top choice for applications demanding reliability and depth.

For those seeking open-source alternatives, LLAVA-NEXT-34B and INTERNVL-CHAT-1.5-24B stand out as strong contenders, often achieving performance levels comparable to proprietary models while being much smaller in size (20-30B) and capable of running on a single A100 GPU or two A6000 GPUs. In VideoQA tasks, current models are still in their early stages and may underperform compared to popular ImageQA models that only takes several video frames instead of full videos. However, for research and study purposes, VIDEO-LLAVA-7B are currently the state-of-the-art for these tasks.

2.6.2 Validation

We validate the accuracy of our generated evaluated data by measuring human performance on our tasks. Then, we evaluate the different approximation methods introduced in Section 2.5.2 to demonstrate their effectiveness.

Validating with human evaluation. To validate TASK-ME-ANYTHING, we first conduct a ($N = 2$) human evaluation on TASK-ME-ANYTHING-RANDOM to check the correctness of the tasks. In these random subsets, annotators achieve an accuracy of 92% – 100% for task instances from different task generators (specifically, humans perform 100% on the ImageQA 2D how-many tasks while 92% on the VideoQA 3D what-rotate tasks), indicating that our tasks are accurate and can be solved by humans. By contrast, GQA [Hudson

and Manning, 2019] and AGQA [Grunde-McLaughlin et al., 2021] report a human performance between 70% – 84%.

Ablating the approximation algorithms. We evaluate the proposed query results approximation algorithms on 1,137 queries across the 4 query types (Table 2.8). To measure the quality of the approximation, we use the evaluation results from TASK-ME-ANYTHING-DB as ground truth query results. From Table 2.8, we can see that the *Active* method outperforms both the *Random* and *Fitting* methods across nearly all query types, yet there is still room for future improvement. More details of experiments and results are in Appendix B.5.

Method	Top-K Query		Threshold Query			Model Compare Query			Model Debug Query		
	MR	HR (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
<i>Random</i>	46.81	42.30	46.88	42.48	44.05	100.00	24.58	37.28	93.39	23.27	35.04
<i>Fitting</i>	34.43	46.77	47.45	46.34	46.46	78.42	47.44	52.59	83.27	32.04	43.86
<i>Active</i>	10.79	70.55	47.39	46.83	46.55	89.94	54.88	61.87	89.95	43.84	56.44

Table 2.8: The performance of query results approximation algorithms. Top-K query uses Mean Rank (MR, lower is better) and Hit Rate (HR, higher is better) as metrics, while other queries use Precision (P), Recall (R), and F1-score (F1).

Comparison with existing benchmarks. According to Figure 2.24, we found that there is a strong positive correlation between models’ rankings on TASK-ME-ANYTHING and on other benchmarks, with strong correlations (≥ 0.8) on the most commonly used ones such as MMMU and MMBench. Notably, the average correlation between TASK-ME-ANYTHING and the other benchmarks is 0.77, which is greater than that of some other benchmarks such as ScienceQA and LLaVABench (0.70 and 0.57). These results suggest that the evaluation results on our benchmark align with those on existing benchmarks.

Analysis on synthetic data in TASK-ME-ANYTHING. TASK-ME-ANYTHING leverages both synthetic and real images/videos for evaluating MLM models. While synthetic data are controllable, low-cost, its effectiveness compared to real data remains unclear, particularly given that most benchmarks rely heavily on real datasets. To quantitatively justify the transferability between evaluations on synthetic and real-world images, we compared 18 models’ rankings on synthetic vs. realistic images and observed strong positive correlations between them. Across the three different image sources in Task-Me-Anything, we found that the correlation between models’ rankings on 2D and 3D synthetic images is the strongest ($r=0.99$). While the

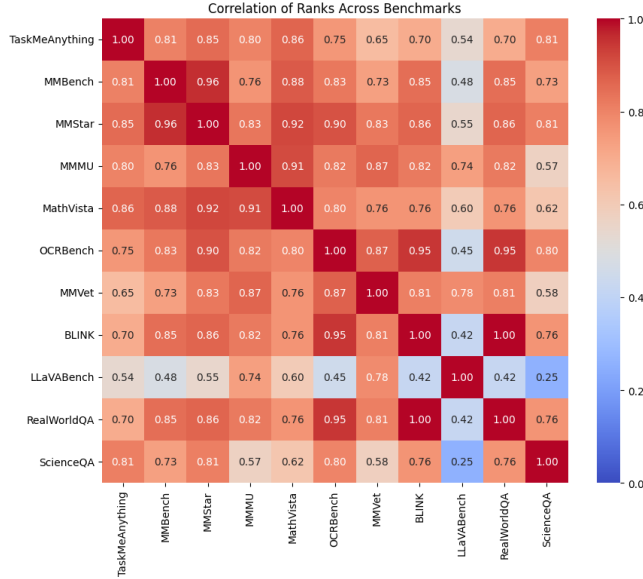


Figure 2.24: Correlations of TASK-ME-ANYTHING with other popular MLM benchmarks.

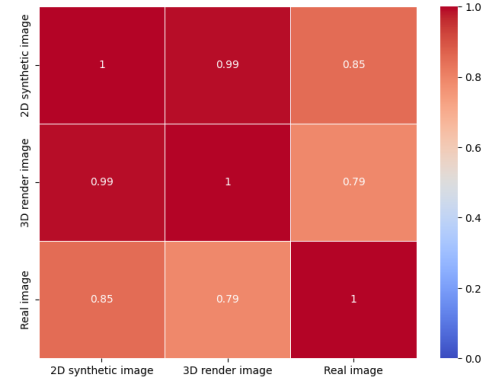


Figure 2.25: Correlations of synthetic data with real data in TASK-ME-ANYTHING.

correlations between models’ rankings on real and 2D images, and between real and 3D images are slightly smaller ($r=0.85$ and 0.79 respectively), these numbers still suggest strong positive correlations between models’ performance on synthetic images and on real images (Figure 2.25).

2.7 Conclusion

This chapter has addressed the fundamental question of how to reliably evaluate grounded visual understanding in vision-language models.

Through SUGARCREPE, we demonstrated that existing compositionality benchmarks contain systematic biases that inflate performance estimates, and introduced a more reliable evaluation methodology. Through TASK-ME-ANYTHING, we developed a programmatic benchmark generation engine that enables fine-grained, query-driven evaluation across over 750 million possible task instances, revealing that current models—while strong at object and attribute recognition—still lack spatial, temporal, and counting capabilities essential for grounded understanding.

Together, these diagnostic tools establish the measurement framework for this thesis: we can now precisely identify *where* and *how* models fail at grounded visual understanding. The subsequent chapters

address how to close these gaps—first through richer training supervision (Chapter 3), then through more aligned visual representations (Chapter 4), and finally through a fully grounded multimodal reasoning system (Chapter 5).

Chapter 3

Improving Grounded Visual Understanding with Rich Visual Instructions

3.1 Introduction

The previous chapter established that current vision-language models often lack true grounded visual understanding, performing well on coarse benchmarks while failing on compositional and fine-grained tasks. A natural question follows: how can we improve grounded visual understanding? In this chapter, we argue that a key bottleneck lies in the *training data*—most visual instruction datasets provide only sparse supervision over objects, attributes, and relationships, leaving the vast majority of visual content unsupervised.

The rapid progress of multimodal and vision–language models (VLMs) has been propelled by increasingly large corpora of image–text pairs [Gadre et al., 2024; Lin et al., 2014; Sharma et al., 2018] and, more recently, by the rise of visual instruction datasets [Liu et al., 2024a; Dai et al., 2024]. These instruction datasets teach models not only to describe images but to interact with them—to follow user queries, answer questions, and perform grounded reasoning. Yet despite their scale, such datasets remain fundamentally incomplete. Most images are only sparsely annotated, leaving the vast majority of objects, attributes, and relationships unseen, unqueried, and unsupervised. As a result, modern VLMs frequently produce confident but ungrounded predictions, reflecting a systematic lack of exhaustive vision–language supervision.

Existing practice tries to fill this gap by relying on powerful LLMs or multimodal LLMs to synthesize

visual instruction data [Liu et al., 2024a; Wang et al., 2023b; Liu et al., 2024b,d]. While this approach is convenient, it inherits several limitations. First, LLM-driven generation is effectively a black box: the reasoning behind a generated question or answer is opaque, making it difficult to ensure coverage or systematically control content. Second, even the strongest models hallucinate [Cui et al., 2023; Ji et al., 2023; Snyder et al., 2024; Huang et al., 2023; Zhao et al., 2023; Sadat et al., 2023; Vakharia et al., 2023], producing answers that do not correspond to what is visually present—precisely the failure mode exhaustive supervision aims to avoid. Third, the dependence on proprietary, large-scale models introduces scalability barriers, including high API costs and licensing restrictions that limit downstream training use [OpenAI]. These factors make current approaches poorly suited for producing dense, faithful, and controllable vision-language supervision at scale.

We present PROVISION, a system for programmatically generating exhaustive vision-language instruction data using scene graphs as a structured intermediary between pixels and language (Figure 3.1). To enable programmatic generation, we leverage *scene graphs* [Krishna et al., 2017b] as a structured representation of image semantics. In a scene graph, each object is represented as a node with attributes such as color, size, or material, while relationships between objects are depicted as directed edges. Given a scene graph, a program can generate questions like “How many red objects are there?” alongside the ground-truth answer; extending this to multiple scene graphs yields comparative questions such as “Which image has the most red objects?”. In total, PROVISION implements 24 single-image and 14 multi-image instruction data generator programs covering objects, attributes, relations, segmentation, and depth.

Beyond leveraging the human-annotated scene graphs in Visual Genome [Krishna et al., 2017b], we also automatically generate scene graphs using a pipeline of state-of-the-art models for object/relation detection [Cheng et al., 2024b; Park et al., 2025], image segmentation [Ravi et al., 2025], and depth estimation [Yang et al., 2024], enabling PROVISION to scale to any image. The resulting PROVISION-10M dataset, created from Visual Genome and DataComp [Gadre et al., 2024], contains over 10 million instruction examples. Our experiments show that this programmatically generated data consistently improves model performance: single-image instruction data yields at least 7% and 8% improvement on CVBench’s 2D and 3D splits, around 3 percentage points on QBench2, RealWorldQA, and MMMU, while multi-image instruction data brings an 8 percentage point improvement on Mantis-Eval. The data proves effective for both

pre-training and instruction tuning stages, with incorporating it in both stages yielding the best performance.

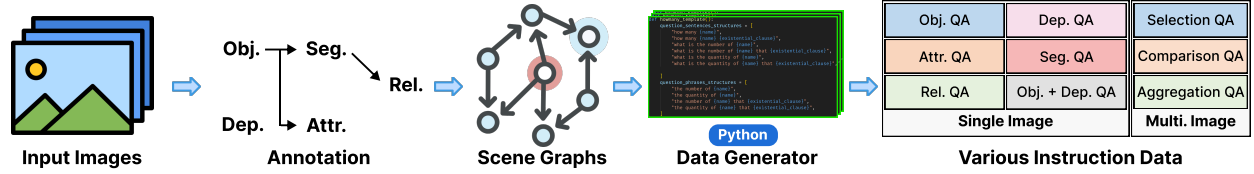


Figure 3.1: Overview of PROVISION system. It can generate a scene graph for any image, enabling programs to synthesize instruction data procedurally from the scene graph. If the scene graph exists, it directly generates instruction data based on it.

3.2 Related Work

Programmatic data generation for VLMs. The dominant paradigm for creating visual instruction data relies on distilling outputs from large language models or multimodal models [Liu et al., 2024a; Wang et al., 2023b; Liu et al., 2024b,d]. While convenient, this approach offers limited control over content coverage, is prone to hallucination, and depends on proprietary systems. An alternative line of work generates training data programmatically from structured annotations, drawing on the tradition of synthetic data generation in computer vision [Wang et al., 2024b]. Our approach follows this direction, using scene graphs to produce instruction data that is deterministic, auditable, and free of model-induced hallucination.

Scene graph-based visual understanding. Scene graphs [Krishna et al., 2017b] provide a structured representation of images by encoding objects as nodes, their attributes as node properties, and inter-object relationships as edges. They have been used extensively for tasks such as image retrieval [Johnson et al., 2015], visual question answering [Hudson and Manning, 2019], and image generation [Johnson et al., 2018]. In this work, we repurpose scene graphs as an intermediary for instruction data synthesis. We extend the standard Visual Genome scene graph definition with segmentation and depth annotations, creating an augmented representation that supports a broader range of vision-centric questions than prior scene-graph-based approaches.

Visual instruction tuning datasets. Visual instruction tuning [Liu et al., 2024a; Dai et al., 2024] has emerged as a critical stage in training multimodal models, teaching them to follow diverse user queries

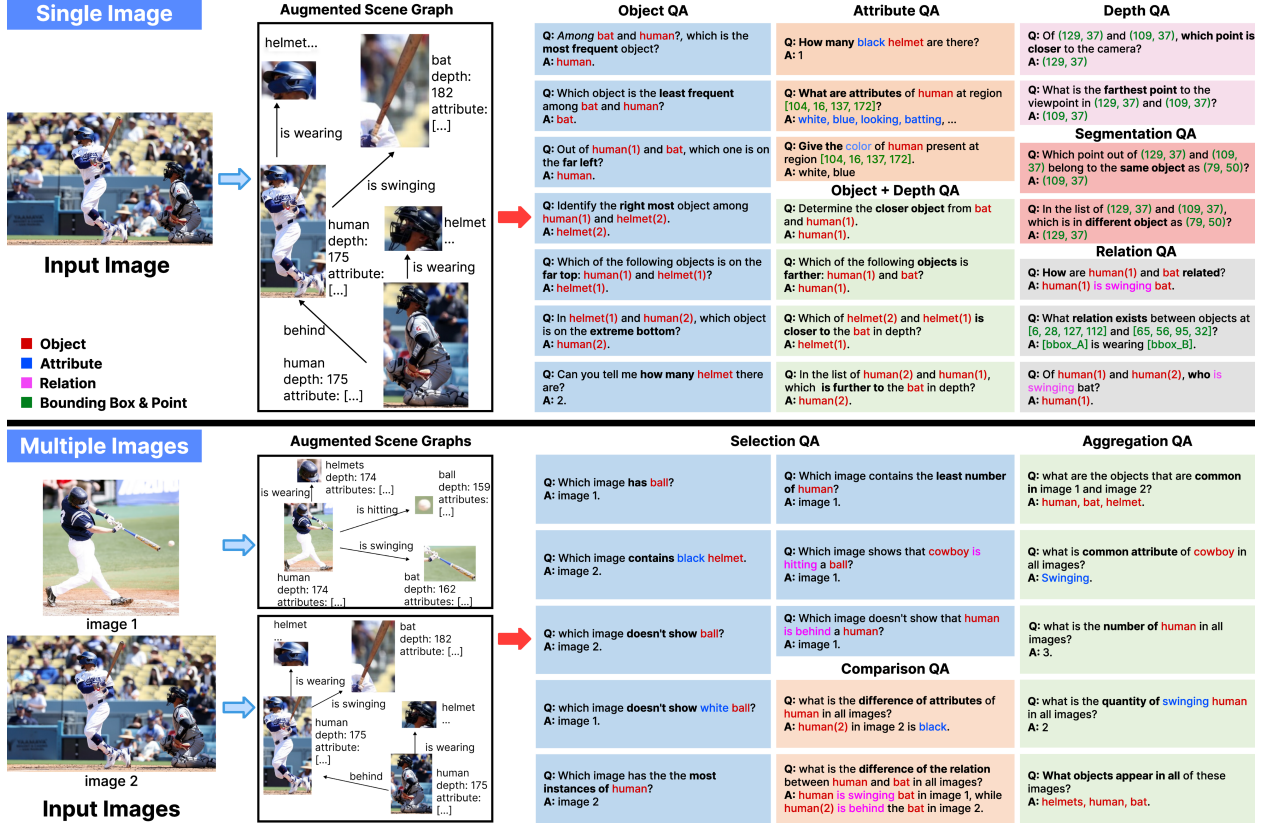


Figure 3.2: We visualize the instruction data generation process and generated examples for both single and multiple image scenarios. Single-image instruction data can be categorized into six dimensions, aiming to improve the model’s ability in retrieving and understanding the annotations. We divide multiple-image instruction data into three categories that help the model understand the relation between different images from the perspective of the relation between their scene graph.

grounded in visual content. Existing datasets such as LLaVA-Instruct [Liu et al., 2024a], InstructBLIP [Dai et al., 2024], and Mantis-Instruct [Jiang et al., 2024] have driven substantial progress, but typically provide only sparse coverage of each image’s visual content. Multi-image instruction datasets remain especially scarce. PROVISION addresses both gaps by generating exhaustive single-image and multi-image instruction data from scene graphs, producing millions of question–answer pairs that systematically cover objects, attributes, spatial relations, depth, and segmentation.

3.3 ProVision: Rich Visual Instruction Data for Grounded Understanding

In this section, we first introduce how we generate vision-centric instruction data programmatically with scene graphs. Then, we present our scene graph generation pipeline that automatically generates a scene graph for any given image.

3.3.1 Generating vision-centric instructions programmatically

Augmented scene graph. We first describe the scene graph definition used throughout this work, which is an augmented version of the standard scene graph representation defined in Visual Genome [Krishna et al., 2017b], including additional depth and segmentation labels. Given an input image x with size (w, h) , which have N objects $\{i_1, \dots, i_N\}$ and each object i_j has a list of attribute a_{attr}^j . The augmented scene graph is $G = (V, E)$, where $V \subseteq \{i_1, \dots, i_N\}$, $E = \{(i_j, i_k, a_{\text{rel}}^{jk}) \mid i_j, i_k \in V\}$ and a_{rel}^{jk} is the relation between objects i_j and i_k . Each object i_j has its corresponding bounding box and label pair a_{det}^j , segmentation a_{seg}^j , and a list of attribute a_{attr}^j . Additionally, we add depth annotation a_{dep}^j as an augmented feature.

Single-image visual instruction data. We implement 24 single-image instruction data generators to transform an augmented scene graph into thousands of high-level perceptual question-answer pairs for each image. Each generator utilizes hundreds of pre-defined templates, which systematically integrate these annotations to produce diverse instruction data. These generators are crafted to cover the model’s ability to compare, retrieve, and reason about basic visual concepts of objects, attributes, and relations based on the detailed information encoded in each scene graph.

Multi-image visual instruction data. Beyond the single-image scenarios, we also conduct 14 instruction data generators for multi-image scenarios. While single-image generators focus on producing instruction data from individual scene graphs, multi-image generators are capable of taking multiple scene graphs as input to generate question-answer pairs that span across images. These multi-image generators enable more complex queries, such as selection (e.g., "Which image contains more red objects?"), comparison (e.g., "What are the objects common in these images?"), and aggregation (e.g., "How many red objects in total in these images?") questions. By leveraging multiple scene graphs simultaneously, these generators produce instruction data that encourages models to develop advanced cross-image reasoning skills.

Instruction template generation. Following [Wang et al., 2024b], we implement a programmatic template

generator for each instruction data generator to improve the structural diversity of the language of the generated instruction data. We summarize the template amounts for each generator in the Appendix. We visualize some example instruction data that our data generators can produce in Figure 3.2.

Note that while we provide a diverse set of instruction data generators in this work, our system is designed to be highly versatile and readily extendable. It allows users to program additional data generators, expanding the space of instruction data that can be generated. By adding new templates or customizing existing ones, users can introduce novel question-answer pairs, explore new types of visual reasoning tasks, and adapt the system to meet evolving needs. This flexibility makes our framework a scalable tool for creating diverse, high-quality instruction data across various multimodal applications.

3.3.2 Generating scene graph for any image

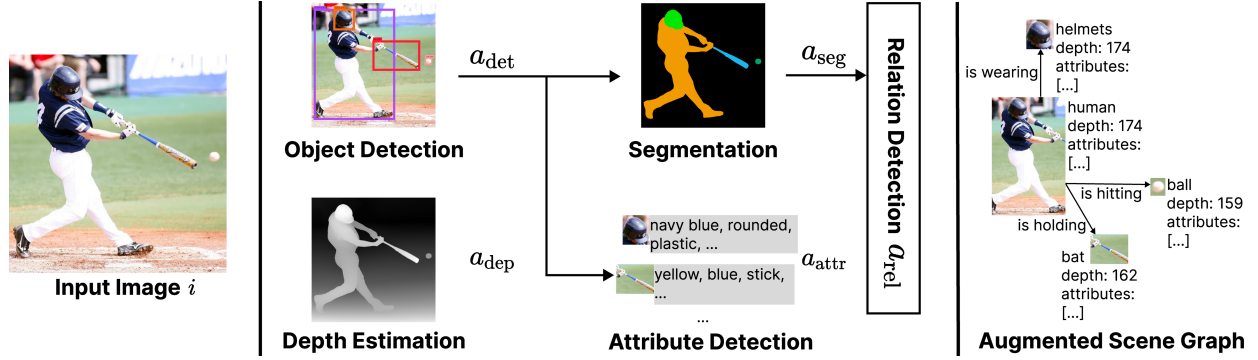


Figure 3.3: Our scene graph generation pipeline. For each input image, it will generate five annotations, including object, depth, segmentation, attribute, and relation, collectively forming an augmented scene graph.

We generate a scene graph with object detection, image segmentation, attribute generation, relation generation, and depth estimation. While we utilize state-of-the-art, openly accessible models for each module, our approach is not limited to these specific models. We provide an overview of the scene graph generation pipeline in Figure 3.3.

Object detection. We start with object detection to seek the bounding boxes and labels for further annotation methods. The object detection model $f_{\text{det}}(x)$ will annotate all bounding boxes and the corresponding labels of all objects. For example, for object j , the object detection method will output $a_{\text{det}}^j = ([x_{\min}^j, y_{\min}^j, x_{\max}^j, y_{\max}^j], l_j)$, where $(x_{\min}^j, y_{\min}^j)$ denotes the left bottom point of the bounding box and l_j denotes the label for object j . In this work, we adopt YOLO-world [Cheng et al., 2024b] as our object

detection model $f_{\text{det}}(x)$.

Image segmentation. We then adopt image segmentation for better object representations. The image segmentation model $f_{\text{seg}}(x, a_{\text{det}})$, SAM-2 [Ravi et al., 2025] in this work, takes the image x and bounding boxes a_{det} from object detection as input. Specifically, the segmentation will draw the pixel-wise segmentation $a_{\text{seg}} \in \mathbb{R}^{w \times h}$ according to a_{det} .

Attribute detection. Inspired by prior work [Zhu et al., 2024], we finetune vision-language models, i.e., CoCa [Yu et al., 2022] and LLaVA-1.5 [Liu et al., 2024a] as attribute detection models. We construct the training data from LSA, a large-scale attribute dataset [Pham et al., 2022]. We use its bounding box annotations to crop each object as a single image and use the corresponding attribute annotation as the target output. For LLaVA-1.5, we use "<image> {object_label}" as the prompt template for finetuning data construction. Based on our automatic and manual evaluations, LLaVA-1.5-13B is better than competitors with a precision of 90%, so we adopt it as our attribute detection model. More details of the evaluation and implementation can be found in Appendix.

Relation Detection. We finally retrieve the relations a_{rel}^{jk} for all pairs of objects i_j and i_k in image x according to their segmentation. To achieve this, we pick an finetuned Osprey model [Park et al., 2025] as $f_{\text{rel}}(x, a_{\text{seg}}^j, a_{\text{seg}}^k)$, which takes the whole image and segmentation of objects i_j and i_k as input, and generate a relation $\tilde{a}_{\text{rel}}^{jk}$. We then ground the generated relation by comparing the similarity between $\tilde{a}_{\text{rel}}^{jk}$ and our relation library and select the top-1 result as the a_{rel}^{jk} .

Depth estimation. Our augmented scene graph also included the pixel-wise depth annotation a_{dep} generated by a depth estimator $f_{\text{dep}}(x)$. In this work, we use Depth Anything V2 [Yang et al., 2024] as our depth estimation model. The pixel-wise depth annotation can be used to infer the depth of objects for comparing depth among objects.

3.3.3 PROVISION-10M Dataset

Leveraging manually-annotated scene graph dataset. We first utilize Visual Genome [Krishna et al., 2017b], one large-scale manually-annotated scene graph dataset to construct our instruction data. Specifically, we augment each scene graph with depth and segmentation annotation using Depth Anything V2 and SAM-2. Then, we generate 1.5 million single-image instruction data (**VG-S**) and 4.2 million multi-image instruction

data (**VG-M**): For **VG-S**, we sample one instruction data per image and generator, while for **VG-M**, we generate 100,000 samples per generator.

Leveraging generated scene graph. Besides, we sample 120,000 high-resolution images with more than 5 objects from the DataComp dataset [Gadre et al., 2024], and use our scene graph generation pipeline as described in Sec. 3.3.2 to generate the augmented scene graph for each image. Based on these generated scene graphs, we follow the same process as above to generate 2.3 million single-image instruction data (**DC-S**) and 4.2 million multi-image instruction data (**DC-M**).

In total, these four splits add up to more than 10 million unique instruction data to form PROVISION-10M. For each generated instruction, we store both a multiple-choice and a short-answer version, ensuring flexibility in the types of questions available for model training.

3.4 Experiments

In this section, we first describe the instruction data we synthesize in this work, followed by the experimental setup, results, and analysis. We found that 1) Our synthesized instruction data can boost model performance, and those from manually-annotated scene graphs are usually better than their counterpart from model-generated scene graphs, 2) The data format (short answer vs. multiple choice) and data scale are important factors to consider for the best performance, and 3) While our data helps when incorporated in either the pre-training or fine-tuning stage, incorporating them in both stages achieves the best performance.

Augmentation vs. replacement. To evaluate the utility of our generated dataset, we adopt two settings: augmentation and replacement. Given a *base dataset* which is an existing dataset used to train MLMs, the augmentation means augmenting the base dataset with our data, while the replacement is to replace a random subset of the base dataset with our data. In particular, we experiment with different augmentation/replacement *ratios*. For example, assume the base data contains 100K samples, an augmentation ratio of 5% indicates including an additional 5K of our data in the training set, while a replacement ratio of 5% means replacing 5K samples of the base data with our data.

Multiple choice vs. short answer. We explore both multiple choice and short answer formats for our generated instruction data, testing three distinct configurations: (1) all data in multiple choice format (multiple choice), (2) all data in short answer format (short answer), and (3) a balanced mix of formats, with

half of the data in multiple choice and half in short answer (half-half). These settings allow us to assess the impact of each answer type on model performance, as well as the versatility of the generated data in supporting different response styles.

Model and training recipe. We use LLaVA-1.5 [Liu et al., 2024a] instruction data as the base dataset and its training recipe for instruction tuning LLaVA-1.5-7B model with single-image instruction data; similarly, we follow Mantis [Jiang et al., 2024] for LoRA [Hu et al., 2022] instruction tuning Mantis-SigLIP-8B with multi-image instruction data and adopt Mantis-Instruct (excluding video-related subsets) as the base dataset. In addition, we experiment with adding our data to both the pre-training and fine-tuning stages of xGen-MM-4B model [Xue et al., 2024].

Benchmarks. We evaluate models on several popular MLM benchmarks including the following single-image benchmarks: CV-Bench (CVB) [Tong et al., 2024], SEED-Bench [Li et al., 2023b,a], MMBench (MMB) [Liu et al., 2023c], MME [Fu et al., 2024a], QBench2 [Wu et al., 2024], MMMU [Yue et al., 2024], RealWorldQA [Team, 2024b], MMStar [Chen et al., 2024b], MMVet [Yu et al.]; and multi-image benchmarks: Mantis-Eval [Jiang et al., 2024] and MMT-Bench (MMT) [Ying et al., 2024].

3.4.1 Instruction Tuning

We exhibit our experiment results on instruction tuning by answering the following questions: (1) do scene graphs help produce applicable instructions, and (2) do they need to be real, or can they be automatically generated?

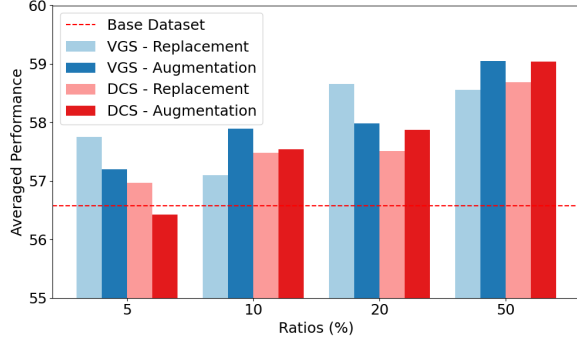
Do scene graphs help produce applicable instructions? This question can be answered affirmatively by Table 3.1 and Table 3.2. For single image instructions (Table 3.1), we compare the model trained with the base dataset and the models trained on four dataset augmentation/replacement ratios and three data formats across eight benchmarking datasets. The results illustrate that, for replacement, (1) instruction tuning the LLaVA-1.5-7B model with **VG-S** data yields improvements over the base dataset (LLaVA-1.5 instruction data) in averaged performance across all settings and achieves the best performance when the replacement ratio at 20%, and (2) on average, model performance is positively related to the amount of replaced multiple choice questions while negatively related to the replacement of short answers. For augmentation, we can see that (1) the model performance on all data formats increases with more data samples from **VG-S** and (2)

	Data Ratio*	Data Format	CVB-2D	CVB-3D	SEED	MMB	MME	QBench2	MMMU	RealWorldQA	Avg.
	LLaVA-1.5 instruction data		58.0	61.0	66.8	66.7	63.2	46.4	36.2	54.2	56.6
Replacement	5%	Short Answer	55.0	66.0	67.1	66.3	64.2	48.5	36.7	52.7	57.1
		Multiple Choice	61.0	61.0	67.5	67.0	63.3	47.8	37.8	54.6	57.5
		Half-Half	60.0	66.0	67.4	66.5	62.5	46.6	37.4	55.6	57.8
	10%	Short Answer	58.0	67.0	67.0	67.4	64.2	49.2	37.8	56.1	58.3
		Multiple Choice	56.0	62.0	67.2	67.1	64.4	48.4	36.8	58.7	57.6
		Half-Half	56.0	64.0	67.0	67.3	63.4	46.5	36.7	56.0	57.1
	20%	Short Answer	59.0	66.0	67.3	66.8	63.4	47.7	36.1	54.5	57.6
		Multiple Choice	57.0	63.0	66.8	68.0	63.2	48.9	37.2	58.6	57.8
		Half-Half	63.0	66.0	67.5	66.7	62.5	46.7	39.1	57.9	58.7
	50%	Short Answer	54.0	69.0	65.9	64.9	60.5	50.2	35.2	55.7	56.9
		Multiple Choice	61.0	68.0	66.3	66.1	61.8	46.1	38.2	56.7	58.0
		Half-Half	65.0	69.0	66.6	65.0	62.3	47.2	38.1	55.3	58.6
Augmentation	5%	Short Answer	55.0	65.0	66.7	66.5	63.9	48.1	37.3	55.6	57.2
		Multiple Choice	60.0	63.0	66.5	67.7	64.1	47.7	37.8	55.7	57.8
		Half-Half	56.0	66.0	66.8	67.1	61.6	46.5	37.6	56.1	57.2
	10%	Short Answer	59.0	69.0	66.7	66.9	62.3	49.0	37.3	54.0	58.0
		Multiple Choice	60.0	64.0	66.4	66.7	63.3	47.6	37.1	56.0	57.6
		Half-Half	57.0	67.0	68.0	68.1	64.2	45.3	38.4	55.0	57.9
	20%	Short Answer	59.0	68.0	67.2	68.0	61.6	48.5	37.6	54.0	58.0
		Multiple Choice	58.0	63.0	67.0	67.7	63.3	46.2	37.6	56.6	57.4
		Half-Half	58.0	67.0	67.7	67.2	63.0	46.7	36.4	57.9	58.0
	50%	Short Answer	57.0	69.0	67.2	68.0	64.5	49.4	37.2	55.3	58.5
		Multiple Choice	61.0	68.0	67.4	67.8	63.3	48.5	36.3	56.6	58.6
		Half-Half	60.0	66.0	67.5	67.6	66.0	48.7	38.9	57.6	59.0

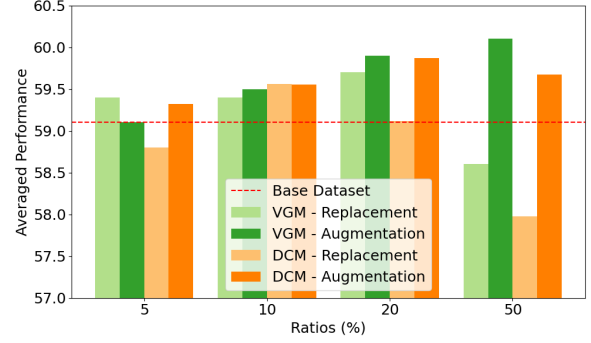
Table 3.1: Results of instruction tuning LLaVA-1.5-7B with **VG-S**, i.e. , single-image instruction data generated from the Visual Genome images and scene graphs. *: data ratio = number of our data added / size of LLaVA-1.5 instruction data.

compared with replacement, augmentation achieves better performance at the same level of data ratio. Results on single image suggest that mixing original data with scene graph-generated short answer and multiple choice format instruction yields competitive results when a substantial portion of the original data is replaced.

For multi-image instruction (Table 3.2), we test the models on two multi-image benchmarks and six single-image benchmarks. We can observe that for the 20% replacement ratio, the half-half format achieves the highest performance for both multi-image and single-image benchmarks, with an average score of 59.7, showing the benefit of mixing data formats. In contrast, at a 50% replacement ratio, the model’s performance generally decreases in both benchmarks, suggesting that excessive replacement with new data may reduce the model’s ability to generalize across tasks. For augmentation, multiple choice format stands out with a score of 60.0 on average at 20% of augmentation, while half-half at 50% augmentation achieves the highest average score of 60.1. This suggests that augmentation, especially with mixed data formats, can effectively enhance the model’s robustness. Interestingly, augmentation generally provides higher performance stability



(a) Results of instruction tuning LLaVA-1.5-7B with **DC-S**, i.e. , single-image instruction data generated from the DataComp images and model-generated scene graphs.



(b) Results of instruction tuning Mantis-SigLIP-8B with **DC-M**, i.e. , multi-image instruction data generated from the DataComp images and model-generated scene graphs.

across both multi-image and single-image benchmarks compared to replacement. On the other hand, half-half format with 10% augmentation and multiple choice format with 20% augmentation show strong performance across multi-image benchmarks (Mantis-Eval and MMT), showing the superiority of scene graph-generated instruction in helping the model learn how to select, compare, and aggregate features on different images.

Do the scene graphs need to be manually annotated, or can they be model generated?

We compare models trained on instruction data from manually annotated (**VG-S**, **VG-M**) and model-generated (**DC-S**, **DC-M**) scene graphs. According to Figure 3.4a (**VG-S** vs. **DC-S**), **DC-S** underperforms **VG-S** at lower data scales. Interestingly, as the data scale increases to a 50% ratio, **DC-S** achieves comparable performance to **VG-S**, suggesting that larger data scales help mitigate initial performance gaps between data from model-generated and human-curated scene graphs.

Moreover, from Figure 3.4b (**VG-M** vs. **DC-M**), we observe that as the ratio increases, the model performance on the replacement setup grows first and decays later. With the increase of replacement or augmentation ratio, **DC-M** underperforms with **VG-M**, suggesting that on multi-image settings, a larger scale on generated scene graphs may trigger edge effects and not always provide stable performance gain to the model training. In conclusion, instruction data from manually annotated scene graphs is in general better than that from model-generated scene graph, yet both data are able to boost model performance in most cases.

		Multi-image benchmark			Single-image benchmark						
	Data Ratio*	Data Format	Mantis-Eval	MMT	SEED	MMB	MME	QBench2	MMMU	RealWorldQA	Avg.
Mantis instruction data			54.4	52.9	68.1	72.8	58.5	70.1	44.3	51.5	59.1
Replacement	5%	Short Answer	59.9	52.5	68.4	73.3	60.0	70.3	41.4	50.3	59.5
		Multiple Choice	57.6	52.8	68.0	73.0	58.4	70.5	43.7	52.3	59.5
		Half-Half	57.6	53.4	68.1	71.7	57.8	71.5	44.4	50.9	59.4
	10%	Short Answer	69.0	54.8	68.6	73.7	57.4	68.4	41.3	50.7	59.2
		Multiple Choice	59.9	53.1	68.3	71.7	57.5	68.6	45.3	51.9	59.5
		Half-Half	59.0	53.7	68.2	72.6	58.6	68.9	43.1	51.5	59.4
	20%	Short Answer	62.7	52.9	68.2	72.9	56.6	70.2	45.4	49.7	59.8
		Multiple Choice	57.6	52.2	68.0	72.9	57.7	67.4	42.0	51.4	58.6
		Half-Half	58.5	58.6	68.7	72.2	59.6	69.4	44.1	51.8	59.7
	50%	Short Answer	57.1	53.2	67.4	70.4	58.6	65.2	42.2	51.2	58.2
		Multiple Choice	55.8	52.5	67.5	69.8	57.5	67.9	42.6	53.5	58.4
		Half-Half	54.8	54.0	67.9	72.1	58.2	66.8	43.7	51.5	58.6
Augmentation	5%	Short Answer	60.4	53.8	68.3	71.2	58.9	70.6	44.3	48.9	59.5
		Multiple Choice	58.1	54.0	68.1	71.7	58.8	70.3	42.4	50.2	59.2
		Half-Half	58.1	52.5	68.0	71.8	58.4	70.1	41.3	52.9	59.1
	10%	Short Answer	60.4	53.0	68.1	72.8	59.2	71.4	44.2	50.6	60.0
		Multiple Choice	60.4	52.7	68.0	72.2	59.2	71.1	43.1	50.2	59.6
		Half-Half	61.3	53.0	68.0	72.9	60.0	67.7	42.0	51.0	59.5
	20%	Short Answer	57.1	53.2	68.5	72.3	60.3	71.6	43.8	50.3	59.6
		Multiple Choice	58.5	52.9	68.6	72.1	60.6	71.6	43.0	51.5	60.0
		Half-Half	60.4	52.8	68.5	72.4	60.6	68.4	43.7	52.7	59.9
	50%	Short Answer	57.6	53.0	68.1	71.6	59.1	70.4	44.3	51.2	59.4
		Multiple Choice	58.5	54.1	68.4	72.4	58.8	70.1	41.4	51.6	59.4
		Half-Half	60.4	53.7	68.1	73.4	60.4	69.7	43.8	51.6	60.1

Table 3.2: Results of instruction tuning Mantis-SigLIP-8B with **VG-M**, i.e. , multi-image instruction data generated from the Visual Genome images and scene graphs. *: data ratio = number of our data added / size of Mantis instruction data.

3.4.2 Pre-training vs. Instruction Tuning

To assess the benefits of incorporating our data at scale during the pre-training stage, and to compare the effects of adding our data in the pre-training versus fine-tuning stages, we adopt the xGen-MM (BLIP-3) [Xue et al., 2024] training methodology and use its pre-training data recipe as a foundation. We establish a baseline by pre-training a model on approximately 10 billion tokens using the xGen-MM (BLIP-3) [Xue et al., 2024] pre-training recipe without our data. Similarly, we apply a baseline fine-tuning recipe of 1 million samples that excludes our data. Details for both recipes are provided in the Appendix.

In the data augmentation setup, we create recipes that incorporate our data into both the pre-training and fine-tuning stages. **At each stage, we ensure that our additional data accounts for around 5% of the final dataset size.** To examine the impact of different types of scene graph data input on our single-image instruction data generation pipeline, we conduct two sets of experiments: one using human-annotated scene graphs (**VG-S**) as the data source, and another using synthesized scene graphs generated by our pipeline

Augment in Pre-train	Augment in Fine-tune	CVB-2D	CVB-3D	SEED	MMB	MMStar	MME	QBench2	MMVet	MMMUS	RealWorldQA	TextVQA	Avg.
X	X	62.9	73.5	70.1	73.9	44.1	62.1	53.9	38.2	41.6	56.2	66.5	58.5
DC-S: DataComp images and model-generated synthetic scene graphs													
✓	✓	64.5	71.9	69.1	73.3	44.8	60.9	57.9	35.2	44.1	59.7	67.0	58.9
✓	✓	67.9	70.8	70.4	73.5	45.5	64.4	53.6	37.7	46.1	58.8	67.1	59.6
✓	✓	68.3	75.5	69.6	74.5	44.4	62.4	54.0	38.5	41.9	61.6	66.5	59.7
VG-S: Visual Genome images and scene graphs													
✓	✓	67.7	72.2	70.0	73.6	45.5	64.2	54.4	36.7	42.9	60.4	67.3	59.5
✓	✓	65.9	73.3	70.5	75.9	44.8	64.9	56.3	36.8	42.2	59.2	67.3	59.7
✓	✓	70.2	73.7	69.9	73.9	44.3	64.0	55.7	40.4	44.8	57.4	66.8	60.1

Table 3.3: Comparison of augmenting the base data recipe with our data (both **DC-S** and **VG-S**) in pre-training, fine-tuning, or both. The first row represents the baseline model’s performance, which uses only the base recipes for both stages without our data. The results demonstrate that augmenting the base data recipes with our data in either stage (pre-training or fine-tuning) improves the model’s average performance across 11 benchmarks, with augmentation in both stages generally achieving the highest average performance. When augmented with 1.5 million samples from **VG-S**, our model shows an average improvement of approximately 1.4% over the baseline across all benchmarks. Bold values indicate the highest score for each benchmark.

(**DC-S**). Results in Table 3.3 reveal several insights:

1. **Performance gains with augmenting our data in the base recipes:** Both **DC-S** and **VG-S** augmentations consistently improve model performance across the 11 evaluated benchmarks compared to the baseline dataset, which does not include our data. This validates the efficacy of incorporating additional vision-centric knowledge into multimodal foundation model training.
2. **Synergistic effect of dual-stage augmentation:** The results suggest that augmentation during both pre-training and fine-tuning stages synergistically enhances the performance. For instance, models trained on datasets augmented in both stages achieve the highest average scores (e.g., **DC-S** with +1.2% and **VG-S** with +1.6%), indicating that the synergized effect of dual-stage augmentation outperforms augmentation at either stage alone.
3. **Comparative effectiveness of VG-S and DC-S:** While both types of data yield improvements, **VG-S**-based augmentation generally provides a marginally higher average score (60.1%) compared to **DC-S** (59.7%) when augmented in both stages. This suggests that a cleaner scene graph data source may enable our pipeline to yield even higher performance.

A key advantage of our pipeline is its scalability, enabling its application in large-scale multimodal language model training. We conduct experiments comparing the addition of 0.75 million versus 1.5 million

Dataset Augmentation Scale	Avg.
No Augmentation	58.5
0.75 Million	59.1
1.5 Million	60.1

Table 3.4: Impact of dataset augmentation scale on model performance. This table compares the model’s performance across the same 11 benchmarks, with varying scales of data augmentation during pre-training: no augmentation, 0.75 million samples, and 1.5 million samples from **VG-S**.

samples of **VG-S** during pre-training, while keeping the fine-tuning recipe consistent. As shown in Table 3.4, scaling the inclusion of **VG-S** in pre-training stage from 0.75 million to 1.5 million samples yields notable gains in average performance across 12 benchmarks, increasing from 61.4% to 62.3%. This result underscores the potential of our data to enhance model capabilities in large-scale MLLM training.

3.5 Conclusion

This chapter demonstrated that grounded visual understanding can be substantially improved through richer training supervision. By leveraging scene graphs as a structured intermediary, **PROVISION** generates exhaustive, programmatic instruction data that covers objects, attributes, relations, segmentation, and depth—aspects that are typically under-supervised in existing datasets. Our experiments show that this data consistently improves model performance across both single-image and multi-image benchmarks, and that the benefits hold across pre-training and instruction tuning stages.

However, even with improved supervision, models still process visual inputs as uniform grids of patches that are agnostic to the underlying scene structure. The next chapter addresses this representational limitation by proposing object-centric visual tokenization—organizing visual tokens around trajectories and objects rather than fixed spatial grids—to more directly align model representations with the structure of the visual world.

Chapter 4

Realizing Grounded Visual Understanding through Grounded Visual Tokenization

4.1 Introduction

The previous chapters addressed grounded visual understanding from the perspectives of evaluation and training data. This chapter tackles a complementary dimension: the *visual representation* itself. We argue that grounded understanding is fundamentally easier to achieve when the model’s input representation is organized around objects and their temporal trajectories, rather than a uniform grid of patches that fragments coherent entities and introduces massive redundancy.

Understanding the world through vision and language increasingly relies on large multimodal models whose performance is fundamentally determined by how visual input is tokenized before interacting with lan-

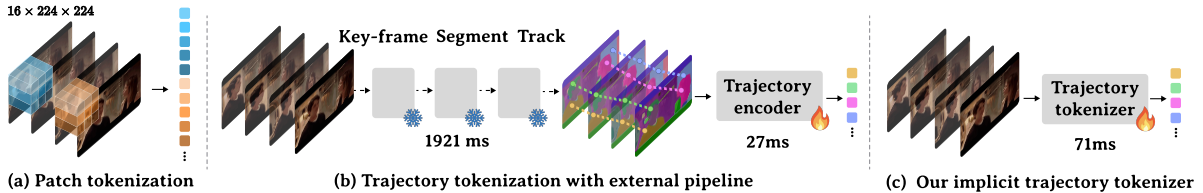


Figure 4.1: (a) Traditional video tokenization splits a video into space-time patches, introducing large number of redundant tokens. (b) We propose to represent a video via panoptic sub-object trajectory (TrajViT), which significantly reduces redundancy but relies on slow, non-differentiable pipelines. (c) We further propose TrajTok, an end-to-end differentiable trajectory tokenizer that learns to implicitly propose trajectory tokens, offering low token counts, efficiency and adaptability to downstream objectives.

guage. As vision–language systems scale in capability and generality, the design of this visual interface—how continuous visual streams are transformed into discrete, language-compatible tokens—has become central to grounding language in pixels. Most current systems, including those operating on video [Neimark et al., 2021; Lin et al., 2022], still rely on patch-based tokenization inherited from early transformer architectures [Vaswani et al., 2017]. Frames (or images) are divided into fixed space–time patches [Neimark et al., 2021], producing a uniform grid of tokens regardless of semantic structure. When applied to video, this paradigm simply extends across time (Figure 4.1a), meaning longer videos yield more tokens in proportion to frame count rather than visual or semantic complexity. This design has several drawbacks for grounding: (1) it creates an interface in which semantically unrelated pixels may be grouped while semantically coherent entities are fragmented across many tokens; (2) it treats all sequences equally, ignoring the difference between, e.g., a static office scene and a dynamic football match; (3) it burdens the transformer with the task of reassembling disjoint patch tokens into coherent perceptual units (e.g., the same object across time). Although recent token-merging approaches attempt to reduce redundancy, they remain largely content-agnostic [Ryoo et al., 2021; Bolya et al., 2023; Arnab et al., 2021] or break down under camera motion [Choudhury et al., 2025b]. These issues are not merely matters of computational cost—they reflect a deeper misalignment between how humans perceive visual scenes and how current models represent them for downstream language reasoning.

To address this, we propose a new paradigm of *grounded video tokenization*. Instead of naively splitting videos into patches, we align tokenization with the fundamental perceptual principles governing object perception and motion [Wagemans et al., 2012; Spelke, 1990; Pylyshyn and Storm, 1988] by organizing tokens to correspond to panoptic sub-object trajectories. Rooted in Spelke’s core cognitive principles [Spelke, 1990] and the Gestalt Principle of *common fate*, we treat object parts as coherent entities that persist over time (Figure 4.1b). By correlating token count with the number of distinct trajectories—rather than frame count—our method directly reflects the complexity of a scene in terms of its constituent objects and their interactions, enabling efficient processing of long sequences. Moreover, because our tokenization divides a video into objects rather than raw pixel divisions, it remains robust to variations in lighting, occlusions, and camera movements—a persistent challenge for patch-based tokenizers [Buch et al., 2022; Lei et al., 2022, 2021].

We operationalize this paradigm in two stages. First, we design **TrajViT**, a video encoder that implements

grounded video tokenization using an external segmentation-and-tracking pipeline coupled with a standard transformer encoder (Section 4.3.1; Figure 4.2). Given a video, TrajViT extracts panoptic trajectories for all object parts, converts each into a fixed-size token, and processes all tokens with a transformer. This approach significantly reduces the number of tokens—by an average of $10\times$ in the Panda dataset [Chen et al., 2024d]—compared to space-time patch tokenization, organizes them into semantically meaningful units with no temporal redundancy, and makes the tokenization invariant to low-level visual variations in lighting and camera motion.

While TrajViT demonstrates the strong potential of trajectory-based tokenization, it is fundamentally limited by its reliance on external, task-agnostic segmentation and tracking models [et al., 2024; Ravi et al., 2025]. This makes the tokenizer a slow, non-differentiable preprocessing step that fixes the semantic granularity of tokens to that of general-purpose segmentation models—which may not be optimal for the downstream task. For instance, understanding a particular dance performance might require tokens representing individual body parts, whereas identifying group formations might benefit from representing each dancer as a single token.

This motivates the second stage: **TrajTok**, an end-to-end differentiable video tokenizer that learns to propose implicit trajectory tokens by formulating trajectory generation as a clustering problem over input pixels in both space and time (Section 4.3.2; Figure 4.1c). TrajTok is more efficient than TrajViT and adapts its tokenization granularity to downstream tasks, trading off pixel-perfect segmentation accuracy for end-task performance. Our trajectory encoder further incorporates adaptive representation inspired by Matryoshka [Kusupati et al., 2022], enabling a flexible number of tokens per trajectory and resolving over-compressed representations for objects undergoing complex motion.

Together, these two systems demonstrate that object-centric tokenization consistently outperforms patch-based approaches across retrieval, classification, and video question answering. Using TrajTok, we train TrajViT2 from scratch under the CLIP objective [Radford et al., 2021] and achieve improvements of +4.8% on Kinetics-400 and +4.1% on SSv2 over a standard video ViT, while matching inference FLOPs of state-of-the-art token-merging methods [Bolya et al., 2023; Choudhury et al., 2025b,a]. We further show that TrajTok is versatile beyond pretraining: as **TrajAdapter**, it enhances probing performance of pretrained encoders without full fine-tuning; as the connector in **TrajVLM**, it improves video question-answering performance,

especially on long-video benchmarks.

4.2 Related Work

Video tokenization and vision transformers. The dominant paradigm for video representation in transformers is space–time patch tokenization, where a video is divided into fixed-size 3D patches that serve as input tokens [Arnab et al., 2021; Neimark et al., 2021; Bertasius et al., 2021]. While straightforward, this approach produces a token count that scales linearly with frame number regardless of scene complexity. Architectures such as ViViT [Arnab et al., 2021] and TimeSformer [Bertasius et al., 2021] mitigate the quadratic cost of self-attention through factorized spatial and temporal attention, but the underlying tokenization remains a uniform grid that fragments objects across many tokens. Our work departs from this paradigm by proposing tokenization aligned with semantic structure rather than spatial geometry.

Token merging and reduction. A growing body of work seeks to reduce the number of tokens processed by vision transformers to improve efficiency. TokenLearner [Ryoo et al., 2021] learns to select a compact set of informative tokens via spatial attention pooling. Token Merging (ToMe) [Bolya et al., 2023] progressively combines similar tokens during the forward pass based on feature similarity. Run-Length Tokenization (RLT) [Choudhury et al., 2025b] merges temporally redundant patches. Adaptive Patch Tokenization (APT) [Choudhury et al., 2025a] dynamically adjusts patch granularity. While these methods successfully reduce token counts, they operate on patches that are themselves content-agnostic—the merging decisions are post-hoc corrections to a fundamentally misaligned tokenization. By contrast, our approach produces semantically grounded tokens from the outset, making downstream merging unnecessary.

Object-centric video representations. Object-centric approaches aim to decompose visual scenes into representations corresponding to individual entities. Slot Attention [Locatello et al., 2020] and its video extensions [Kipf et al., 2022; Elsayed et al., 2022] learn to bind object slots through iterative attention, though typically on synthetic or simple scenes. SAVi++ [Elsayed et al., 2022] extends this to real-world video with depth supervision. In the video understanding literature, object-centric features have been explored for action recognition [Herzig et al., 2022; Wu et al., 2019] and video question answering [Le et al., 2020]. Our work shares the motivation of organizing representations around objects but differs in that we formulate object discovery as a tokenization step that directly interfaces with standard transformer encoders and scales to

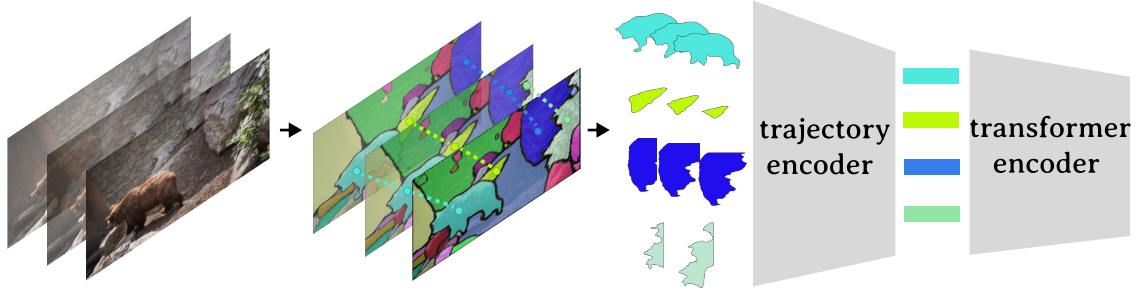


Figure 4.2: Overview of TrajViT. Given a video, we first panoptically extract the trajectories for all objects. Our trajectory encoder converts these dynamic object trajectories into fixed sized embeddings, which serve as the input to the transformer encoder.

real-world video benchmarks.

Segmentation and tracking for video understanding. Modern segmentation and tracking models [Ravi et al., 2025; Kirillov et al., 2023; Cheng and Schwing, 2022; Yang et al., 2023] have reached impressive accuracy on pixel-level benchmarks, but their use as a preprocessing step for video understanding remains underexplored. Prior work has used off-the-shelf detectors to extract region features for downstream tasks [Anderson et al., 2018; Zhang et al., 2021], primarily in the image domain. Our TrajViT extends this idea to video by extracting full panoptic sub-object trajectories and encoding them as tokens. TrajTok then internalizes this pipeline, replacing external segmentation with a lightweight learned module that jointly optimizes grouping and downstream performance—bridging the gap between pixel-level segmentation and high-level video understanding.

4.3 Grounded Visual Tokenization

4.3.1 Grounded Visual Tokenization with External Vision Models

Our first approach, TrajViT, aims to represent videos using semantically meaningful and compact tokens derived from panoptic sub-object trajectories. As illustrated in Figure 4.2, we panoptically extract trajectories for all objects in a video and convert them into fixed-size embeddings to be used as the transformer encoder input. We find that trajectory extraction and encoding each pose distinct challenges. First, existing tracking solutions, which propagate segmentation masks from frame to frame, cannot handle newly appearing objects mid-video and are prone to long propagation time [Ravi et al., 2025; Yang et al., 2023; Cheng et al., 2024a],

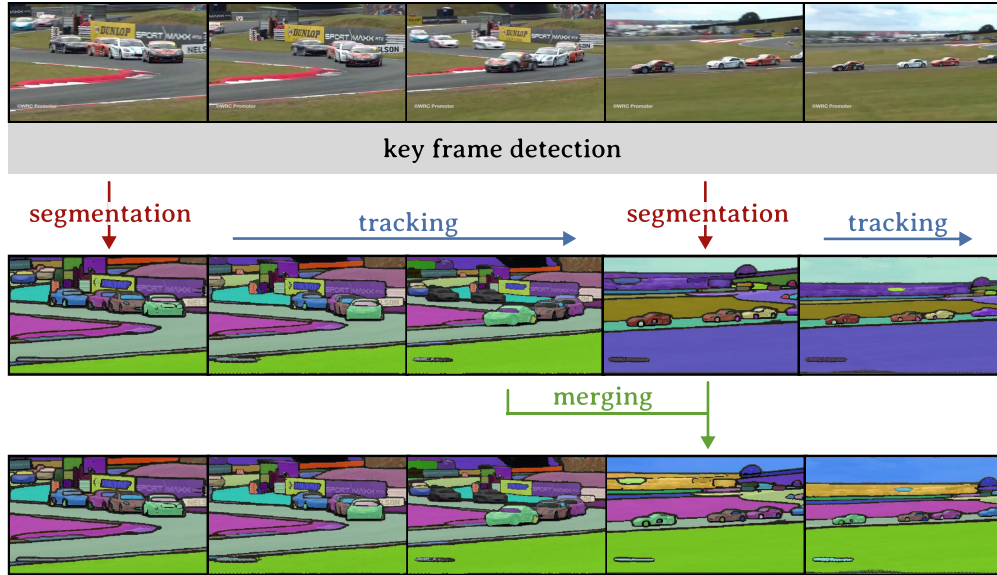


Figure 4.3: Our parallel trajectory generation pipeline. We use key frame detection to break a video into subclips. We segment and track objects in each clip in parallel and finally merge objects between clips. This paradigm captures objects that emerge over time while reducing overall tracking latency.

necessitating a new trajectory generation pipeline. Second, the trajectories we generated can contain a varying number of pixels and appear in a varying number of frames, but the final representations need to be fixed-size embeddings to be compatible with the transformer architecture, requiring a new trajectory encoding module. We detail how we tackle these challenges in this section.



Figure 4.4: Visualizations of generated trajectories.

Panoptic Sub-object Trajectory Extraction

Our objective is to accurately segment and track all objects in a video—including backgrounds—while keeping the process efficient. We design a pipeline that couples an off-the-shelf segmentation and tracking model, can capture objects that emerge over time, and can be parallelized to reduce latency.

Discovering objects by segmenting key frames. Given a video of length T with time index $t \in \{1, \dots, T\}$, we begin by selecting key frames $\{\hat{t}_i\}_{i=1}^K$ where new objects are likely to appear. Specifically, we compare the color histogram and luminance intensity between consecutive frames and flag those exceeding a manual threshold (details in supplementary). Then, we panoptically segment each key frame to capture all objects. We employ the DirectSAM model [Chen et al., 2024a] as it is lightweight with 84M parameters and can produce fine-grained segments in one forward pass with consistent granularity across images, reducing the risk of merging failure in the later merging stage. We denote the segmentation mask derived in key frames as $\{M_{\hat{t}_i}\}_{i=1}^K$.

Tracking objects within each clip. The detected key frames naturally partition the video into multiple clips. Within each clip $\hat{t}_i \leq t < \hat{t}_{i+1}$, we employ an off-the-shelf tracker (SAM2 [Ravi et al., 2025]) to propagate the segmentation mask $M_{\hat{t}_i}$ from the key frame $\hat{t}_i$ to subsequent ones. This per-clip process can be executed in parallel for a batch of clips for efficiency (Figure 4.3). In practice, we additionally split a clip at the midpoint if its length exceeds a certain threshold to shorten the propagation time. We use hiera-small variants of SAM2 with 46M parameters.

Merging objects between consecutive clips. Finally, we link the identities of objects between consecutive clips. Specifically, we perform an additional tracking step from the final frame of a clip ($\hat{t}_{i+1} - 1$) to the first frame of the subsequent clip ($\hat{t}_{i+1}$), generating a propagated mask $M'_{\hat{t}_{i+1}}$. We then compare each segment in $M'_{\hat{t}_{i+1}}$ with those in the original key frame mask $M_{\hat{t}_{i+1}}$. If the Intersection over Union (IoU) for any pair of segments exceeds a threshold of 0.8, we consider them to represent the same object and merge their trajectories.

This split-track-merge pipeline efficiently discovers and tracks objects across entire videos. The visualizations of generated trajectories are shown in supplementary. While we observe occasional inconsistent tracking, i.e., segments of the same object in consecutive clips being tagged as different trajectories due to imperfect merging, it seldomly hurts model performance because we find that these inconsistencies often

coincide with situations where the object undergoes a large appearance change. In such cases, creating a new trajectory might benefit downstream tasks, as it accounts for significant changes in objects.

Video Encoding with Trajectory Tokens

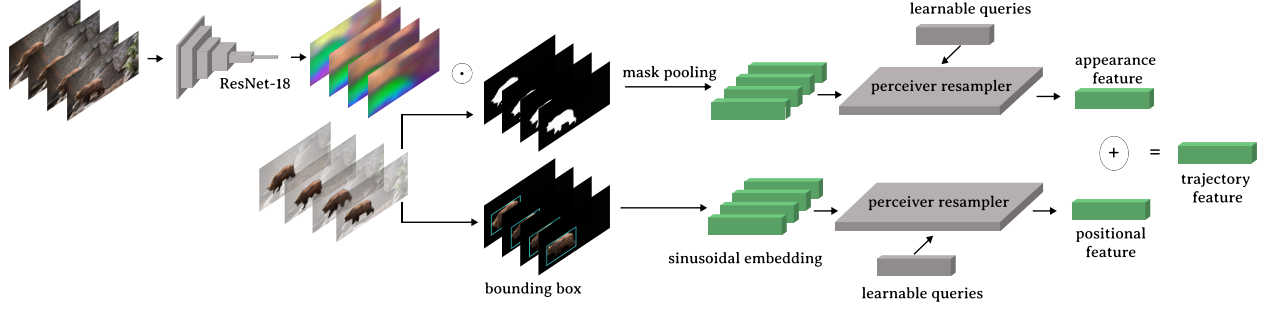


Figure 4.5: Architecture of trajectory encoder. we employ a two-branch design that encodes a trajectory’s appearance and temporal position separately. At each frame, we represent the appearance of a segment by mask pooling its feature, and represent its position by bounding box coordinates. Both features are then aggregated across frames via perceiver resampler and added together to form the trajectory feature.

We aim to obtain fixed-size representations of trajectories and utilize them as input for a transformer model. To achieve this, we develop a *trajectory encoder* that produces an appearance embedding and a temporal position embedding for each trajectory (Figure 4.5).

Trajectory appearance encoding. For trajectory appearance encoding, we first extract hierarchical visual features from each frame. Given a video consisting of frames $\{I_t\}_{t=1}^T$, we apply a lightweight convolutional network [He et al., 2016] to obtain corresponding hierarchical feature maps $\{F_t\}_{t=1}^T$ for each frame. Then, for each trajectory, we extract its appearance feature via masked feature pooling. Specifically, for trajectory i at frame t , we compute:

$$f_t^i = \frac{\sum_{x,y} M_t^i(x,y) \cdot F_t(x,y)}{\sum_{x,y} M_t^i(x,y) + \epsilon} \quad (4.1)$$

where x, y denotes the spatial position, M_t^i is the trajectory’s segmentation mask, F_t is the feature map, and ϵ is a small constant. This operation pools the feature inside the region-of-interest for each trajectory. The resulting f_t^i represents the trajectory i ’s feature at frame t . Since a trajectory can span a varying number of frames, we need a mechanism to summarize a dynamic sequence of the per-frame features into a fixed-size embedding. We choose to use one layer of perceiver resampler module [Alayrac et al., 2022], which is

a lightweight cross-attention layer that learns a predefined number of latent input queries to attend to the sequence of per-frame features. As we set the number of latent queries to be one, the perceiver resampler outputs a single embedding that serves as the appearance representation of the trajectory. We use rotary embedding [Su et al., 2024] as the positional embedding of perceiver resampler module for better test-time frame number generalization.

Trajectory temporal position encoding. To represent the temporal position information of each trajectory, we extract bounding box coordinates $c_t^i = (x_1^i, y_1^i, x_2^i, y_2^i)$ at each frame of the trajectory i . These coordinates are then mapped to high-dimensional vectors using sinusoidal embedding [Mildenhall et al., 2021]. Since the bounding box sequence for a trajectory is also of variable length, we again employ the perceiver resampler module to summarize the sequence into a single temporal position embedding.

Finally, we construct the trajectory embedding by summing the appearance embedding and the temporal position embedding, which serves as a single token of the input to the transformer. The trajectory encoder has 20M parameters, much smaller than the transformer encoder (e.g., 304M for ViT-Large), yet can significantly reduce the number of tokens for efficient video processing, without dropping information in raw pixels.

Model Training. Our final video encoder **TrajViT** consists of this trajectory encoder followed by a standard transformer module. We train it with the CLIP objective [Radford et al., 2021]. Given a dataset of video-caption pairs, CLIP objective learn video and text representations by jointly training a video encoder and a text encoder with contrastive loss.

Seamless Integration with Image Data

Incorporating image data is a common practice when training video encoders, as the size of publicly available image data is several orders of magnitude larger than that of video data. TrajViT can integrate image data naturally without architectural modifications. Given an image, we apply the same segmentation process as in videos, and then treat each segmented region as a trajectory of length one. This enables seamless integration of image data into video pretraining. In contrast, models that use space-time patch tokenization cannot directly process image data because of their reliance on 3D convolutions or MLP layers that map space-time patches to fixed-size tokens. These modules require channel dimensions that match the spatial-temporal patch size, preventing direct image processing. As a result, space-time patch based approaches typically pretrain

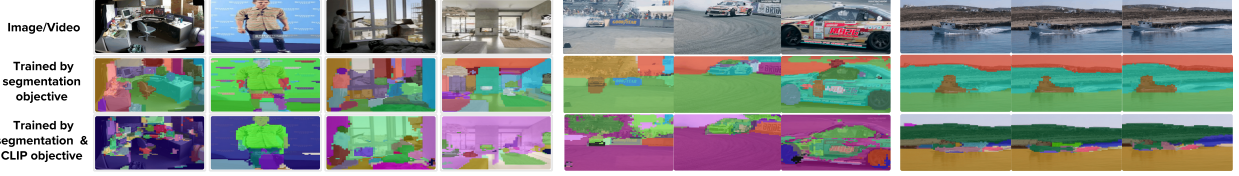


Figure 4.6: Training with downstream understanding tasks reshapes the segmentation granularity. We visualize the trajectory masks produced by our segmenter when trained with only segmentation supervision versus jointly with segmentation and CLIP objectives. The CLIP objective reshapes the segmentation granularity, producing finer foreground object masks while merging background regions.

ViT2D on images, transfer the weights to ViT3D, and then fine-tune on video data, while ours allows joint training on images and videos in one stage.

4.3.2 End-to-End Grounded Visual Tokenization

While TrajViT demonstrates that trajectory-based tokenization substantially improves video understanding, its reliance on an external segmentation-and-tracking pipeline introduces several practical limitations: it is slow, non-differentiable, and fixes the semantic granularity of tokens to that of general-purpose segmentation models. In this section, we present TrajTok, an end-to-end differentiable trajectory tokenizer that replaces the external pipeline with a lightweight, jointly trained module. TrajTok learns to implicitly propose trajectory tokens by formulating trajectory generation as a clustering problem over input pixels in both space and time.

Let $\mathbf{V} \in \mathbb{R}^{T \times H \times W \times 3}$ denote an input video with T frames and spatial resolution $H \times W$. Our goal is to learn a mapping $\mathcal{T} : \mathbf{V} \rightarrow \mathbf{Z}$, where $\mathbf{Z} \in \mathbb{R}^{N \times d}$ is a set of N trajectory tokens with dimension d . N is not fixed and depends on semantic complexity of the video.

The tokenizer consists of two differentiable components, which were trained jointly: a *universal segmenter* that partitions the input into semantic groups, and a *trajectory encoder* that aggregates these groups into compact latent tokens. We visualize the architecture in Figure 4.7.

Universal Segmenter for Trajectory Grouping

The segmenter is a lightweight and efficient module that performs effective semantic grouping in a single feedforward pass. It decouples video duration from the final token count. We value robust semantic grouping over pixel-perfect segmentation masks, and design a simple and efficient module to achieve this.

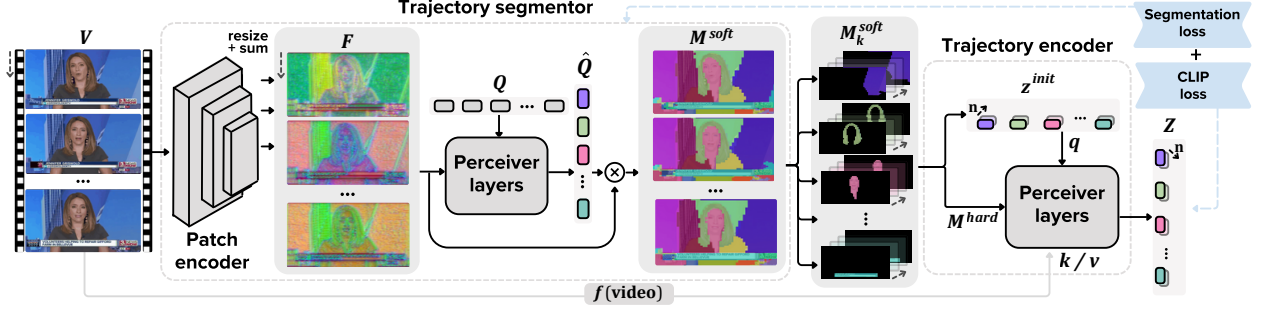


Figure 4.7: Overview of the TrajTok architecture. TrajTok comprises a *trajectory segmenter* and a *trajectory encoder*. The segmenter proposes trajectory masks for all objects in an image or video within a single forward pass. The encoder then aggregates raw video pixels or encoded visual features (parameterized by f in the figure) according to these masks to produce trajectory tokens. The number of tokens per trajectory can be flexibly adjusted based on the available compute budget.

Frame-wise feature extraction. We first extract a high-resolution feature map from V using a lightweight patch encoder. We use ConvNeXt [Liu et al., 2022] architecture as it naturally provides multi-scale feature maps. We extract features frame-wise. Multi-scale maps are resized to their highest resolution (1/4 of original image size) and summed to form the final dense feature representation $\mathbf{F} \in \mathbb{R}^{T \times h \times w \times d}$, where $h, w = H/4, W/4$.

Learnable queries for semantic grouping. We introduce a set of N_q learnable latent queries $\mathbf{Q} \in \mathbb{R}^{N_q \times d}$ that act as cluster prototypes. These queries are processed through a stack of Perceiver [Jaegle et al., 2021] layers. Within each perceiver layer, queries $\mathbf{Q}$ attend to the dense features $\mathbf{F}$ using cross-attention. To handle inputs with variable frame counts T and encode spatiotemporal structure, we apply 1D Rotary Positional Embeddings (RoPE) [Su et al., 2024] to the patch features $\mathbf{F}$ before attention. The resulting processed queries, $\hat{\mathbf{Q}} = \text{Perceiver}(\mathbf{Q}, \text{RoPE}(\mathbf{F}))$, encapsulate the semantic information necessary for segmentation.

Soft segmentation. We generate segmentation masks by computing the similarity between processed queries and patch features. A soft segmentation map $\mathbf{M}^{soft} \in [0, 1]^{N_q \times T \times h \times w}$ is obtained via softmax over the query dimension of the dot-product similarity:

$$\mathbf{M}_{k,t,i,j}^{soft} = \text{softmax}_k(\hat{\mathbf{q}}_k \cdot \mathbf{F}_{t,i,j}) \quad (4.2)$$

where $\hat{\mathbf{q}}_k$ is the k -th processed query and $\mathbf{F}_{t,i,j}$ is the feature at time t and spatial location (i, j) . We find that feature maps at 1/4 resolution provide sufficient detail for grouping, obviating the need for any

compute-heavy decoders used in off-the-shelf segmenters [Ye et al., 2025; Cheng et al., 2022]. Furthermore, we detach the gradients of $\mathbf{F}$ before entering the Perceiver layers to prevent unstable co-adaptation between patch features and learnable queries.

While the number of learnable queries N_q is fixed, the number of trajectories N can vary. Queries that produce empty masks are discarded, and long videos are divided into temporal chunks that can be processed in parallel. This mechanism allows the tokenizer to propose a dynamic number of tokens that scales naturally with scene complexity.

Training the segmenter. The segmenter can be trained either independently or jointly with the other objectives. We use supervised learning for the segmenter with pseudo ground-truth masks (generated via the TrajViT [Zheng et al., 2025] pipeline). We find that a combination of *Dice loss* [Sudre et al., 2017] and *Focal loss* [Lin et al., 2017], without standard *cross-entropy*, yields the best downstream understanding. This combination prioritizes the discovery of all object regions over strict pixel-level class accuracy. This recipe likely arises because pixel-level precision is less critical for the downstream video benchmarks and applications we evaluate. In particular, Dice loss plays a central role in the discovery of all object regions within the visual input, ensuring robust semantic grouping.

Trajectory Encoder

The trajectory encoder aggregates patch-level feature maps into compact tokens corresponding to segmented regions. Unlike the TrajViT encoder described in the previous section, the TrajTok encoder accepts both soft and hard segmentations to ensure differentiability while maintaining disentangled representations. We default to use the patch encoder’s features F as the input feature map, but in practice it can be provided by any pretrained feature $f(video)$, enabling our tokenizer to operate as a plug-in feature adapter across diverse downstream tasks.

Trajectory proposal generation. Initial trajectory embeddings are generated by a weighted aggregation of features using the soft masks. The proposal embedding $\mathbf{z}_k^{\text{init}}$ for the k -th trajectory is computed as:

$$\mathbf{z}_k^{\text{init}} = \sum_{t,i,j} \mathbf{M}_{k,t,i,j}^{\text{soft}} \cdot \mathbf{F}_{t,i,j} \quad (4.3)$$

This soft aggregation allows gradients from downstream tasks to flow back into the segmenter. However,

weighted summing can lead to information loss and blurred representations, which we address next.

Trajectory embedding refinement. To sharpen these representations, we employ a second Perceiver module. The initial proposals $\mathbf{z}^{\text{init}}$ serve as cluster representations. We can now use them as queries to extract meaningful representations from $\mathbf{F}$. To ensure disentanglement, we enforce *masked* cross-attention using hard segmentation maps, $\mathbf{M}^{\text{hard}}$, obtained by applying an argmax to $\mathbf{M}^{\text{soft}}$ and converting to one-hot binary assignments. The k -th query $\mathbf{z}_k^{\text{init}}$ is only allowed to attend to features $\mathbf{F}_{t,i,j}$ where $\mathbf{M}_{k,t,i,j}^{\text{hard}} = 1$. This refinement recovers fine-grained motion and texture details specific to the trajectory’s region.

Adaptive token number per trajectory. In practice, assigning a single token to each trajectory can be overly restrictive, especially for trajectories that span long durations, exhibit complex motion, or undergo substantial appearance changes. We introduce an adaptive token mechanism inspired by Matryoshka representations [Kusupati et al., 2022] to balance efficiency and expressivity. Given a predefined compute budget, the encoder can emit $n \in \{1, 2, 4\}$ tokens per trajectory, enabling a flexible trade-off between efficiency and expressivity. This mechanism is applied during the Trajectory embedding refinement, where each trajectory token can be expanded into multiple sub-tokens. We illustrate the details next.

For each of the initial trajectory embedding $\mathbf{z}_k^{\text{init}}$, we duplicate the token n times and associate each copy with a distinct learnable query vector. During the subsequent attention step, these queries interact with the same set of patch features, allowing them to capture complementary aspects of the same trajectory. However, we observe minimal performance gain with a naive initialization of these queries, as they tend to attend to the same dominant regions without explicit encouragement of diversity. To encourage diversity among sub-tokens, these queries are initialized with Fourier positional embeddings with angular offsets that maximally separate them in feature space.

To train such tokens, similar to Matryoshka Representations, we randomly sample $n \in \{1, 2, 4\}$ for each batch during training so that a single model can handle multiple token granularities. At inference, n can be adjusted according to available computational resources.

4.4 Experiments

We evaluate both TrajViT and TrajTok on a diverse set of video understanding tasks. Our experiments are organized into four parts: first, we evaluate training a video encoder from scratch using trajectory-based

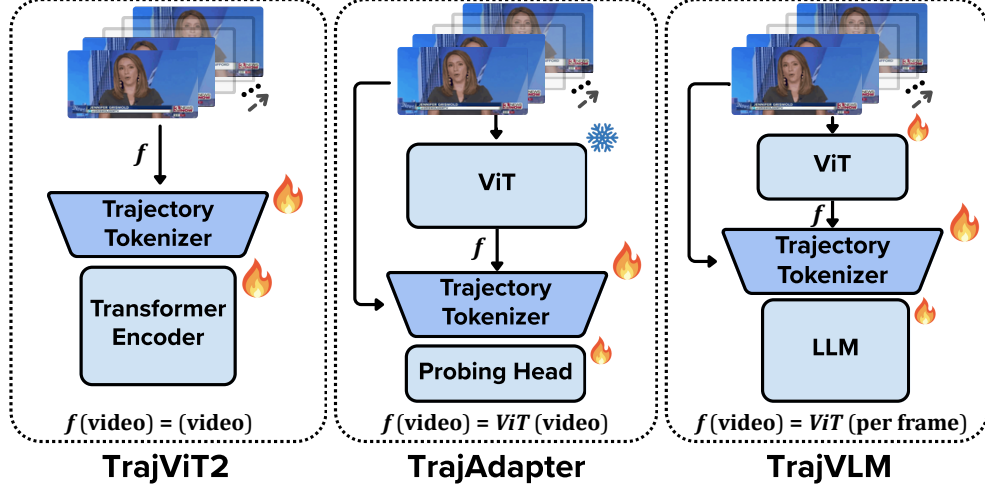


Figure 4.8: TrajTok is a versatile module applicable across pretraining, feature adaptation, and finetuning stages. We demonstrate its use in three scenarios: TrajViT2, which trains a visual encoder from scratch; TrajAdapter, which adapts pretrained features for downstream tasks; and TrajVLM, which uses TrajTok as a connector in LLaVA-style large vision–language models.

tokenization (both the external-pipeline and the end-to-end approaches); second, we show that TrajTok can adapt pretrained features as a plug-in module; third, we demonstrate the effectiveness of trajectory tokenization in vision-language models; and fourth, we present ablation studies for both systems.

4.4.1 Training a Video Encoder from Scratch

We first evaluate the scenario where trajectory tokenization is used to train a video encoder from scratch under the CLIP objective. We present results for both TrajViT (which uses an external segmentation-and-tracking pipeline) and TrajViT2 (which uses TrajTok end-to-end), and compare them against standard baselines.

Experimental Setup

Training recipe. All models use the transformer architecture of ViT-Large [Dosovitskiy et al., 2021] without pretrained weights and are trained with the CLIP objective [Radford et al., 2021] on randomly sampled subsets of the Panda-70M video captioning dataset [Chen et al., 2024d]. For the TrajViT experiments, we uniformly sample 16 frames per video during training. We partition the dataset into three scales—2M, 4M, and 8M video samples—to analyze scaling performance; TrajViT and ViT3D are trained on all three scales, while other baselines are trained on the 4M subset due to computational constraints. In one experiment

that adds image data, we use 50M randomly sampled image-text pairs from DataComp-1B [Gadre et al., 2024]. For the TrajViT2 experiments, the training corpus contains 4M video clips from Panda-70M and 15M image-caption pairs from CC3M [Changpinyo et al., 2021] and CC12M [Sharma et al., 2018]. We uniformly sample 8 frames per video during training and 16 frames during evaluation, and use a global batch size of 1024 images and 128 videos for 20 epochs on 8 A100 GPUs. After pretraining, all encoders are frozen for downstream evaluation.

Baselines. We compare against the following baselines in a strict apple-to-apple manner, ensuring they share the same training dataset and recipe as the corresponding trajectory-based model.

- **Space-time ViT (ViT3D):** A widely used baseline that tokenizes videos using fixed space-time patches and processes them with a standard transformer encoder [Arnab et al., 2021].
- **AutoMerge:** A variant of TrajViT which uses a learnable token merging mechanism on top of space-time patches to replace trajectory tokens. This serves as a control to study the impact of trajectory tokens (architecture detail in supplementary). We adjust its produced token count to match its FLOPS with that of TrajViT at 16-frame videos for a fair comparison. *This baseline is only used in the TrajViT comparison.*
- **Video token merging methods:** We compare with state-of-the-art token merging encoders including ToMe [Bolya et al., 2023], RLE/RLT [Choudhury et al., 2025b], and TokenLearner [Ryoo et al., 2021], which reduce video token counts by dropping similar patches or intermediate patch features. We follow the default token reduction hyperparameters from the original papers.
- **ViViT [Arnab et al., 2021]:** A widely-used video architecture that improves video processing efficiency by factorizing spatial and temporal attention.

All baseline models follow their original token number settings. In the TrajViT2 experiments, we additionally include TrajViT for direct comparison between the external-pipeline and end-to-end approaches.

TrajTok tokenizer details. For TrajViT2, we use ConvNext-tiny [Liu et al., 2022] as the architecture for the patch encoder. The perceiver modules in the segmenter and trajectory encoder both have 2 layers and 8 attention heads. We use 128 learnable queries inside the segmenter to cluster visual inputs into trajectories. Ablations of these design choices are presented in Section 4.4.4. When using TrajTok in a pretraining task

where the training data is fully controllable, we initialize the segmenter from scratch and jointly train the tokenizer with other modules using an added segmentation loss. However, when using the tokenizer as a feature adaptor in downstream tasks, we do not assume access to large-scale labeled segmentation data. We therefore pretrain a universal segmenter that can be reused across tasks without segmentation supervision during adaptation. To achieve this, we annotate 8M videos [Chen et al., 2024d] and 15M images [Sharma et al., 2018; Changpinyo et al., 2021] with panoptic object trajectory masks generated by the TrajViT pipeline, which serve as pseudo ground truth. Optimization details are described in the supplementary material. *This trained segmenter is reused in both TrajAdapter and TrajVLM.*

Benchmarks. We evaluate on a broad set of visual understanding benchmarks. For video-text retrieval, we use ActivityNet-caption [Krishna et al., 2017a], Charades [Sigurdsson et al., 2016], MSR-VTT [Xu et al., 2016], and VATEX [Wang et al., 2019] in a zero-shot setting; TrajViT2 is additionally evaluated on image-text retrieval with COCO [Lin et al., 2014] and Flickr30K [Plummer et al., 2015]. For action classification, we evaluate on Something-Something V2 (SSV2) [Goyal et al., 2017], Kinetics-400 (K400) [Kay et al., 2017], and UCF-101 [Soomro et al., 2012] using a multi-head attention probing (MAP) head following prior works [Wang et al., 2024c; Zhao et al., 2024; Yuan et al., 2023], aggregating predictions from 4 temporal views per video [Arnab et al., 2021; Yuan et al., 2023; Wang et al., 2024c]; TrajViT2 is additionally evaluated on ImageNet-1K [Deng et al., 2009], CIFAR-100 [Krizhevsky et al., 2009], and Caltech-101 [Fei-Fei et al., 2004]. For fine-grained spatial-temporal understanding (TrajViT only), we evaluate on AVAv2 [Gu et al., 2018] for spatial-temporal detection and on temporal clip retrieval tasks from ActivityNet [Krishna et al., 2017a] and YouCook [Das et al., 2013].

Results

Video-text retrieval. We evaluate zero-shot video-text retrieval using the pretrained video and text encoders. As shown in Table 4.1, both TrajViT and TrajViT2 achieve significant improvements over all baselines. We attribute this to the nature of video-text retrieval tasks, where textual descriptions primarily focus on objects and their interactions. By leveraging semantic tokens derived from object trajectories, our models are better at identifying and matching objects, actions, and their relationships with the textual queries. TrajViT2 further improves upon TrajViT, attaining higher recall on all retrieval datasets (e.g., +4.1% vid2txt R@5

Model	ActivityNet		VATEX		MSR-VTT		Charades		COCO		Flickr30K	
	txt2vid	vid2txt	txt2vid	vid2txt	txt2vid	vid2txt	txt2vid	vid2txt	txt2img	img2txt	txt2img	img2txt
ViT3D	37.1	35.6	36.4	60.2	31.4	58.1	13.8	12.9	61.5	67.4	75.2	83.6
TokenLearner	36.4	36.2	34.3	58.8	30.2	57.6	12.0	12.5	59.3	65.8	73.0	81.0
ViViT	33.9	33.2	34.0	57.6	29.6	55.2	12.3	11.8	57.8	64.1	71.6	79.3
RLT	35.9	35.0	35.0	58.4	30.5	57.1	12.2	12.3	59.6	66.1	72.8	81.2
TrajViT	38.4	38.1	36.0	61.1	31.6	61.2	14.7	14.6	63.2	69.0	77.4	85.1
TrajViT-2	40.1	42.2	37.9	65.0	33.2	65.3	15.9	16.5	68.1	75.6	83.5	90.4

Table 4.1: Zero-shot video & image retrieval performance (R@5). TrajViT2 consistently surpasses all baselines by a large margin.

on ActivityNet and +4.0% on VATEX) and additionally achieves strong image-text retrieval on COCO and Flickr30K.

Model	K400	SSV2	IN-1K	CIFAR-100	Caltech-101
ViT3D	54.2	46.3	69.4	52.1	88.2
TokenLearner	52.9	42.4	67.3	49.5	86.7
ViViT	51.2	43.1	66.8	48.2	86.3
RLT	52.5	43.6	67.0	49.0	86.5
TrajViT	55.3	45.7	68.1	50.8	87.4
TrajViT-2	59.1	48.7	67.2	52.6	88.5

Table 4.2: Action & image classification (attentive probe, top-1). IN-1K stands for ImageNet-1K. TrajViT2 improves upon TrajViT and outperforms all baselines in most of the benchmarks.

Action and image classification. As shown in Table 4.2, TrajViT outperforms all baselines on K400 and UCF-101, and TrajViT2 achieves further improvements (e.g., +3.8% on K400 and +3.0% on SSV2). On ImageNet, TrajViT2 performs slightly lower than ViT3D, likely because ImageNet images typically contain a single centered foreground object, causing the segmenter to produce too few segments. Despite this, TrajViT2 matches or surpasses all other baselines on cross-domain and multi-object datasets, underscoring the strength of trajectory-level tokenization.

Scaling the number of input frames. We examine how different models perform as the number of input frames T increases. We use the ActivityNet dense captioning retrieval task as our benchmark. This task requires retrieving long video clips ranging from 1 to 2 minutes, where incorporating more than 16 frames can help the model better align with the provided complex and densely annotated captions. We analyze both model performance and computational efficiency under varying temporal input lengths. For TrajViT, we note

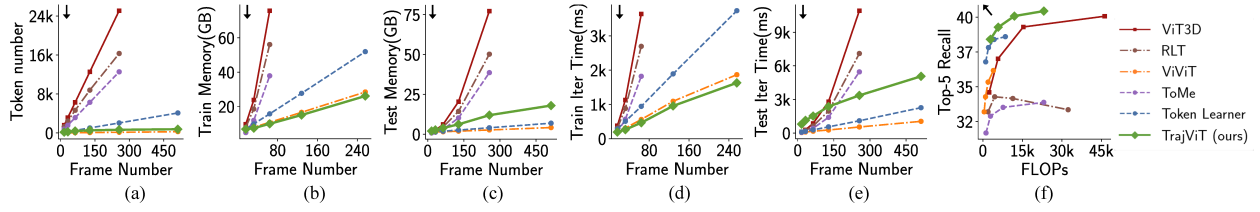


Figure 4.9: Comparison of inference frame number scaling in ActivityNet video-to-text retrieval task. Scaling with our tokenization paradigm obtains a better trade-off than baselines in terms of efficiency and accuracy.

that generating trajectories is a one-time cost for a video: we precompute and store trajectories used for every training epoch. At inference, we consider both the trajectory generation and the model forward pass as part of TrajViT’s resource consumption.

We show results in Figure 4.9. Compared to ViT3D, TrajViT scales effectively to a significantly larger number of frames while maintaining low resource consumption. We observe that TrajViT can generalize to more input frames with a smaller number of tokens (Figure 4.9a), lower training GPU memory (Figure 4.9b), and faster training iteration time (Figure 4.9d). When taking trajectory generation overhead into account at inference, TrajViT runs faster only for videos with more than 64 frames (Figure 4.9e), yet it consumes less GPU memory regardless of the number of frames (Figure 4.9c). Most importantly, Figure 4.9f demonstrates that **under the same computational cost, our approach consistently achieves better performance compared to all baselines**. These results highlight the potential of our design in building an efficient and strong long-term video understanding model. Note that the inference runtime and FLOPs of TrajViT are bottlenecked by the tracking pipeline, which is a potential direction to improve in our future work.

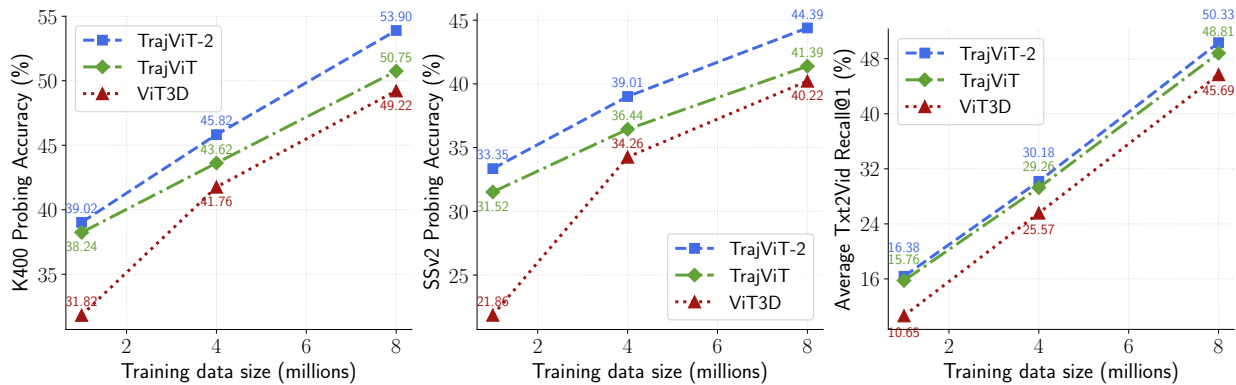


Figure 4.10: Scaling with video training data. TrajViT2 exhibits stronger scaling behavior than TrajViT and sustains a consistent performance margin over ViT3D at every data scale.

Scaling pretraining video data size. To examine how trajectory-based models scale with data, we train TrajViT, TrajViT2, and ViT3D on 1M, 4M, and 8M video subsets of Panda-70M. As shown in Figure 4.10, both TrajViT and TrajViT2 consistently outperform ViT3D across all scales. However, TrajViT’s performance gain over ViT3D diminishes as dataset size increases, while TrajViT2 exhibits a much stronger scaling trend. At the largest scale, TrajViT2 continues to outperform ViT3D by a large margin across both classification and retrieval tasks. We attribute this improvement to the end-to-end differentiability of TrajViT2’s tokenizer: the segmenter can flexibly adjust its segmentation behavior in response to the pretraining objective, rather than relying on fixed, heuristic segmentations. We give qualitative illustration in Figure 4.6.

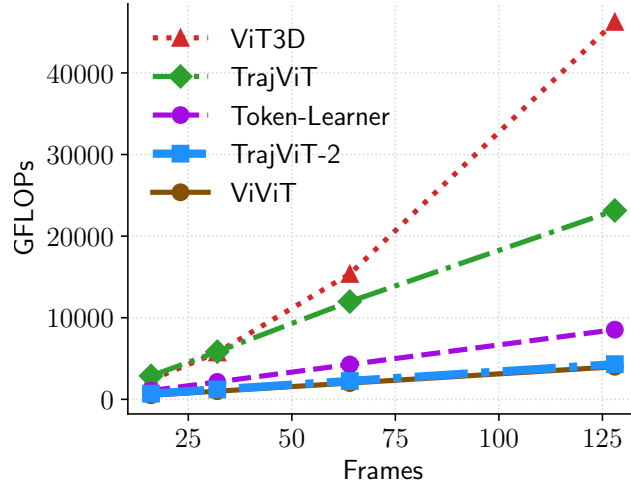


Figure 4.11: Test time FLOPs comparison under different frame numbers.

Efficiency. Another drawback of TrajViT is its heavy computational overhead caused by the external pipeline. TrajViT2 resolves this by replacing it with a lightweight, fully integrated segmenter. The entire trajectory tokenizer contains only 46M parameters—an order of magnitude smaller than the 304M parameters of the ViT-Large backbone. We compare inference FLOPs across input frame counts from 16 to 128 in Figure 4.11. TrajViT2 achieves nearly the same computational cost as the most efficient baseline, ViViT, in stark contrast to the quadratic scaling of patch-based ViT3D and the high-slope linear scaling of TrajViT. These results demonstrate that TrajViT2 achieves superior efficiency while maintaining strong performance.

Incorporating image data. Most video encoders are initialized from image-pretrained models [Arnab et al., 2021; Choudhury et al., 2025b; Wang et al., 2024c; Zhao et al., 2024; Lei et al., 2022]. We investigate whether TrajViT can also benefit from image data. Unlike ViT3D, which requires separate image pretraining before

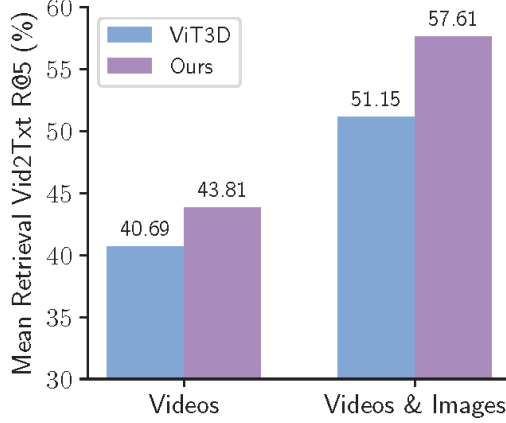


Figure 4.12: Incorporating image data. TrajViT benefits more from adding image data into pretraining as it requires no architectural modifications.

Probing Method	VideoMAE-v2		V-JEPA2	
	K400	SSv2	K400	SSv2
Linear probing	79.4	59.1	84.5	73.7
Attentive probing	80.2	59.7	85.1	74.2
Perceiver probing	79.9	59.8	84.7	74.2
TrajAdapter (1 token/traj)	82.0	60.4	87.2	74.6
TrajAdapter (2 token/traj)	82.4	60.8	87.8	74.9
TrajAdapter (4 token/traj)	82.5	60.9	88.0	75.1

Table 4.3: Probing results on different video backbones. Top-1 accuracy (%) on Kinetics-400 (K400) and Something-Something-V2 (SSv2) using various probing strategies. TrajAdapter consistently improves the downstream probing accuracy of pretrained backbone features, and performance further increases with more tokens per trajectory.

video adaptation, TrajViT can be jointly trained on image and video data as discussed in Sec. 4.3.1. We conduct an experiment using the largest 8M video subset alongside the DataComp-50M image dataset [Gadre et al., 2024]. We plot the average retrieval performance in Figure 4.12. Compared to ViT3D, **TrajViT achieves a larger performance boost when incorporating image data**, highlighting its ability to leverage image and video data more efficiently.

4.4.2 Adapting Pretrained Features

In practice, as pretrained large vision encoders continue to improve, it becomes increasingly desirable to directly reuse their output feature maps—often dense patch-level tokens—for downstream tasks. We show that TrajTok can be directly plugged in as a lightweight adapter module to reorganize these dense feature maps

into a compact set of trajectory tokens. We show this design not only reduces token number for downstream models but also provides a cost-effective way to enhance the probing performance of in downstream tasks without full fine-tuning.

Training and evaluation setup. We take video action classification as an example to demonstrate this setting. As illustrated in the second part of Figure 4.8, TrajTok is inserted after a frozen ViT backbone to reorganize output tokens, which are then used by an attentive probing head to predict classification logits. The segmenter is pretrained and *kept frozen* during probing, while the trajectory encoder is trained jointly with the probing head. For pretrained backbones, we adopt VideoMAE-v2 [Wang et al., 2023c] and V-JEPA-2 [Assran et al., 2025], both using their ViT-Huge variants. We evaluate action recognition accuracy on the Kinetics-400 and Something-Something V2 (SSv2) benchmarks. Videos are uniformly sampled to 16 frames and sent to segmenter in one forward pass, producing a maximum of 128 trajectory tokens.

We compare our approach against three baselines: (1) naive linear probing, (2) attentive probing without adaptation, and (3) a Perceiver module of identical size and number of learnable queries as our trajectory encoder but without trajectory priors. In addition, we enable the adaptive token number mechanism in the trajectory encoder and report results for varying numbers of tokens per trajectory.

Results. Table 4.3 summarizes the top-1 classification accuracy of different probing strategies. Compared to both linear and attentive probing, TrajAdapter consistently achieves higher accuracy across datasets. Furthermore, our method outperforms the Perceiver-only variant, indicating that the improvement arises not merely from additional parameters but from the incorporation of trajectory priors. In fact, naively inserting a Perceiver module does not yield any gain over the attentive probing baseline. We also observe a steady performance increase as the token number per trajectory grows, even though the single-token configuration already surpasses conventional probing methods. These results demonstrate that the proposed tokenizer is not only effective for end-to-end video representation learning, but also serves as a plug-in adapter that enhances features in pretrained ViT backbones in a parameter-efficient manner.

4.4.3 Vision-Language Models

We evaluate trajectory-based tokenization in the context of vision-language models, first by connecting TrajViT to an LLM, and then by using TrajTok as the connector module in a VLM.

TrajViT-LLM

Model	NextQA		TempCompass				VideoMME	ActivityNetQA		MovieChat		MLVU
	OE	MC	MC	Yes/No	Cap.Match	Cap.	Acc.	Acc.	Score	Acc.	Score	Acc.
ViT3D	49.2	29.5	39.1	50.1	58.4	30.4	28.9	30.7	2.6	34.7	3.1	19.2
TrajViT	65.1	37.0	40.0	50.6	59.9	31.1	32.0	38.0	2.7	36.7	3.1	32.2

Table 4.4: Comparison of ViT3D and TrajViT on VideoLLM QA tasks. We report results on NextQA (WUPS score in open-ended subsets, accuracy in multi-choice subsets), TempCompass (accuracy for four subsets of multi-choice, yes/no, caption matching, and captioning), VideoMME (accuracy), MovieChat (accuracy and GPT score), ActivityNetQA (accuracy and GPT score), and MLVU (accuracy). The VideoLLM that uses our video encoder consistently outperforms the one with ViT3D across all video QA benchmarks.

We evaluate various VideoQA tasks by connecting TrajViT to the Llama-3 language model [Liu et al., 2024c]. To focus on the impact of video encoders, we adopt the simple setup in [Maaz et al., 2024]: A linear layer is used to connect the trained video encoder and LLM. We train two VideoLLM variants with ViT3D and TrajViT as video encoder (the variants that pretrained on 8M video data and 50M image data), and fine-tune the linear connector using the instruction tuning dataset from LLaVA-video [Zhang et al., 2024e]. We train both VideoLLMs for 3 epochs with a batch size of 32 and evaluate the models on various QA benchmarks, including VideoMME [Fu et al., 2024b], NextQA [Xiao et al., 2021], ActivityNetQA [Yu et al., 2019], MLVU [Zhou et al., 2024], TempCompass [Liu et al., 2024e], and MovieChat [Song et al., 2024]. We follow the setup and evaluation metrics in the LMMs-Eval repository [Li et al., 2024a] for all benchmarks. The results are shown in Table 4.4. **TrajViT-LLM outperforms ViT3D-LLM across all QA benchmarks**, possibly because the modern VideoQA tasks often require reasoning about fine-grained objects and events in video, in which our trajectory-based tokenization helps. **TrajViT-LLM trains 4.2x faster and has 18x fewer inference FLOPs in average than ViT3D-LLM** because of the significantly fewer vision tokens proposed by our video encoder. Additionally, we specifically test how the trained VideoLLM performs in long videos by varying its input number of frames in MovieChat long video benchmark. We find that our model scales significantly better than ViT3D in both inference FLOPs and accuracy (Figure 4.13), demonstrating its strong potential in building long-term video models.

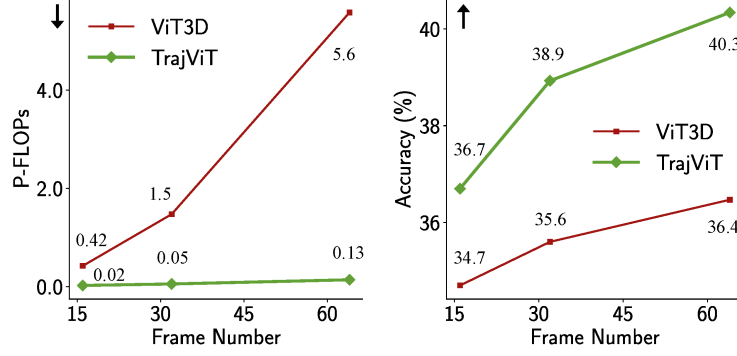


Figure 4.13: Accuracy and inference FLOPs at MovieChat long video benchmarks with input frame scaling. The VideoLLM with TrajViT as video encoder scales significantly better than the one with ViT3D in both accuracy and efficiency.

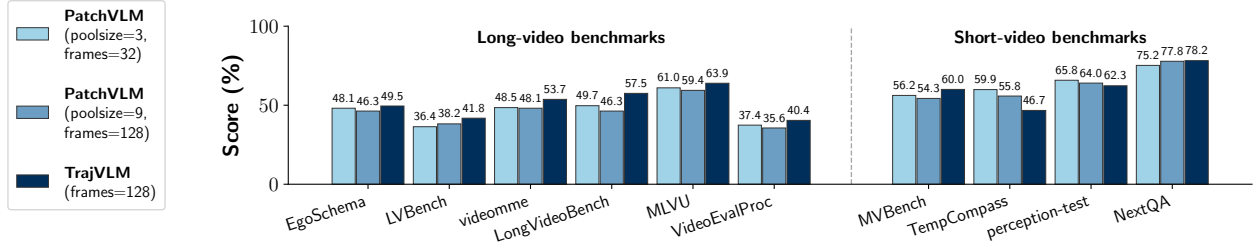


Figure 4.14: VideoQA results for TrajTok applying to large vision-language model. VLM with TrajTok as connector (TrajVLM) notably outperforms patch pooling baseline (PatchVLM) in long-video benchmarks, while the performance is mixed for short-video benchmark.

TrajVLM: A Vision-Language Model with TrajTok

Finally, we demonstrate that TrajTok can also serve as a connector between a vision encoder and a language model, providing an object-centric alternative to the patch-pooling connectors commonly used in large vision-language models (VLMs) [Liu et al., 2024a; Qwen Team, 2024; Huang et al., 2024; Deitke et al., 2025]. To this end, we build a small-scale model named **TrajVLM** by integrating our tokenizer into a standard LLaVA-style architecture (Figure 4.8 part 3). The goal of this experiment is not to compete with state-of-the-art VLMs, but rather to provide an *apple-to-apple* comparison between two connector designs: TrajTok and commonly-used patch pooling. Scaling TrajVLM to larger models with increased compute remains a future direction.

Architecture and Baseline. We use **Qwen3-4B** [Yang et al., 2025a] as the language model backbone and **SigLIP2-Huge** [Tschannen et al., 2025] as the vision encoder. For the baseline connector, we follow the

design of **Molmo** [Deitke et al., 2025], which employs per-frame, patch-based attention pooling. Specifically, each $x \times x$ spatial patch window is pooled into a single vector via a multi-head attention layer, where the mean patch embedding serves as the query. Notably, similar patch-pooling strategies are widely used in many of today’s most widely-used open-source vision–language models [Qwen Team, 2024; Xu et al., 2024; Wang et al., 2025b].

Training Data and Recipe. We adopt a subset of Molmo-2’s training corpus [Clark et al., 2025], including the PixMo captioning split, synthetic VideoQA split, and academic QA datasets (details in supplementary). Training follows Molmo’s two-stage procedure:

- **Pretraining.** All parameters are pretrained on the PixMo captioning split for one epoch to align visual features with the language model.
- **Fine-tuning.** The model is then fine-tuned for 10,000 steps on the remaining QA datasets.

All experiments are conducted on 8 A100 GPUs with a sequence length of 8,192 tokens. For TrajVLM frame sampling, we uniformly sample 128 video frames during both training and evaluation. TrajTok connector processes 128 frames by truncating them into 16-frame clips, each proposing a maximum of 128 tokens. For baseline PatchVLM, we train two versions of the model: a version that uses common patch pooling size $x = 3$, but can only support 32 frames due to sequence length limits; Another version that uses patch pool size $x = 9$ so that it can support 128-frame during training and the resulting number of visual tokens roughly matches that of our trajectory tokenizer, ensuring a fair comparison.

Results. We evaluate VLMs in common video QA benchmarks [Ma et al., 2025; Liu et al., 2024e; Pătrăucean et al., 2023; Xiao et al., 2021; Li et al., 2024b; Mangalam et al., 2023; Zhang et al., 2023c; Fu et al., 2024b; Zhang et al., 2024b; Zhou et al., 2025]. Figure 4.14 shows that TrajVLM consistently outperforms the patch-pooling baselines on long-video benchmarks, including a notable +8.8% on LongVideoBench and +5.4% on LVBench over PatchVLM with default poolsize=3. We attribute our improvement to the fact that TrajTok produces semantically structured tokens that better support long-range reasoning while reducing redundancy. Notably, increasing the pooling window size in PatchVLM does not improve long-video performance on most benchmarks, indicating that naively trading off spatial resolution with temporal support is insufficient for long-range reasoning. For Short-video benchmarks, we observe increased performance on [Xiao et al., 2021; Li et al., 2024b] but decreased performance on [Liu et al., 2024e; Pătrăucean et al.,

2023]. Overall, these results validate TrajTok as an effective connector for VLMs, particularly in long-video understanding.

4.4.4 Ablation Studies

We present ablation studies for both TrajViT (external pipeline) and TrajTok (end-to-end), covering the impact of individual components and design choices.

TrajViT Ablations

Impact of segmentation and tracking. When generating trajectories, our tokenization paradigm relies on segmentation and tracking models, both of which introduce unique inductive biases in generated tokens: segmentation reduces spatial redundancy and adds semantic information, while tracking reduces temporal redundancy. To understand their contributions, we analyze the individual impact of segmentation and tracking on model performance: the segmentation-only baseline segments each frame individually without considering temporal redundancy, while the tracking-only baseline performs tracking on 16x16 patches without segmentation. We use the patch tracking model in [Jabri et al., 2020]. Both baselines use the same tokenizer architecture as TrajViT. As shown in Table 4.5, Both baselines underperform TrajViT. The segmentation-only model improves ViT3D on retrieval tasks but underperforms in both classification tasks, perhaps because spatially semantic tokens help the model better capture objects but fail to recognize actions due to the lack of temporal connections. Additionally, we find that patch-based tracking is not robust, as tracking in patches is less semantically meaningful than tracking object segments. This results in the patch-tracking model underperforming TrajViT.

Trajectory generation pipeline. We ablate different design choices of the trajectory generation pipeline and show performance in Table 4.6.

Segmentation model. The model that uses segments from DirectSAM consistently outperforms the one with segments from SAM, likely due to DirectSAM’s ability to produce more detailed segmentation masks.

Trajectory resolution. Both DirectSAM and SAM2 are originally trained at a resolution of 1024. While using their original resolution provides more precise boundary in the generated masks, we observe minimal impact on downstream performance. Given the $4\times$ reduction in compute, we adopt 512×512 resolution as

our default setting.

Frame partitioning strategy. We find that using frame stride only underperforms in retrieval-related tasks, possibly because some objects in the mid-video fail to be detected at their first appearance. Using key frame only degrades performance in the SSV2 classification task. We suspect that this is because object changes rarely occur in SSV2, leading to very few detected key frames. As a result, each token is overloaded to represent lengthy trajectories.

Trajectory encoder architecture. The ablations for different components of trajectory encoder architecture are shown in Table 4.6.

Feature extractor. Extracting features from ResNet-18 significantly outperform hierarchical transformer such as Hiera-small, perhaps because convolutional filters provide better spatial features, especially for small modules.

Necessity of hierarchical features. Incorporating hierarchical features improves performance, likely because they help compensate for information loss during mask pooling operations when extracting region-specific features.

Necessity of the position branch. Although the appearance token may already encode some positional information from ResNet features, we find that an explicit position branch remains essential for strong performance.

Feature aggregation across frames. For aggregating per-frame features, we find that using perceiver resampler significantly outperforms simple mean-pooling, highlighting the importance of modeling temporal variation.

Number of query tokens per trajectory. Increasing the number of query tokens per trajectory provides only marginal performance improvements. This suggests that trajectory information is already heavily compressed before reaching the perceiver resampler. Addressing this potential bottleneck is left to future work.

TrajTok Ablations

We ablate the design choices of TrajTok under TrajViT2 setting, which trains a video encoder jointly optimized by the segmentation loss and CLIP loss. All experiments are trained on 1M video-caption pairs randomly sampled from Panda-10M for 10 epochs using 4 GPUs.

Variation	SSV2	K400	Retrieval
<i>vit3d</i>	17.2	21.8	12.0
Segmentation-only	16.4	19.9	17.0
Tracking-only	19.3	18.1	15.4
TrajViT	22.7	25.9	17.8

Table 4.5: Individual impact of segmentation and tracking. We can see that removing segmentation (tracking-only) or tracking (segmentation-only) significantly hurts model performance.

	Objectives	Variation	SSV2	K400	Retrieval
		TrajViT	22.7	25.9	17.8
traj-gen	Resolution	1024→512	22.5 (↓ 0.2)	26.3 (↑ 0.4)	18.0 (↑ 0.2)
	Seg. Type	D-SAM→SAM	22.1 (↓ 0.6)	23.5 (↓ 2.4)	16.7 (↓ 1.1)
	Clip Par.	Key frame only	20.6 (↓ 2.1)	25.7 (↓ 0.2)	17.6 (↓ 0.2)
		Frame cut only	22.7 (0)	25.3 (↓ 0.6)	17.3 (↓ 0.5)
traj-encoder	Hiera. Feat.	Yes→No	22.4 (↓ 0.3)	25.1 (↓ 0.8)	17.5 (↓ 0.3)
	Pos. Branch	Yes→No	20.6 (↓ 2.1)	25.3 (↓ 0.6)	17.4 (↓ 0.4)
	ConvNet	ResNet→Hiera	18.3 (↓ 4.4)	21 (↓ 4.9)	16.0 (↓ 1.8)
	Feature Aggr.	Percei.→Mean	19.2 (↓ 3.5)	23.4 (↓ 2.5)	16.5 (↓ 1.3)
	Query Token	1→2	22.7 (0)	26.0 (↑ 0.1)	17.8 (0)
		1→4	22.8 (↑ 0.1)	26.2 (↑ 0.3)	17.9 (↑ 0.1)

Table 4.6: Ablation study of both trajectory generation pipeline and trajectory encoder architecture.

Ablation of segmenter design. We ablate the major design choices of the segmenter, including backbone hierarchy, gradient detachment, output resolution, and segmentation loss functions. We use VEQ and STQ metrics to quantify video panoptic segmentation quality following [Ye et al., 2025], and use average txt2vid R@5 accuracy across video retrieval benchmarks [Caba Heilbron et al., 2015; Wang et al., 2019; Xu et al., 2016; Sigurdsson et al., 2016] to quantify video understanding performance. As summarized in Table 4.7, removing hierarchical features yields consistent drops across VEQ, STQ, and retrieval accuracy, confirming the importance of multi-scale representations. Removing the gradient detachment of the patch-feature inside the perceiver causes large declines in segmentation quality due to coupled updates between queries and patch features. Increasing the output resolution slightly improves VEQ/STQ, but has negligible impact on retrieval, indicating that coarse masks are sufficient for semantic grouping. Among loss components, dice loss is the most critical: removing it severely harms both segmentation and understanding performance.

Ablation of trajectory encoder design. We also ablate the key components of the trajectory encoder,

Module	Variation	VEQ (%)	STQ (%)	Retrieval (R@5)
Default Architecture		42.3	70.1	22.1
Backbone	no hierarchical features	39.3 (↓ 3.0)	66.2 (↓ 3.9)	19.2 (↓ 2.9)
Output Res.	56→224	44.1 (↑ 1.8)	73.0 (↑ 2.9)	22.0 (↓ 0.1)
Perceiver	no detach gradient	34.1 (↓ 8.2)	59.3 (↓ 10.8)	18.3 (↓ 3.8)
Seg. Loss	– dice loss	39.0 (↓ 3.3)	68.9 (↓ 1.2)	16.7 (↓ 5.4)
Seg. Loss	– focal loss	41.2 (↓ 1.1)	67.4 (↓ 2.7)	22.1 (0.0)
Seg. Loss	+ cross-entropy loss	42.1 (↓ 0.2)	71.3 (↑ 1.2)	21.3 (↓ 0.8)

Table 4.7: Ablations of segmenter design.

Module	Variation	1 token/traj	2 token/traj	4 token/traj
Default Architecture		22.1	23.0	23.2
Attention Mask	w/o mask	17.4 (↓ 4.7)	17.9 (↓ 5.1)	18.3 (↓ 4.9)
Query Init.	Fourier → random	22.1 (0.0)	22.5 (↓ 0.5)	21.9 (↓ 1.3)
Perceiver Depth	2 → 4	22.3 (↑ 0.2)	22.9 (↓ 0.1)	23.5 (↑ 0.3)

Table 4.8: Ablations of trajectory encoder design.

including the use of attention masks, query initialization strategies, and perceiver depth under the same retrieval task (Table 4.8). Removing the hard attention mask significantly degrades performance by weakening the association between trajectory tokens and their assigned regions. For query initialization under multi-token per trajectory setting, increasing the number of tokens per trajectory would not improve performance if Fourier-based query initialization is replaced by random initialization. That might be because different slots might extract the same trajectory information without explicit diversity encouragement. Finally, increasing perceiver depth yields only marginal improvements at higher computational cost, suggesting that a shallow perceiver is already sufficient for capturing local dynamics within trajectories.

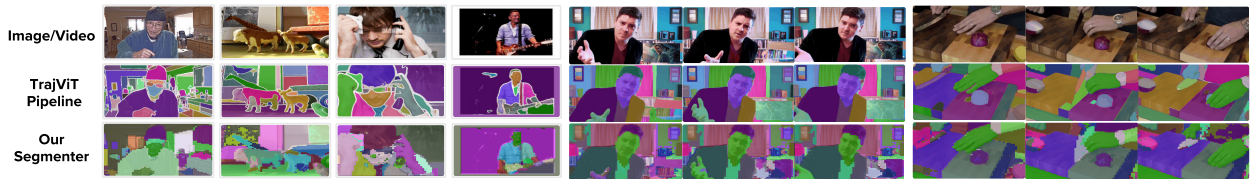


Figure 4.15: Trajectory masks produced by our segmenter vs. by TrajViT pipeline. While our segmenter produces coarser masks and may miss very small objects, it demonstrates strong semantic grouping ability that is sufficient for downstream understanding tasks.

4.5 Conclusion

This chapter has presented two complementary approaches to trajectory-based video tokenization. TrajViT demonstrated that aligning tokenization with perceptual principles of object persistence and common fate yields substantial improvements across video understanding tasks while significantly reducing token counts. TrajTok then addressed TrajViT’s reliance on an external pipeline by introducing an end-to-end differentiable tokenizer that is faster, more flexible, and achieves even stronger performance. Together, these results establish trajectory-based tokenization as a principled and practical alternative to patch-based approaches, offering a more semantically aligned visual interface for multimodal reasoning.

With better evaluation (Chapter 2), richer supervision (Chapter 3), and more grounded representations (this chapter) in place, the next chapter presents a complete vision-language model that brings these threads together—demonstrating that grounded capabilities such as pointing, tracking, and spatially precise reasoning become possible when the model architecture is designed with grounding as a first-class objective.

Chapter 5

Unlocking Grounded Multimodal Reasoning in Vision-Language Models

5.1 Introduction

The preceding chapters have built toward grounded visual understanding from three directions: reliable evaluation of model capabilities (Chapter 2), richer training supervision through scene-graph-based instruction data (Chapter 3), and object-centric visual representations that align tokenization with scene structure (Chapter 4). This chapter presents Molmo2, a fully open vision-language model that brings these threads together by making grounding a first-class capability—enabling the model to not only describe what it sees, but to *point*, *track*, and *reason* over visual content with spatial and temporal precision.

Visual data—especially video—is now ubiquitous, streaming continuously from phones, home cameras, social media, autonomous systems, and industrial sensors [Do et al., 2025]. Understanding this video is fundamental for applications such as video search, robotics, assistive technologies, sports analytics, and autonomous driving [Liang et al., 2024; Licardo et al., 2024; LinLin et al., 2024]. Yet the strongest video–language models remain proprietary [Team, 2024a; OpenAI, 2024; Wang et al., 2024a], with closed weights, data, and training recipes. A key missing capability in these models is *grounding*: the ability to answer “How many times does the robot grasp the red block?” by emitting *points* for each grasp event in space and time, or to identify “When did the cup fall off the table?” by returning a *track* of the cup so users can precisely

locate the event. Although image grounding is now standard [Cheng et al., 2025], video grounding is only supported in some proprietary systems, and even there in a limited form.

We present **Molmo2** (Multimodal Open Language Model), a family of fully open state-of-the-art vision-language models that support single image, multi-image, and video inputs, bridging this gap by bringing grounding capabilities to video understanding. To promote open research, we release our training data, model weights, and training code, all constructed without distilling from proprietary models.

A core contribution is a suite of 9 novel datasets targeting crucial skills underrepresented in existing open data for video and multi-image inputs. These include: (1) two open-vocabulary video pointing and tracking datasets (265k instances) that enable models to pinpoint when and where events or objects occur; (2) a dense video captioning corpus (104k videos) with captions far longer and more detailed than prior work [Zhang et al., 2025; Chen et al., 2024c]; (3) two long-form QA datasets (215k instances) with rich human-crafted answers; (4) two long-video QA datasets (around 1.3M instances) addressing long-duration content [Qian et al., 2024]; and (5) two multi-image datasets for pointing and document understanding. Our data collection pipelines include spoken narration transcribed and enriched with frame-level details (Figure 5.1), human-LLM collaboration for long-form QA without proprietary distillation [Maaz et al., 2024; Liu et al., 2024f; Jiang et al., 2024; Li et al., 2025; Chen et al., 2024c], and synthetic generation from our own captioning models.

For grounding, we extend the 2D pointing paradigm from image-based VLMs [Deitke et al., 2025; Jiang et al., 2025; Yuan et al., 2024] into the temporal domain, creating dedicated datasets for video pointing and video tracking with broad vocabulary coverage that goes well beyond existing narrow-scope benchmarks [Ahmad et al., 2025; Munasinghe et al., 2025]. All Molmo2 variants are trained in a two-stage pipeline—image-only pre-training followed by joint supervised fine-tuning on images, videos, and multi-image inputs—and we introduce several training innovations: a novel token-weighting scheme to balance diverse tasks, sequence packing and message-tree scheduling for training efficiency, and bi-directional attention between visual tokens. Our models achieve competitive results with the best open-weight video models on standard benchmarks [Li et al., 2024c; Zhang et al., 2025], significantly surpass previous fully-open models, and set a new state-of-the-art on video grounding tasks [Ahmad et al., 2025; Munasinghe et al., 2025].

5.2 Related Work

Video-language models. Video understanding has advanced rapidly with the development of video-language models that extend image-based architectures to temporal sequences. Proprietary systems such as Gemini [Team, 2024a] and GPT-4o [OpenAI, 2024] currently lead on many benchmarks, while open-weight models like Qwen2-VL [Wang et al., 2024a], LLaVA-Video [Zhang et al., 2025], and VideoChat [Li et al., 2024c] have narrowed the gap. However, most open models are trained on data distilled from proprietary systems, limiting transparency and reproducibility. Long-video understanding remains a particular challenge for open models, with performance degrading on content exceeding a few minutes [Qian et al., 2024]. Molmo2 addresses both limitations by training exclusively on open data and introducing dedicated long-video datasets.

Video grounding and pointing. Grounding—localizing objects and events in space and time—is a critical capability for practical video understanding. Image-level pointing and grounding have become standard in recent VLMs [Deitke et al., 2025; Jiang et al., 2025; Yuan et al., 2024; Cheng et al., 2025], but video grounding remains nascent. Existing video grounding datasets are typically narrow in vocabulary and domain [Ahmad et al., 2025; Munasinghe et al., 2025], and referring video object segmentation benchmarks [Bellver et al., 2020; Seo et al., 2020; Ding et al., 2023] focus on mask prediction rather than the point-based interaction paradigm suited to generalist VLMs. Molmo2 extends pointing into the temporal domain, enabling spatio-temporal localization of events and continuous object tracking expressed as sequences of points, supported by large-scale open-vocabulary training data.

Multimodal training data curation. The quality and diversity of training data is a primary bottleneck for multimodal models. Large-scale image–text corpora [Gadre et al., 2024] provide broad coverage but shallow supervision, while visual instruction datasets [Liu et al., 2024a; Dai et al., 2024; Jiang et al., 2024] offer richer task-oriented training but are often generated by distilling proprietary models [Maaz et al., 2024; Chen et al., 2024c; Liu et al., 2024f]. For video, the scarcity of high-quality open data is acute: existing captions tend to be short and generic, and QA datasets rarely cover long-form or grounded interactions. Our data contribution addresses these gaps through human annotation pipelines (spoken narration for dense captioning,

human-LLM collaboration for QA), synthetic generation from open models, and careful curation of academic benchmarks into training-ready formats.

5.3 Molmo 2: Grounded Multimodal Reasoning in Practice

5.3.1 Data

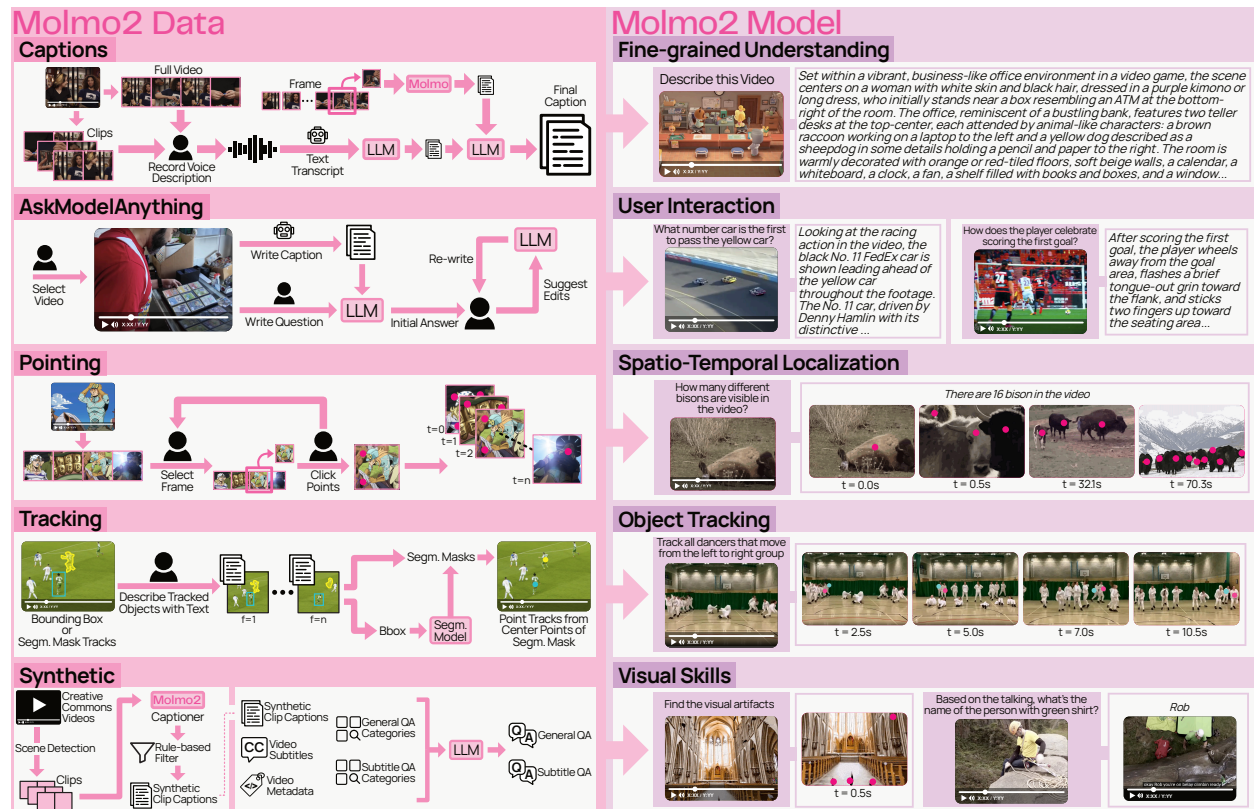


Figure 5.1: Molmo2 is trained on the largest fully open video-centric multimodal corpus to date, including nine new datasets for dense video captioning, long-form and long-video QA, and open-vocabulary pointing and tracking over images, multi-images, and videos. Molmo2 accepts single images, image sets, and videos as input and can produce both free-form language and grounded outputs such as spatio-temporal points, object tracks, and grounded chain-of-thoughts that localize objects and events over time. Across diverse video-language and grounding benchmarks, Molmo2 matches or surpasses prior open models, approaches proprietary systems, and remains fully open.

We create five human-annotated datasets and four synthetic datasets, and additionally curate two datasets by repurposing existing open-source data.

Molmo2-Cap (human). We collect 104k video-level and 431k clip-level dense captions from annotators, targeting both high detail and broad diversity. Videos are drawn from multiple large-scale sources [Zellers et al., 2022; Wang et al., 2025a, 2023d; Zhang et al., 2025], starting from a pool of over 10M clips, then filtered for informativeness and sampled for diversity to obtain a balanced subset.

Obtaining dense video captions is challenging because annotators must describe dynamic events alongside fine-grained visual details [Krishna et al., 2017a]. We use a two-stage pipeline: annotators first describe short clips, then summarize the entire video. As in PixMo-Cap [Deitke et al., 2025], annotators speak their descriptions, which are transcribed with Whisper-1 and then rewritten by a text-only LLM for coherence. We condition annotators to describe dynamic visual details (e.g., object or event changes over time) by prompting them with a set of predefined questions. To add any missing low-level details, we use Molmo to generate frame-level captions and an LLM to merge the clip and frame captions into a single long caption. This produces the densest video caption dataset to date, averaging 924 words per video, compared to 75 words in Video Localized Narratives [Voigtlaender et al., 2023], 89 and 100 in RCap and RDCap [Cho et al., 2025], 280 in ShareGPT4-Video [Chen et al., 2024c], and 547 in LLaVA-Video-178K [Zhang et al., 2025].

Molmo2-AskModelAnything (human). We collect 140k human-authored video QA pairs. Using video captions, we cluster videos into 31 categories and sample them evenly to promote data diversity. Annotators then write specific, fine-grained questions (e.g., about text, actions, or temporal relations), while we discourage counting questions (handled separately by pointing data), overly generic prompts, or questions requiring expert knowledge. For each question, we first obtain an initial answer from an LLM (Claude Sonnet 4.5) conditioned on a caption generated by an early Molmo2 captioner. Annotators either accept the answer or iteratively refine it through dialogue with the LLM. Finally, we post-process all QA pairs with an LLM filter to remove non-English, mismatched, or counting questions.

Molmo2-CapQA and -SubtitleQA (synthetic). To build large-scale synthetic video QA, we use a video captioner trained on Molmo2-Cap to caption videos from YT-Temporal [Zellers et al., 2022] and YouTube keyword search. We segment each video into multiple scenes and caption each scene instead of the entire video to encourage detailed descriptions. An LLM then uses these captions and video metadata to generate 1M QA pairs (200k videos, 5 QA per video). For SubtitleQA, we transcribe the video audio with Whisper-

1 [Radford et al., 2023] and additionally prompt the LLM with the transcript to create 300k QA pairs (100k videos, 3 QA per video) that require reasoning over both visual content and language.

Molmo2-VideoPoint (human). To improve Molmo2’s counting and spatial-temporal localization, we collect over 250k video pointing queries on 111k videos, with an average of 4.5 points per video, targeting seven diverse categories: objects, animals, actions/events, referring expressions, indirect references, spatial references, and visual artifacts. We generate queries by using LLM on video captions from an early version of Molmo2. Annotators first identify the frame where an object appears and then click on its exact location in the frame. Frames were obtained at 2 fps.

Molmo2-VideoTrack (human). We collect point-based object-tracking data covering 3.6k video clips and 15k complex natural language queries, with an average of 2.28 objects per query. Our dataset collection follows Ref-VOS [Bellver et al., 2020] by asking users to re-label existing tracking annotations. For each video, we display either segmentation or bounding box object tracks, and ask annotators to craft non-trivial text queries that apply to a subset of object. The queries are then validated in a separate validation round. We source videos and tracks from diverse open-source segmentation tracks [Bellver et al., 2020; Ding et al., 2025; Miao et al., 2022; Ravi et al., 2025] and bounding-box tracks [Sun et al., 2022; Zhang et al., 2023d; Scott et al., 2024; Wang et al., 2022; Giancola et al., 2018; Dendorfer et al., 2020; Zheng et al., 2024; Du et al., 2018; Varga et al., 2022; Yu et al., 2020].

AcademicVideoPoint and AcademicVideoTrack (curated). For pointing, we convert existing object tracking annotations from three datasets, BURST [Athar et al., 2023], LV-VIS [Wang et al., 2023a] and OVIS [Qi et al., 2022], into 12k pointing and counting QAs. We first obtain the timestamp of the first frame in which an object appears and then randomly sample a point in the object’s mask with a Gaussian distribution around the mask center. For tracking, we repurpose 7 existing Ref-VOS datasets [Khoreva et al., 2018; Seo et al., 2020; Ding et al., 2023; Athar et al., 2023; Wang et al., 2023a; Yan et al., 2024; Athar et al., 2025] to obtain point tracking supervision data. In addition, we process 11 bounding-box based tracking datasets [Zhang et al., 2024a; Huang et al., 2019; Peng et al., 2024; Muller et al., 2018; Hong et al., 2023; Fan et al., 2019; Zhang et al., 2023a; Lamdouar et al., 2020; Wang et al., 2021, 2025c; Zhu et al., 2023] by using SAM-2 to generate segmentation masks and corresponding point tasks.

Molmo2-MultiImageQA (human). We collect QA data on semantically related image sets to support real-world multi-image queries. We form image sets by grouping images whose captions (generated by a PixMo-Cap-trained model) have high sentence-level similarity; each set contains 2–6 images (2.83 on average). Human annotators then write questions over each set, and answers are refined through the same human–LLM loop as above. In total, we construct 47k image sets from 105k unique images and 75k QA pairs.

Molmo2-MultiImagePoint and -SynMultiImageQA (synthetic). To improve multi-image grounding, we curate over 470k pointing and counting samples from sentence-level label similarities in PixMo-Points. Each example contains 2–5 related images (3.24 on average). We normalize human-provided labels (lowercasing, punctuation/whitespace cleanup, basic synonym merging) and use an LLM to derive a single canonical sentence label shared across the set. These canonicalized labels are then used to point and count the same entities across multiple images. During training, we sample from the pre-canonicalized labels to preserve annotation diversity, while using canonicalized labels to organize the data. For Molmo2-SynMultiImageQA, we adapt CoSyn [Yang et al., 2025b] to create 188k synthetic multi-image examples with text-rich images such as charts, tables, and documents.

5.3.2 Training

This section provides an overview of our model and training pipeline. See the Appendix for additional details.

Architecture

Our model architecture follows the common design of combining a pre-trained LLM and a vision transformer (ViT) [Dosovitskiy et al., 2021] via a connector module [Deitke et al., 2025; Liu et al., 2024a]. Visual inputs are split or resized into fixed-size crops, which are encoded into patch-level features by the ViT. The patch-level features are then pooled, projected by the connector, and passed as visual tokens, along with any text inputs, to the LLM. Figure 5.2 provides an overview.

Cropping. For input images, we use a single crop of the down-scaled image as well as up to K overlapping crops tiling the image to allow higher-resolution processing [Deitke et al., 2025]. Images that cannot be tiled by K crops are downsampled. We use $K = 8$ during training and $K = 24$ during inference. For videos, we

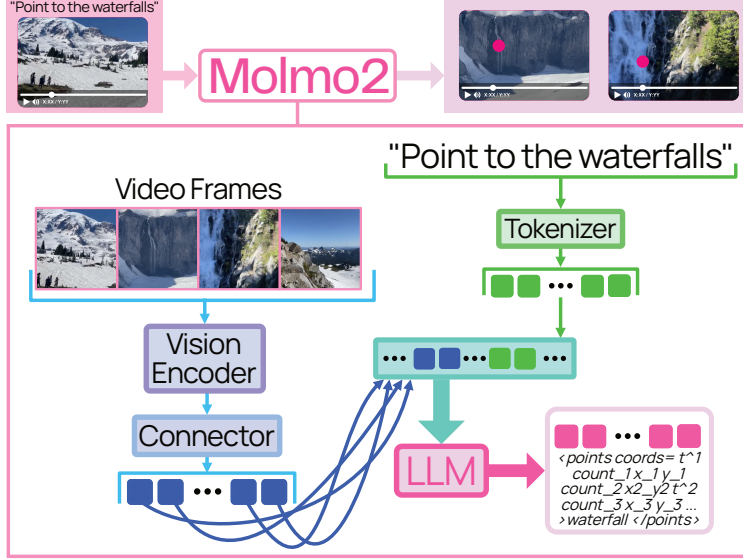


Figure 5.2: Molmo2 follows the standard design of connecting a vision encoder and a language model to process video inputs.

sample $F = 128$ frames and up to a maximum of $S = 2$ fps as single crops (downscaling if needed) to reduce computational costs when processing long videos. If the video length is longer than F/S , we use uniform sampling up to F frames. In both cases, the last frame is always included to retain information at the end of the video.

Vision-language connector. The connector uses features from the third-to-last and ninth-from-last ViT layers, following [Deitke et al., 2025]. For images, 2×2 patch windows are pooled into a single vector using a multi-headed attention layer, where the mean of the patches serves as the query. For video frames, a 3×3 patch window is used instead to reduce the token count (using the same parameters). Finally, the pooled features are projected using a shared MLP.

LLM. The LLM takes as input the visual tokens interleaved with text timestamps (for videos) or image indices (for multi-image input). For multi-crop images, we include column tokens [Deitke et al., 2025] to indicate the image’s aspect ratio. We do not include column-tokens for single-crop images since they are always square. We also add image and frame start tokens and include subtitles (marked with text timestamps) as text after the visual input if available. We allow image tokens (even if they are from different frames/images) to forward-attend to one another [Gao et al., 2025; Team et al., 2025], which we find can increase performance.

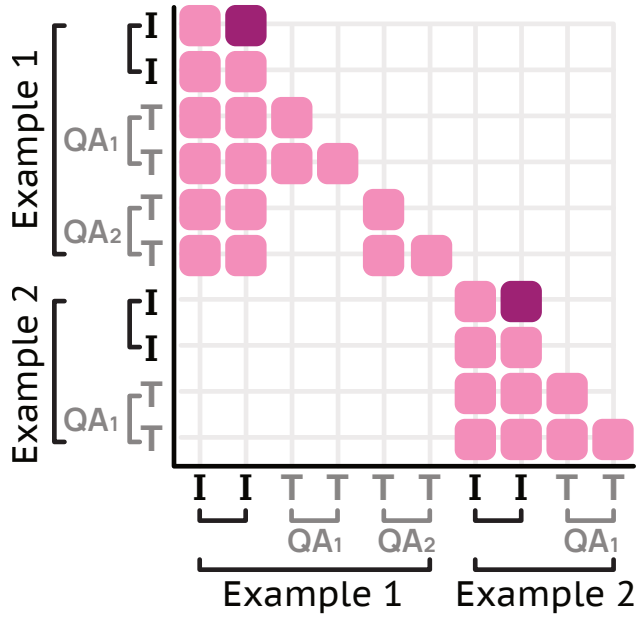


Figure 5.3: Attention mask for a *packed* sequence with two examples. The first contains two QA pairs for one image. Frame tokens (dark pink) have forward attention, while masking blocks cross-attention between different examples (lower-left empty block) and between distinct QA pairs within the same example (upper empty block).

Training

We use a simple two-stage design: a light-weight image-only pre-training stage and a joint video/image supervised fine-tuning (SFT) stage. We train on the Molmo2 data, image data from Pixmo, and various open-source datasets.

Pre-training. Our pre-training stage includes dense captioning with length conditioning and transcript prediction using PixMo-Cap, following [Deitke et al., 2025]. We add NLP data using the supervised fine-tuning data from Tulu [Lambert et al., 2025], filtered to remove non-English content and code, to better preserve language capabilities. Additionally, we add pointing data from PixMo-Points, PixMo-Count, and CoSyn-Point [Yang et al., 2025b]. We find that adding pointing data during pre-training leads to better pointing performance. We use 60% captioning, 30% image pointing, and 10% natural language for the mixing ratios. We train for 32k steps with a batch size of 128, which results in about 4 epochs of training on PixMo-Cap. All parameters are fine-tuned, and we use separate learning rates for the ViT, connector and LLM following [Deitke et al., 2025].

SFT. Our data mixture combines PixMo [Deitke et al., 2025], the Molmo2 datasets, Tulu, and other open-source video and image datasets. We sample datasets proportionally to the square root of each dataset size, with the addition of some manual rebalancing, such as downsampling large synthetic datasets. We train for 25k steps with a batch size of 128 and max sequence length of 16384.

Pointing and tracking. We represent point coordinates with a compressed plain-text format that includes normalized x and y coordinates, a timestamp (for video) or an image index (for images), and an integer ID that is unique for each distinct object to enable tracking and counting. Points are sorted based on time/image index, then x, y coordinates. During SFT, we use a maximum of 24 crops instead of 8 for 30% of images with pointing annotations to ensure that pointing can generalize to high-resolution images. For tracking, we also add auxiliary tasks of predicting only the first and last frames in which the objects appear, or tracking from an input query and point.

Token weighting. Our data includes both multiple-choice questions with a single output token and long video captions with 4,000+ output tokens. These long-output examples can easily become the large majority of loss tokens even if they are sampled rarely, which can cause degradation on short-answer or multiple-choice tasks. As a solution, we downweight the loss on individual tokens by $\frac{1}{\sqrt{n}}$ where n is the number of answer tokens, which better balances long and short output training examples in the loss. For pointing examples, we use a fixed weight of 0.2 to avoid biasing the model towards examples with a small number of points.

Packing. Examples can also have anywhere from 16k (videos with subtitles) to hundreds (pure-text or small images) of tokens. To avoid wasteful padding when creating training batches, we use packing to merge multiple short examples into a single long sequences. Packing is non-trivial for vision-language models due to the need to efficiently pack both crops for the ViT and tokens for the LLM, and the need to support models with different approaches to converting images/videos into tokens. We develop an on-the-fly packing algorithm that builds maximally efficient packed sequences from a small pool of in-memory examples and can be integrated into standard PyTorch data loaders.

Message trees. We encode videos and images with multiple annotations as *message-trees*. The visual input is encoded as the first message, and each annotation becomes a different branch. The tree is linearized as a single sequence with a custom attention mask to prevent branches from cross-attending to each other. On average, examples in our data have 4 annotations, and packing is able to fit 3.8 examples into a 16348 token sequence, leading to 15x training efficiency. Figure 5.3 shows the attention masking.

Model	NextQA test	PerceptionTest test	MYBench test	Tomato test	MotionBench val	TempCompass test MCQ	Video-MME test	Video-MME- Sub test	LongVideoBench val	MLVU test MCQ	LYBench test	VideoEvalPro test	Ego Schema test	Molmo2 Caption test F1 Score	Molmo2 Count val accuracy	Short QA avg.	Long QA avg.	Average	Elo Score	Elo Rank
API call only																				
GPT-5	86.3	79.4	74.1	53.0	65.4	80.4	83.3	86.9	72.6	77.7	65.2	68.8	75.6	50.1	35.8	73.1	76.3	70.6	1037	8
GPT-5 mini	83.2	72.0	66.5	44.1	59.9	74.9	77.3	82.3	69.7	69.1	54.7	60.1	70.9	56.6	29.8	66.8	69.8	65.0	1082	3
Gemini 2.5 Pro	85.3	78.4	70.6	48.6	62.0	81.9	87.8	87.8	76.8	81.5	75.7	78.4	72.2	42.1	35.8	71.1	80.4	71.2	1099	1
Gemini 2.5 Flash	81.8	74.7	67.0	39.1	59.3	80.2	84.2	84.2	73.1	75.1	64.9	69.6	70.2	46.0	31.9	67.0	74.5	66.7	1088	2
Claude Sonnet 4.5	79.2	64.3	62.1	39.6	58.5	72.8	74.2	80.5	65.1	64.0	50.5	50.5	73.1	26.0	27.2	62.8	66.4	59.6	1014	10
Open weights only																				
InternVL3.5-4B	80.3	68.1	71.2	26.8	56.5	68.8	65.4	68.6	60.8	52.0	43.2	46.5	58.9	7.7	26.3	62.0	56.5	53.4	941	17
InternVL3.5-8B	81.7	72.7	72.1	24.6	56.6	70.3	66.0	68.6	62.1	53.2	43.4	48.1	58.6	7.8	26.1	63.0	57.1	54.1	948	16
Qwen3-VL-4B	81.4	70.7	68.9	31.8	58.6	70.8	69.3	74.0	62.8	58.4	56.2	49.8	68.4	25.2	25.3	63.7	62.7	58.1	1053	6
Qwen3-VL-8B	83.4	72.7	68.7	35.7	56.9	74.3	71.4	75.2	62.4	57.6	58.0	50.3	<u>69.8</u>	26.7	29.6	65.3	<u>63.5</u>	59.5	1059	4
Keye-VL-1.5-8B	75.8	64.2	56.9	33.0	55.1	75.5	73.0	76.2	66.0	53.8	42.8	<u>54.9</u>	56.3	25.4	27.2	60.1	60.4	55.7	957	15
GLM-4.1V-9B	81.3	74.2	68.4	30.0	59.0	72.3	68.2	75.6	65.7	56.6	44.0	51.1	62.6	18.4	26.6	64.2	60.5	56.9	967	12
MiniCPM-V-4.5-8B	78.8	70.9	60.5	29.8	59.7	72.7	67.9	73.5	63.9	60.6	50.4	<u>54.9</u>	49.6	29.3	26.3	62.1	60.1	56.6	983	11
Eagle2.5-8B	85.0	81.0	74.8	31.0	55.7	<u>74.4</u>	<u>72.4</u>	<u>75.7</u>	<u>66.4</u>	60.4	50.9	58.6	72.2	22.8	28.9	67.0	65.2	60.7	1028	9
Open models																				
PLM-3B	83.4	79.3	74.7	30.9	60.4	69.3	54.9	59.4	57.9	48.4	40.4	46.2	66.9	12.3	24.4	66.3	53.5	53.9	849	19
PLM-8B	84.1	82.7	77.1	33.2	61.4	72.7	58.3	65.4	56.9	52.6	44.5	47.2	68.8	10.9	26.6	68.5	56.2	56.2	860	18
LLaVA-Video-7B	83.2	68.8	58.6	24.9	54.2	66.6	63.3	69.7	58.2	52.8	44.2	47.8	57.3	19.9	21.4	59.4	56.2	52.7	963	13
VideoChat-Flash-7B	<u>85.5</u>	76.5	74.0	32.5	60.6	69.4	65.3	69.7	64.7	56.0	48.2	51.2	51.3	14.8	21.6	66.4	58.1	56.1	962	14
Molmo2 family: Open weights, Open data (no distillation), Open code																				
Molmo2-4B	85.2	81.7	75.5	<u>38.3</u>	<u>62.6</u>	72.5	67.6	73.1	67.3	63.7	47.8	<u>54.9</u>	59.5	<u>31.8</u>	<u>30.4</u>	<u>69.3</u>	62.0	<u>60.8</u>	1051	7
Molmo2-8B	86.3	<u>82.5</u>	<u>76.3</u>	38.4	62.7	72.5	68.8	74.3	65.9	<u>61.6</u>	49.6	54.3	61.6	33.3	31.5	69.8	62.3	61.3	1059	4

Table 5.1: Video benchmark results for a range of proprietary APIs, open-weight baselines, video-specialized models, and our Molmo2 family across video understanding, captioning, and counting benchmarks. The result of the best-performing open-weight model is in **bold**.

5.4 Evaluation

We evaluate Molmo2 on standard video academic benchmarks and on our new benchmarks for video captioning, counting, and pointing, as well as a large-scale human-preference study. Then we report results for ablations, task-specific Molmo2 variants, and test-time scaling. See the Appendix for additional evaluations on single image, multi-image, NLP benchmarks, additional discussion, and results with the fully open OLMo [Walsh et al., 2025] language model.

5.4.1 Overall results

We evaluate captioning by constructing Molmo2-CapTest, an eval set of 693 creative-commons licensed videos with at least four human-annotated captions. We use an LLM-as-a-judge to compute precision, recall, and F1 for statements made in the model’s caption relative to statements from the annotator’s captions, similar to Molmo’s image captioning metric [Deitke et al., 2025]. For counting, we construct Molmo2-VideoCount by using our Molmo2-VideoPoint pipeline to collect 533 diverse examples that cover object, action, and

animal queries with up to 60 points.

For the human preference study, we collect questions from human annotators and manually filter them to prioritize open-ended questions over straightforward ones, resulting in 450 questions. We add another 51 videos for captioning queries. We sample two model outputs and gather pairwise preferences on them from annotators. We collect over 85K ratings (501 per model pair). From this data, we calculate an Elo ranking using the Bradley-Terry model [Chiang et al., 2024].

We obtain results for all models on all tasks. We prioritize author-published results but fill in missing results with the best previously reported values from technical reports or papers. If data is still missing, we compute it ourselves. We try to follow the author’s eval setup but note that eval details (e.g. , prompting or number of frames) are sometimes not public, so results should be interpreted carefully.

During inference, we use 128 frames and greedy decoding for all tasks; for the human evaluations, we use $\text{top_p}=0.95$, $\text{temperature}=0.7$, and $\text{frequency_penalty}=0.1$, which produces more natural results.

Results are in Table 5.1; we highlight a few key takeaways:

- Molmo2 is SoTA on short video benchmarks, captioning, and counting among non-proprietary models
- Molmo2 outperforms previous fully-open models but lags behind the best open-weight models. We believe this is due to a lack of open-source long (10+ minutes) training data, and computational limitations that made it challenging to run ultra-long context training.
- Molmo2 ranks equal or better than other open-weight models on human preference, and is far ahead of previous fully-open models.

5.4.2 Grounding results

Video counting and pointing. For counting, we also evaluate on BURST-VideoCount, a counting benchmark derived from the ground-truth tracks in the BURST test set [Athar et al., 2023]. We report the close accuracy metric (correct if $|pred - gt| \leq \Delta$, where $\Delta = 1 + \lfloor 0.05 \times gt \rfloor$), which rewards being close to the correct answer. For pointing, we build Molmo2-VideoPointTest (Molmo2-VP) by running SAM 2 [Ravi et al., 2025] to gather object segmentation masks within a 3-second window from the annotated spatial-temporal points in Molmo2-VideoPoint, and manually filter out examples with incorrect masks, leaving a total of 137 examples.

Model	BURST VC (test)		Molmo2-VC		Molmo2-VP
	Acc.	Close acc.	Acc.	Close acc.	F1
<i>API call only</i>					
GPT-5	43.1	73.7	35.8	50.3	2.9
GPT-5 mini	46.0	73.0	29.8	49.3	3.6
Gemini 2.5 Pro	41.6	70.0	35.8	56.5	17.0
Gemini 2.5 Flash	38.7	70.0	<u>31.9</u>	48.2	8.4
Claude Sonnet 4.5	42.4	72.6	27.2	45.1	2.9
<i>Open weights only</i>					
Qwen3-VL-4B	38.9	74.7	25.3	44.3	0.0
Qwen3-VL-8B	42.0	74.4	29.6	47.7	1.5
<i>Molmo2 family: Open weights, Open data, Open code</i>					
Molmo2-4B	<u>61.6</u>	<u>76.0</u>	30.4	<u>52.4</u>	<u>28.4</u>
Molmo2-8B	62.8	77.5	31.5	52.3	32.9

Table 5.2: Video counting and pointing results. Molmo2 scores highest on BURST-VC and Molmo2-VP and second highest on Molmo2-VC’s close accuracy, slightly behind Gemini 2.5 Pro.

For video pointing, we report F1 metric measuring how well the generated points match the ground-truth masks.

Results are shown in Table 5.2. Molmo2 is strong on the close metric, out-performing GPT 5. For Molmo2-VP, we carefully tune the prompts and try both point and bbox formats for our baseline models; however, we were unable to find a formulation that achieved very strong performance. Gemini Pro 2.5 reached the best score, but Molmo2 still significantly outperforms it.

Video object tracking. We evaluate video tracking on referring video object segmentation (VOS) benchmarks, where a point is considered correct if it lies within the ground truth segmentation mask. We additionally introduce Molmo2-Track, a benchmark covering diverse domains with complex object movements and occlusions (see the Appendix). Following [Ahmad et al., 2025], we use SAM2 to convert point predictions to segmentation masks for evaluation. We report the Jaccard and F-measure ($\mathcal{J}\&\mathcal{F}$) metrics for measuring segmentation quality across all frames, and the F1 score for the points at 1 fps. For API models, we generate

Model	MeVIS	Ref-YT		Ref-Davis		ReasonVOS		Molmo2-Track	
	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	F1	$\mathcal{J}\&\mathcal{F}$	F1	$\mathcal{J}\&\mathcal{F}$	F1	$\mathcal{J}\&\mathcal{F}$	F1
GPT-5	23.4	23.2	21.9	24.1	22.8	19.0	16.0	13.2	9.1
GPT-5 mini	15.7	9.8	6.3	7.7	2.3	11.1	6.0	9.3	3.4
Gemini 2.5 Pro	40.7	51.4	44.9	62.3	59.3	51.6	49.8	36.1	33.8
Gemini 2.5 Flash	27.6	37.8	34.3	41.1	37.0	24.7	28.7	27.0	25.8
LISA-7B	–	50.2	–	58.4	–	31.4	–	–	–
TrackGPT-7B	–	56.4	–	63.2	–	–	–	–	–
VideoLISA	44.4	63.7	–	68.8	–	47.5	–	–	–
VideoGLaMM	45.2	66.8	–	69.5	–	33.9	–	–	–
VideoChat-TPO	47.0	63.9	–	–	–	–	–	–	–
Molmo + SAM2	46.9	63.6	–	38.8	–	45.7	–	–	–
VideoMolmo-7B	53.9	67.3	–	<u>72.5</u>	–	51.1	–	–	–
Molmo2-4B	<u>59.5</u>	69.1	79.4	72.0	<u>79.4</u>	<u>61.0</u>	67.2	54.4	<u>49.9</u>
Molmo2-8B	60.0	69.1	79.4	72.7	80.7	62.6	<u>66.7</u>	54.4	50.0

Table 5.3: Video object tracking results. Molmo2 outperforms API-based and open models across all benchmarks. F1 scores are omitted for VOS models that only generate a single mask.

the bounding box and extract their center points as they fail to generate accurate points. Table 5.3 shows the results: 1) Molmo2 outperforms all baselines, including specialized segmentation models (in gray), across all benchmarks, particularly excelling on ReasonVOS and Molmo2-Track which require complex reasoning and occlusion handling skills. 2) Gemini 2.5 Pro is the strongest API model but still struggles to generate accurate object tracks.

5.4.3 Ablations and specialized models

We train specialized 4B models on subsets of our data and use them for ablations. Gray row shows specialized models with default settings; key takeaways are in the captions.

Video ablations . Table 5.4 shows results and ablations with video-only and video-captioning-only data. Molmo2-QA, Molmo2-Cap, and bi-directional attention and token-weighting all improve QA performance (5.4b, 5.4c). Captions based solely on human transcripts (V) produce worse results than those that include frame-level captions (VF), but training on a mixture of these captions does not lead to improvements (5.4d).

Data	QA avg.	Cap. F1
Molmo2-4B	66.7	31.8
Video-Only	64.8	32.6
Video-Cap. Only	-	<u>32.3</u>

(a) **Specialization.** Joint training with images improves results on video benchmarks but degrades video captioning.

Model	QA avg.	Cap. F1
bidir+weighting	64.8	<u>32.6</u>
no bidir	<u>64.4</u>	30.8
no weighting	64.0	34.0

(b) **Modeling.** Bidirectional attention significantly improves model performance, while token weighting degrades video captioning.

Data	QA avg.	Cap. F1
academic	62.9	4.7
+QA	64.5	17.2
+Cap	65.3	<u>30.3</u>
+Cap/QA	<u>64.8</u>	32.6

(c) **Video SFT data.** Both Molmo2-Cap and Molmo2-QA improve performance compared to academic datasets only.

Data	Cap. R	Cap. P	Cap. F1
V	13.2	68.5	22.1
VF	23.1	58.9	33.2
VF+V	22.3	59.5	<u>32.5</u>
VF+F	<u>22.4</u>	59.4	<u>32.5</u>
VF+V+F	22.1	<u>59.8</u>	32.3

(d) **Caption data.** Using the video and frame merged caption (VF) is critical, but adding video (V) and/or frame (F) captions does bring improvements.

Table 5.4: Video ablations. For ablations (a)(b)(c) we train models on only video data; ablation (d) has models with only video captions.

Model	BVC	MVC	MVP Strategy	BVC	MVC
Molmo2-4B	61.6	30.4	28.4 count	61.3	28.1
Pointing-Only	61.5	34.5	31.8 point then count	61.5	34.5

(a) **Specialization.** Training a specialized Point-Only model on video pointing data performs better than joint training.

(b) **Counting strategy.** Pointing is the key ingredient in Molmo2’s counting abilities.

Data	BVC	MVC	MVP	Upsampling	BVC	MVC	MVP
both	<u>61.5</u>	34.5	<u>31.8</u>	med-high	61.5	34.5	31.8
Molmo2-VP	60.0	<u>34.3</u>	35.0	no	62.4	32.1	28.1
Academic-VP	61.6	9.0	9.0				

(c) **Data source.** Including both Molmo2- and AcademicVideoPoints performs the best overall.

(d) **Sampling strategy.** Upsampling medium and high-count examples helps on MVC and MVP.

Table 5.5: Counting and pointing ablations. BVC represents Burst-VideoCount accuracy; and MVC and MVP are Molmo2-VideoCount accuracy and Molmo2-VideoPoint F1 on the validation sets.

Video counting and pointing. Table 5.5 reports the performance of a specialized pointing model and ablating counting strategy, data, and data sampling. We observe that our two sources of pointing are complementary (Table 5.5c), that pointing before counting is much better than directly predicting the count (Table 5.5b), and that upsampling high-frequency points improves both counting and pointing (Table 5.5d). The specialized model is better than the joint model (Table 5.5a) despite Molmo2-VideoPoint being upsampled, suggesting this task is hard to learn in the joint setting.

Video object tracking. Table 5.6 shows ablations on task mixtures and data sources. Specialized tracking model outperforms the joint model while adding counting improves performance (Table 5.6a), subsequently

adding bounding box tracks and the Molmo2-VideoTrack dataset leads to improvements (Table 5.6b), and supporting temporal grounding from the same annotations helps while adding point-based single object tracking causes slight degradation (Table 5.6c).

Model	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	Data	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	Strategy	$\mathcal{J}\&\mathcal{F}$	F1	HOTA
Molmo2-4B	64.7	69.2	66.8	Academic (VOS)	64.3	68.8	66.7	Track	64.2	68.4	66.2
Tracking-Only	64.9	70.0	68.4	+Academic (bbox)	63.9	69.3	67.5	+ Grounding	64.8	69.4	67.2
Tracking+Point	65.7	71.1	69.4	+Molmo2 (VideoTrack)	64.9	70.0	68.4	+ Single-point	64.3	68.8	66.7

(a) **Specialization.** Specializing on tracking outperforms the joint model. Combining pointing helps performance.

(b) **Tracking data source.** We see progressive improvements from academic VOS, bounding box (bbox) tracks, to Molmo2 data.

(c) **Tracking sub-tasks** ablated on Academic VOS only. Temporal grounding helps, while single-point object tracking slightly degrades performance.

Table 5.6: Tracking ablations. We report average metrics across the five tracking benchmarks (the valid-u split for MeViS). HOTA measures association accuracy.

Image ablations. We train a 4B model using only images and text data and achieve an average of 79.8 on the 11-benchmark validation set from Molmo [Deitke et al., 2025], compared to 80.7 for Molmo2-4B, showing a positive transfer from videos. Both surpass Molmo-7B-D (76.9). See the Appendix for more comparisons and multi-image benchmarks.

5.5 Conclusion

This chapter presented Molmo2, a fully open vision-language model that demonstrates how grounded visual understanding—the central theme of this thesis—can unlock powerful multimodal reasoning capabilities. By constructing 9 novel datasets without proprietary distillation, introducing training innovations such as token weighting, sequence packing, and message trees, and enabling open-vocabulary pointing and tracking across images, multi-image contexts, and videos, Molmo2 achieves state-of-the-art results on video grounding tasks while remaining competitive on standard benchmarks.

Molmo2 serves as the culmination of the thesis arc: reliable evaluation (Chapter 2) identifies where models fail at grounding; richer supervision (Chapter 3) teaches models to attend to visual details; object-centric representations (Chapter 4) align the model’s visual interface with scene structure; and Molmo2 (this chapter) shows that when all these elements come together, models can support spatially precise, temporally grounded reasoning that goes beyond surface-level visual-language alignment.

Chapter 6

Conclusion

This thesis studied a central question in vision-language modeling: how to move from impressive surface-level performance toward grounded visual understanding. While recent vision-language models can produce fluent descriptions and strong benchmark results, these outcomes do not always reflect a genuine ability to connect language to the visual world. A model may answer correctly without attending to the right evidence, may rely on shortcuts in data, or may fail when asked to localize, track, or reason precisely about objects and events. The core argument of this thesis is that progress in multimodal AI should be measured not only by generative quality or aggregate accuracy, but by whether models are grounded in the entities, structure, and dynamics present in images and videos.

To advance this goal, this thesis approached grounded visual understanding from four complementary directions: evaluation, supervision, representation, and reasoning.

First, this thesis showed that reliable progress requires better ways to measure grounding. Existing benchmarks often make it difficult to distinguish true visual understanding from the exploitation of annotation artifacts or dataset biases. To address this, the thesis introduced evaluation methods that more directly test whether models align language with visual evidence, and benchmark construction methods that allow more systematic and fine-grained analysis across diverse tasks. These contributions support a broader shift in evaluation: rather than treating visual understanding as a single number on a leaderboard, it should be studied as a collection of grounded capabilities with distinct failure modes.

Second, this thesis demonstrated that stronger grounding can be learned from richer supervision. Instead

of relying only on loosely aligned image-text pairs, the work explored instruction data derived from structured visual information, including objects, attributes, relations, segmentation, and depth. This type of supervision injects explicit grounding signals into training and helps bridge the gap between generative multimodal modeling and spatially faithful understanding. The resulting gains suggest that data quality and data structure matter as much as data scale, especially when the target capability involves localization, fine-grained reference, and visual reasoning.

Third, this thesis argued that grounded understanding depends not only on training data and objectives, but also on the visual units through which models perceive the world. Standard patch-based tokenization provides a general-purpose representation, but it is often poorly matched to object persistence, motion, and compositional structure in video. By developing object- and trajectory-centric representations, this thesis showed that more grounded tokenizations can better preserve visual structure and improve both efficiency and downstream performance. This points to an important lesson: representation design remains a fundamental lever for multimodal learning, particularly when the goal is not just recognition, but coherent understanding over time.

Finally, this thesis showed that grounding is not merely a perception problem; it is a foundation for multimodal reasoning. A model that cannot reliably identify what is where, or how entities evolve across frames, will struggle to point, track, compare, or reason spatially. By developing models, datasets, and training strategies for these settings, the thesis connected grounding to more advanced capabilities such as video pointing, tracking, and spatially precise multimodal reasoning. These results suggest that grounding is not an isolated module to be added after the fact, but an organizing principle that supports more robust reasoning in open-world visual environments.

Taken together, the contributions of this thesis support a unified view of grounded visual understanding. Reliable multimodal systems should do more than generate plausible language about visual content. They should identify the relevant entities in a scene, connect expressions to those entities faithfully, preserve structure across space and time, and support reasoning that remains anchored to visual evidence. Evaluation, supervision, representation, and reasoning are often studied separately, but this thesis shows that they are deeply connected. Better evaluation reveals the need for richer supervision. Richer supervision motivates more appropriate representations. Better representations enable stronger grounded reasoning. Progress

emerges when these pieces are developed together.

At the same time, grounded visual understanding remains far from solved. Current models still struggle in challenging real-world settings involving ambiguity, long-tail concepts, cluttered scenes, fine-grained spatial relations, long-horizon temporal dependencies, and complex multi-entity interactions. They remain sensitive to annotation artifacts, inconsistencies in supervision, and mismatches between benchmark success and real capability. Even when grounding is improved, models may still lack robustness, calibration, or interpretability in deployment settings. These limitations highlight that grounding should not be treated as a binary property that a model either has or does not have, but as an ongoing research target requiring advances in data, architectures, objectives, and evaluation methodology.

Several directions follow naturally from this thesis.

One important frontier is *visual grounding for spatial intelligence*. The grounding capabilities explored in this thesis operate primarily in 2D image and video space, but real-world understanding is fundamentally three-dimensional. Extending grounded visual understanding to 3D—reasoning about the volumetric layout of a scene, the spatial relationships between objects in depth, and the physical affordances of surfaces and structures—is critical for applications in robotics, augmented reality, autonomous navigation, and embodied AI. A spatially intelligent model should not only point to an object in a frame, but understand where that object sits in 3D space relative to the viewer and other entities, how it can be grasped or manipulated, and how the scene would look from a different viewpoint. Grounding in 3D space demands new representations, new supervision signals (e.g., from depth sensors, multi-view geometry, or simulation), and new evaluation protocols that go beyond 2D localization metrics.

Another pressing challenge is *grounding in long-form video*. While this thesis demonstrated progress in video pointing and tracking, most current models and benchmarks operate over relatively short clips. In practice, many real-world video understanding tasks require finding and grounding specific moments, objects, or events within hours of footage—a “needle in a haystack” problem. A surveillance system must locate a specific incident across a full day of recording; an analyst must find a particular play in a full-length sports broadcast; a user must retrieve a specific conversation from a lengthy meeting video. Grounding in such long-horizon settings demands models that can efficiently search, attend to, and reason over vast temporal contexts without losing the precision required to localize fine-grained events. Current approaches that uniformly

sample frames or truncate context are fundamentally limited for these tasks, calling for new architectures, hierarchical retrieval strategies, and streaming approaches that maintain grounded understanding at scale.

Beyond these two directions, future work should also develop training pipelines that unify generative supervision with richer geometric, temporal, and referential structure at scale, and continue rethinking visual representations so that the units of computation better reflect the objects, regions, trajectories, and relations that matter for downstream reasoning. More broadly, future work should aim to close the gap between multimodal pattern recognition and multimodal world understanding.

In this sense, grounded visual understanding is both a scientific goal and a practical one. Scientifically, it provides a clearer lens for studying what vision-language models truly learn. Practically, it is essential for building systems that can be trusted in applications that require precision, interpretability, and interaction with the real world. Whether the setting is assistive technology, robotics, scientific analysis, search, or human-AI collaboration, the usefulness of a multimodal system depends on whether its outputs are tied to the right visual evidence.

This thesis argues that grounding should be placed at the center of multimodal AI research. By improving how grounded capabilities are measured, how they are taught, how visual information is represented, and how reasoning is built on top of perception, this thesis takes several steps toward that goal. The broader message is simple: vision-language models should not only speak fluently about the world, but understand it in a way that is visually faithful, spatially precise, and temporally coherent. That is the path toward more reliable and more capable multimodal systems.

Bibliography

- Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*.
- Manoj Acharya, Kushal Kafle, and Christopher Kanan. 2019. TallyQA: Answering complex counting questions. In *AAAI*.
- Ghazi Shazan Ahmad, Ahmed Heakl, Hanan Gani, Abdelrahman Shaker, Zhiqiang Shen, Fahad Shahbaz Khan, and Salman Khan. 2025. Videomolmo: Spatio-temporal grounding meets pointing. *arXiv preprint arXiv:2506.05336*.
- Anonymous et al. 2024. Directsam: Fast and accurate object segmentation with a minimal pipeline. *arXiv preprint arXiv:2402.14327*.
- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736.
- Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.
- Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Deep compositional question answering with neural module networks. In *CVPR*, volume 1, page 3.
- Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021.

- Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846.
- Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zhohus, et al. 2025. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*.
- Ali Athar, Xueqing Deng, and Liang-Chieh Chen. 2025. Vicas: A dataset for combining holistic and pixel-level video understanding using captions with grounded segmentation. In *CVPR*.
- Ali Athar, Jonathon Luiten, Paul Voigtlaender, Tarasha Khurana, Achal Dave, Bastian Leibe, and Deva Ramanan. 2023. Burst: A benchmark for unifying object recognition, segmentation and tracking in video. In *WACV*.
- Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.
- Dina Bashkirova, Arijit Ray, Rupayan Mallick, Sarah Adel Bargal, Jianming Zhang, Ranjay Krishna, and Kate Saenko. 2023. Lasagna: Layered score distillation for disentangled object relighting. *arXiv preprint arXiv:2312.00833*.
- M Bellver, C Ventura, C Silberer, I Kazakos, J Torres, and X Giro-i Nieto. 2020. Refvos: a closer look at referring expressions for video object segmentation. *arXiv preprint arXiv:2010.00263*.
- Gedas Bertasius, Heng Wang, and Lorenzo Torresani. 2021. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*.
- Federico Bianchi, Pratyusha Kalluri, Esin Durmus, Faisal Ladhak, Myra Cheng, Debora Nozza, Tatsunori Hashimoto, Dan Jurafsky, James Zou, and Aylin Caliskan. 2022. Easily accessible text-to-image generation amplifies demographic stereotypes at large scale. *arXiv preprint arXiv:2211.03759*.
- Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token merging: Your vit but faster. In *ICLR*.

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Shyamal Buch, Cristóbal Eyzaguirre, Adrien Gaidon, Jiajun Wu, Li Fei-Fei, and Juan Carlos Niebles. 2022. Revisiting the " video" in video-language understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2917–2927.
- Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Paola Cascante-Bonilla, Khaled Shehada, James Seale Smith, Sivan Doveh, Donghyun Kim, Rameswar Panda, Gül Varol, Aude Oliva, Vicente Ordonez, Rogerio Feris, and Leonid Karlinsky. 2023. Going beyond nouns with vision & language models using synthetic data.
- Paola Cascante-Bonilla, Hui Wu, Letao Wang, Rogerio S Feris, and Vicente Ordonez. 2022. Simvqa: Exploring simulated environments for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5056–5066.
- Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. 2021. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3558–3568.
- Delong Chen, Samuel Cahyawijaya, Jianfeng Liu, Baoyuan Wang, and Pascale Fung. 2024a. Subobject-level image tokenization. *arXiv preprint arXiv:2402.14327*.
- Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. 2024b. Are we on the right way for evaluating large vision-language models? *CoRR*, abs/2403.20330.
- Lin Chen, Xilin Wei, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Bin Lin, Zhenyu Tang, Li Yuan, Yu Qiao, Dahua Lin, Feng Zhao, and Jiaqi Wang. 2024c. Sharegpt4video:

- Improving video understanding and generation with better captions. In *NeurIPS Track on Datasets and Benchmarks*.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is chatgpt’s behavior changing over time? *arXiv preprint arXiv:2307.09009*.
- Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, et al. 2024d. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13320–13331.
- Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. 2022. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*.
- Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. 2022. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299.
- Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. 2024a. Putting the object back into video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3151–3161.
- Ho Kei Cheng and Alexander G Schwing. 2022. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 640–658. Springer.
- Long Cheng, Jiafei Duan, Yi Ru Wang, Haoquan Fang, Boyang Li, Yushan Huang, Elvis Wang, Ainaz Eftekhari, Jason Lee, Wentao Yuan, et al. 2025. Pointarena: Probing multimodal grounding through language-guided pointing. *arXiv preprint arXiv:2505.09990*.
- Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. 2024b. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16901–16911.

- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating LLMs by human preference. In *ICML*.
- Jaemin Cho, Abhay Zala, and Mohit Bansal. 2022. Dall-eval: Probing the reasoning skills and social biases of text-to-image generative transformers. *arXiv preprint arXiv:2202.04053*.
- Jang Hyun Cho, Andrea Madotto, Effrosyni Mavroudi, Triantafyllos Afouras, Tushar Nagarajan, Muhammad Maaz, Yale Song, Tengyu Ma, Shuming Hu, Hanoona Rasheed, Peize Sun, Po-Yao Huang, Daniel Bolya, Suyog Jain, Miguel Martin, Huiyu Wang, Nikhila Ravi, Shashank Jain, Temmy Stark, Shane Moon, Babak Damavandi, Vivian Lee, Andrew Westbury, Salman Khan, Philipp Krähenbühl, Piotr Dollár, Lorenzo Torresani, Kristen Grauman, and Christoph Feichtenhofer. 2025. Perceptionlm: Open-access data and models for detailed visual understanding. *arXiv preprint arXiv:2504.13180*.
- Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, László A. Jeni, and Kris M. Kitani. 2025a. Accelerating vision transformers with adaptive patch sizes. *arXiv preprint arXiv:2510.18091*.
- Rohan Choudhury et al. 2025b. Don’t look twice: Faster video transformers with run-length tokenization. *arXiv:2503.00000*.
- Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Rohun Tripathi, Sangho Lee, Mohammadreza Salehi, Jason Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Ali Farhadi, and Ranjay Krishna. 2025. Molmo 2: Open weights and open data for state-of-the-art video and image models. Technical Report.
- Blender Online Community. 2018. Blender - a 3d modelling and rendering package.
- OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. <https://github.com/open-compass/opencompass>.
- MJ Cresswell. 1973. Logics and languages.
- Chenhang Cui, Yiyang Zhou, Xinyu Yang, Shirley Wu, Linjun Zhang, James Zou, and Huaxiu Yao. 2023. Holistic analysis of hallucination in gpt-4v(ision): Bias and interference challenges.

- Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2024. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 36.
- Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. 2013. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2634–2641.
- Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. 2025. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*.
- Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. 2024. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36.
- Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2023. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153.
- Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. 2020. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale

- hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, and Chen Change Loy. 2023. Mevis: A large-scale benchmark for video segmentation with motion expressions. In *ICCV*.
- Henghui Ding, Kaining Ying, Chang Liu, Shuting He, Xudong Jiang, Yu-Gang Jiang, Philip HS Torr, and Song Bai. 2025. Mosev2: A more challenging dataset for video object segmentation in complex scenes. *arXiv preprint arXiv:2508.05630*.
- Anuj Diwan, Layne Berry, Eunsol Choi, David Harwath, and Kyle Mahowald. 2022. Why is winoground hard? investigating failures in visuolinguistic compositionality. *arXiv preprint arXiv:2211.00768*.
- Thi-Thu-Trang Do, Quyet-Thang Huynh, Kyungbaek Kim, and Van-Quyet Nguyen. 2025. A survey on video big data analytics: architecture, technologies, and open research challenges. *Applied Sciences*.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.
- Sivan Doveh, Assaf Arbelle, Sivan Harary, Eli Schwartz, Roei Herzig, Raja Giryes, Rogerio Feris, Rameswar Panda, Shimon Ullman, and Leonid Karlinsky. 2023. Teaching structured vision & language concepts to vision & language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2657–2668.
- Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. 2018. The unmanned aerial vehicle benchmark: Object detection and tracking. In *ECCV*.
- Gamaleldin Elsayed, Aravindh Mahendran, Sjoerd Van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. 2022. Savi++: Towards end-to-end object-centric learning from real-world videos. *Advances in Neural Information Processing Systems*, 35:28940–28954.

- Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. 2019. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*.
- Li Fei-Fei, Rob Fergus, and Pietro Perona. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*.
- Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. 2024a. Mme: A comprehensive evaluation benchmark for multimodal large language models.
- Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. 2024b. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*.
- Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. 2024c. Blink: Multimodal large language models can see but not perceive. *arXiv preprint arXiv:2404.12390*.
- Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruva Ghosh, Jieyu Zhang, et al. 2024. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 36.
- Mona Gandhi, Mustafa Omer Gul, Eva Prakash, Madeleine Grunde-McLaughlin, Ranjay Krishna, and Maneesh Agrawala. 2022. Measuring compositional consistency for video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5046–5055.
- Mingze Gao, Jingyu Liu, Mingda Li, Jiangtao Xie, Qingbin Liu, Bo Zhao, Xi Chen, and Hui Xiong. 2025. Tc-llava: Rethinking the transfer from image to video understanding with temporal considerations. In *AAAI*.
- Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. 2018. Soccernet: A scalable dataset for action spotting in soccer videos. In *CVPR Workshop on Computer Vision in Sports*.

- Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. 2017. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850.
- Madeleine Grunde-McLaughlin, Ranjay Krishna, and Maneesh Agrawala. 2021. Agqa: A benchmark for compositional spatio-temporal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. 2018. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6047–6056.
- Agrim Gupta, Piotr Dollar, and Ross Girshick. 2019. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364.
- Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*.
- Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224. IEEE.
- Jiawei Han, Jian Pei, and Hanghang Tong. 2022. *Data mining: concepts and techniques*. Morgan kaufmann.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Roei Herzig, Elad Ben-Avraham, Karttikeya Mangalam, Amir Bar, Gal Chechik, Anna Rohrbach, Trevor Darrell, and Amir Globerson. 2022. Object-region video transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3148–3159.

- Lingyi Hong, Wenchao Chen, Zhongying Liu, Wei Zhang, Pinxue Guo, Zhaoyu Chen, and Wenqiang Zhang. 2023. Lvos: A benchmark for long-term video object segmentation. In *ICCV*.
- Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. 2024. Sugarcrepe: Fixing hackable benchmarks for vision-language compositionality. *Advances in neural information processing systems*, 36.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.
- Lianghua Huang, Xin Zhao, and Kaiqi Huang. 2019. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*.
- Xiaohu Huang, Hao Zhou, and Kai Han. 2024. Prunevid: Visual token pruning for efficient video large language models. *arXiv preprint arXiv:2412.16117*.
- Drew A Hudson and Christopher D Manning. 2019. 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.
- Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.
- Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. 2021. Openclip.
- Allan Jabri, Andrew Owens, and Alexei Efros. 2020. Space-time correspondence as a contrastive random walk. *Advances in neural information processing systems*, 33:19545–19560.

- Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR.
- Theo MV Janssen and Barbara H Partee. 1997. Compositionality. In *Handbook of logic and language*, pages 417–473. Elsevier.
- Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. 2020. 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.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.
- Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR.
- Dongfu Jiang, Xuan He, Huaye Zeng, Cong Wei, Max Ku, Qian Liu, and Wenhui Chen. 2024. MANTIS: Interleaved multi-image instruction tuning. *TMLR*.
- Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. 2025. Detect anything via next point prediction. *arXiv preprint arXiv:2510.12798*.
- Peng Jin, Ryuichi Takanobu, Caiwan Zhang, Xiaochun Cao, and Li Yuan. 2023. Chat-univi: Unified visual representation empowers large language models with image and video understanding. *arXiv preprint arXiv:2311.08046*.
- Justin Johnson, Agrim Gupta, and Li Fei-Fei. 2018. Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228.
- Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross

- Girshick. 2017. 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.
- Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678.
- Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher D Manning. 2021. Mind your outliers! investigating the negative impact of outliers on active learning for visual question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7265–7281.
- Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.
- Anna Khoreva, Anna Rohrbach, and Bernt Schiele. 2018. Video object segmentation with language referring expressions. In *ACCV*.
- Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. 2022. Conditional Object-Centric Learning from Video. In *International Conference on Learning Representations (ICLR)*.
- John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.
- Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026.
- Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017a. Dense-captioning events in videos. In *ICCV*.

- Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017b. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.
- Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.
- Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. 2022. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2025. Tülu 3: Pushing frontiers in open language model post-training. In *COLM*.
- Hala Lamdouar, Charig Yang, Weidi Xie, and Andrew Zisserman. 2020. Betrayed by motion: Camouflaged object discovery via motion segmentation. In *ACCV*.
- Thao Minh Le, Vuong Le, Svetha Venkatesh, and Truyen Tran. 2020. Hierarchical conditional relation networks for video question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9972–9981.
- Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *International Conference on Machine Learning*, pages 1078–1088. PMLR.
- Jie Lei, Tamara L Berg, and Mohit Bansal. 2022. Revealing single frame bias for video-and-language learning. *arXiv preprint arXiv:2206.03428*.
- Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L Berg, Mohit Bansal, and Jingjing Liu. 2021. Less is more: Clipbert for video-and-language learning via sparse sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7331–7341.

- Andrew Li, Rahul Thapa, Rahul Chalamala, Qingyang Wu, Kezhen Chen, and James Zou. 2025. SMIR: Efficient synthetic data pipeline to improve multi-image reasoning. *arXiv preprint arXiv:2501.03675*.
- Bo Li, Peiyuan Zhang, Kaichen Zhang, Fanyi Pu, Xinrun Du, Yuhao Dong, Haotian Liu, Yuanhan Zhang, Ge Zhang, Chunyuan Li, and Ziwei Liu. 2024a. Lmms-eval: Accelerating the development of large multimodal models.
- Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. 2023a. Seed-bench-2: Benchmarking multimodal large language models. *arXiv preprint arXiv:2311.17092*.
- Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. 2023b. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023c. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*.
- Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR.
- Kunchang Li, Yali Wang, Yanan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. 2024b. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Xinhao Li, Yi Wang, Jiashuo Yu, Xiangyu Zeng, Yuhao Zhu, Haian Huang, Jianfei Gao, Kunchang Li, Yanan He, Chenting Wang, Yu Qiao, Yali Wang, and Limin Wang. 2024c. Videochat-flash: Hierarchical compression for long-context video modeling. *arXiv preprint arXiv:2501.00574*.
- Liang Liang, Haihua Ma, Le Zhao, Xiaopeng Xie, Chengxin Hua, Miao Zhang, and Yonghui Zhang. 2024. Vehicle detection algorithms for autonomous driving: A review. *Sensors*.
- Josip Tomo Licardo, Mihael Domjan, and Tihomir Orehovački. 2024. Intelligent robotics—a systematic review of emerging technologies and trends. *Electronics*.

- Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. 2023. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*.
- Kevin Lin, Linjie Li, Chung-Ching Lin, Faisal Ahmed, Zhe Gan, Zicheng Liu, Yumao Lu, and Lijuan Wang. 2022. Swinbert: End-to-end transformers with sparse attention for video captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17949–17958.
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.
- Hong LinLin, Lee Sangheang, and Song GuanTing. 2024. Cam-vtrans: real-time sports training utilizing multi-modal robot data. *Frontiers in Neurorobotics*.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024a. Visual instruction tuning. *Advances in neural information processing systems*, 36.
- Jiacheng Liu, Wenya Wang, Dianzhuo Wang, Noah A Smith, Yejin Choi, and Hannaneh Hajishirzi. 2023a. Vera: A general-purpose plausibility estimation model for commonsense statements. *arXiv preprint arXiv:2305.03695*.
- Jihao Liu, Xin Huang, Jinliang Zheng, Boxiao Liu, Jia Wang, Osamu Yoshie, Yu Liu, and Hongsheng Li. 2024b. Mm-instruct: Generated visual instructions for large multimodal model alignment.
- Ruyang Liu, Haoran Tang, Haibo Liu, Yixiao Ge, Ying Shan, Chen Li, and Jiankun Yang. 2024c. Ppllava: Varied video sequence understanding with prompt guidance. *arXiv preprint arXiv:2411.02327*.
- Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2023b. Trustworthy llms: a survey and guideline for evaluating large language models’ alignment. *arXiv preprint arXiv:2308.05374*.

- Yangzhou Liu, Yue Cao, Zhangwei Gao, Weiyun Wang, Zhe Chen, Wenhai Wang, Hao Tian, Lewei Lu, Xizhou Zhu, Tong Lu, et al. 2024d. Mminstruct: A high-quality multi-modal instruction tuning dataset with extensive diversity. *arXiv preprint arXiv:2407.15838*.
- Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. 2023c. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*.
- Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. 2024e. Tempcompass: Do video llms really understand videos? *arXiv preprint arXiv: 2403.00476*.
- Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.
- Ziyu Liu, Tao Chu, Yuhang Zang, Xilin Wei, Xiaoyi Dong, Pan Zhang, Zijian Liang, Yuanjun Xiong, Yu Qiao, Dahua Lin, and Jiaqi Wang. 2024f. MMDU: A multi-turn multi-image dialog understanding benchmark and instruction-tuning dataset for LVLMS. In *NeurIPS Track on Datasets and Benchmarks*.
- Francesco Locatello, Dirk Weissenborn, and et al. 2020. Object-centric learning with slot attention. In *NeurIPS*.
- Wentao Ma, Weiming Ren, Yiming Jia, Zhuofeng Li, Ping Nie, Ge Zhang, and Wenhui Chen. 2025. Videoeval-pro: Robust and realistic long video understanding evaluation. *arXiv preprint arXiv:2505.14640*.
- Zixian Ma, Jerry Hong, Mustafa Omer Gul, Mona Gandhi, Irena Gao, and Ranjay Krishna. 2022. Crepe: Can vision-language foundation models reason compositionally? *arXiv preprint arXiv:2212.07796*.
- Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. 2024. Video-ChatGPT: Towards detailed video understanding via large vision and language models. In *ACL*.
- Kartikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. 2023. Egoschema: A diagnostic benchmark for very long-form video language understanding. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

- Jiaxu Miao, Xiaohan Wang, Yu Wu, Wei Li, Xu Zhang, Yunchao Wei, and Yi Yang. 2022. Large-scale video panoptic segmentation in the wild: A benchmark. In *CVPR*.
- Oscar Michel, Anand Bhattad, Eli VanderBilt, Ranjay Krishna, Aniruddha Kembhavi, and Tanmay Gupta. 2024. Object 3dit: Language-guided 3d-aware image editing. *Advances in Neural Information Processing Systems*, 36.
- Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106.
- Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229.
- John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.
- Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. 2018. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*.
- Shehan Munasinghe, Hanan Gani, Wenqi Zhu, Jiale Cao, Eric Xing, Fahad Shahbaz Khan, and Salman Khan. 2025. Videoglamm: A large multimodal model for pixel-level visual grounding in videos. In *CVPR*.
- Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. 2021. Video transformer network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3163–3172.
- Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. 2021. 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*.
- OpenAI. Terms of use. Accessed: 2024-11-01.

OpenAI. 2022. Chatgpt.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. GPT-4o mini system card.

Jae Sung Park, Zixian Ma, Linjie Li, Chenhao Zheng, Cheng-Yu Hsieh, Ximing Lu, Khyathi Chandu, Quan Kong, Norimasa Kobori, Ali Farhadi, Yejin Choi, and Ranjay Krishna. 2025. Synthetic visual genome. In *CVPR*.

Viorica Pătrăucean, Lucas Smaira, Ankush Gupta, et al. 2023. Perception test: A diagnostic benchmark for multimodal video models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Liang Peng, Junyuan Gao, Xinran Liu, Weihong Li, Shaohua Dong, Zhipeng Zhang, Heng Fan, and Libo Zhang. 2024. Vasttrack: Vast category visual object tracking. In *NeurIPS*.

Khoi Pham, Kushal Kafle, Zhe Lin, Zhihong Ding, Scott Cohen, Quan Tran, and Abhinav Shrivastava. 2022. Improving closed and open-vocabulary attribute prediction using transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Bryan A. Plummer, Liwei Wang, Christopher Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Mahima Pushkarna, Andrew Zaldivar, and Oddur Kjartansson. 2022. Data cards: Purposeful and transparent dataset documentation for responsible ai. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1776–1826.

Zenon W Pylyshyn and Ron W Storm. 1988. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial vision*, 3(3):179–197.

Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip Torr, and Song Bai. 2022. Occluded video instance segmentation: A benchmark. *IJCV*.

- Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. 2024. Streaming long video understanding with large language models. In *NeurIPS*.
- Qwen Team. 2024. Qwen2.5-vl: A versatile vision-language model. *arXiv preprint arXiv:2409.12174*.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *ICML*.
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.
- Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. 2025. Sam 2: Segment anything in images and videos. In *ICLR*.
- Arijit Ray, Filip Radenovic, Abhimanyu Dubey, Bryan A. Plummer, Ranjay Krishna, and Kate Saenko. 2023. Cola: How to adapt vision-language models to compose objects localized with attributes?
- Yuval Reif and Roy Schwartz. 2023. Fighting bias with bias: Promoting model robustness by amplifying dataset biases. *arXiv preprint arXiv:2305.18917*.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.
- Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. 2021. Token-learner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*.
- Mobashir Sadat, Zhengyu Zhou, Lukas Lange, Jun Araki, Arsalan Gundroo, Bingqing Wang, Rakesh R

- Menon, Md Rizwan Parvez, and Zhe Feng. 2023. Delucionqa: Detecting hallucinations in domain-specific question answering. *arXiv preprint arXiv:2312.05200*.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.
- Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350.
- Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.
- Atom Scott, Ikuma Uchida, Ning Ding, Rikuhei Umemoto, Rory Bunker, Ren Kobayashi, Takeshi Koyama, Masaki Onishi, Yoshinari Kameda, and Keisuke Fujii. 2024. Teamtrack: A dataset for multi-sport multi-object tracking in full-pitch videos. In *CVPR Workshop on Computer Vision in Sports*.
- Seonguk Seo, Joon-Young Lee, and Bohyung Han. 2020. Urvos: Unified referring video object segmentation network with a large-scale benchmark. In *ECCV*.
- Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.
- Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. 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.
- Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. 2022. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15638–15650.

- Harman Singh, Pengchuan Zhang, Qifan Wang, Mengjiao Wang, Wenhan Xiong, Jingfei Du, and Yu Chen. 2023. Coarse-to-fine contrastive learning in image-text-graph space for improved vision-language compositionality.
- Ben Snyder, Marius Moisescu, and Muhammad Bilal Zafar. 2024. On early detection of hallucinations in factual question answering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2721–2732.
- Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. 2024. Moviechat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18221–18232.
- Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.
- Elizabeth S Spelke. 1990. Principles of object perception. *Cognitive science*, 14(1):29–56.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.
- Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. 2017. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *International Workshop on Deep Learning in Medical Image Analysis*, pages 240–248. Springer.
- Peize Sun, Jinkun Cao, Yi Jiang, Zehuan Yuan, Song Bai, Kris Kitani, and Ping Luo. 2022. Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In *CVPR*.
- Gemini Team. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivi re, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

X.ai Team. 2024b. Grok-1.5 vision preview. <https://x.ai/blog/grok-1.5v>.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5238–5248.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *arXiv preprint arXiv:2406.16860*.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. 2025. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*.

Michael Tschannen, Manoj Kumar, Andreas Steiner, Xiaohua Zhai, Neil Houlsby, and Lucas Beyer. 2023. Image captioners are scalable vision learners too. *arXiv preprint arXiv:2306.07915*.

Priyesh Vakharia, Devavrat Joshi, Meenal Chavan, Dhananjay Sonawane, Bhriugu Garg, and Parsa Mazaheri. 2023. Don’t believe everything you read: Enhancing summarization interpretability through automatic identification of hallucinations in large language models. *arXiv preprint arXiv:2312.14346*.

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

Leon Amadeus Varga, Benjamin Kiefer, Martin Messmer, and Andreas Zell. 2022. Seadronessee: A maritime benchmark for detecting humans in open water. In *WACV*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

- Paul Voigtlaender, Soravit Changpinyo, Jordi Pont-Tuset, Radu Soricut, and Vittorio Ferrari. 2023. Connecting vision and language with video localized narratives. In *CVPR*.
- Johan Wagemans, James H Elder, Michael Kubovy, Stephen E Palmer, Mary A Peterson, Manish Singh, and Rüdiger Von der Heydt. 2012. A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172.
- Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, et al. 2025. 2 olmo 2 furious (colm’s version). In *COLM*.
- Haochen Wang, Cilin Yan, Shuai Wang, Xiaolong Jiang, XU Tang, Yao Hu, Weidi Xie, and Efstratios Gavves. 2023a. Towards open-vocabulary video instance segmentation. In *ICCV*.
- Jianyong Wang and Jiawei Han. 2004. Bide: Efficient mining of frequent closed sequences. In *Proceedings. 20th international conference on data engineering*, pages 79–90. IEEE.
- Jie Wang, Yuzhou Peng, Xiaodong Yang, Ting Wang, and Yanming Zhang. 2022. Sportstrack: An innovative method for tracking athletes in sports scenes. *arXiv preprint arXiv:2211.07173*.
- Junke Wang, Lingchen Meng, Zejia Weng, Bo He, Zuxuan Wu, and Yu-Gang Jiang. 2023b. To see is to believe: Prompting gpt-4v for better visual instruction tuning. *arXiv preprint arXiv:2311.07574*.
- Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. 2023c. Videomae v2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14549–14560.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024a. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.
- Qiuhe Wang, Yukai Shi, Jiarong Ou, Rui Chen, Ke Lin, Jiahao Wang, Boyuan Jiang, Haotian Yang, Mingwu Zheng, Xin Tao, Fei Yang, Pengfei Wan, and Di Zhang. 2025a. Koala-36m: A large-scale video dataset improving consistency between fine-grained conditions and video content. In *CVPR*.

- Shijian Wang, Linxin Song, Jieyu Zhang, Ryotaro Shimizu, Ao Luo, Li Yao, Cunjian Chen, Julian McAuley, and Hanqian Wu. 2024b. Template matters: Understanding the role of instruction templates in multimodal language model evaluation and training. *arXiv preprint arXiv:2412.08307*.
- Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. 2025b. Internv13.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*.
- Xiao Wang, Liye Jin, Xufeng Lou, Shiao Wang, Lan Chen, Bowei Jiang, and Zhipeng Zhang. 2025c. Reasoningtrack: Chain-of-thought reasoning for long-term vision-language tracking. *arXiv preprint arXiv:2508.05221*.
- Xiao Wang, Xiujun Shu, Zhipeng Zhang, Bo Jiang, Yaowei Wang, Yonghong Tian, and Feng Wu. 2021. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In *CVPR*.
- Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. 2019. VateX: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4581–4591.
- Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. 2023d. Internvid: A large-scale video-text dataset for multimodal understanding and generation. In *ICLR*.
- Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, et al. 2024c. Internvideo2: Scaling foundation models for multimodal video understanding. In *European Conference on Computer Vision*, pages 396–416. Springer.
- Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. 2019. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 284–293.
- Haoning Wu, Zicheng Zhang, Erli Zhang, Chaofeng Chen, Liang Liao, Annan Wang, Chunyi Li, Wenxiu Sun,

- Qiong Yan, Guangtao Zhai, and Weisi Lin. 2024. Q-bench: A benchmark for general-purpose foundation models on low-level vision. In *ICLR*.
- Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. 2024. Slowfast-llava: A strong training-free baseline for video large language models. *arXiv preprint arXiv:2407.15841*.
- Le Xue, Manli Shu, Anas Awadalla, Jun Wang, An Yan, Senthil Purushwalkam, Honglu Zhou, Viraj Prabhu, Yutong Dai, Michael S Ryoo, et al. 2024. xGen-MM (BLIP-3): A family of open large multimodal models. *arXiv preprint arXiv:2408.08872*.
- Cilin Yan, Haochen Wang, Shilin Yan, Xiaolong Jiang, Yao Hu, Guoliang Kang, Weidi Xie, and Efstratios Gavves. 2024. Visa: Reasoning video object segmentation via large language models. In *ECCV*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. 2023. Track anything: Segment anything meets videos.
- Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. 2024. Depth anything v2. *arXiv preprint arXiv:2406.09414*.
- Yue Yang, Ajay Patel, Matt Deitke, Tanmay Gupta, Luca Weihs, Andrew Head, Mark Yatskar, Chris Callison-Burch, Ranjay Krishna, Aniruddha Kembhavi, et al. 2025b. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. In *ACL*.

- Mingqiao Ye, Seoung Wug Oh, Lei Ke, and Joon-Young Lee. 2025. Entitysam: Segment everything in video. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24234–24243.
- Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, Jiayi Lei, Quanfeng Lu, Runjian Chen, Peng Xu, Renrui Zhang, Haozhe Zhang, Peng Gao, Yali Wang, Yu Qiao, Ping Luo, Kaipeng Zhang, and Wenqi Shao. 2024. Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi.
- Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*.
- Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. 2022. Coca: Contrastive captioners are image-text foundation models.
- Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. In *Forty-first International Conference on Machine Learning*.
- Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. 2019. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9127–9134.
- Liangzhe Yuan, Nitesh Bharadwaj Gundavarapu, Long Zhao, Hao Zhou, Yin Cui, Lu Jiang, Xuan Yang, Menglin Jia, Tobias Weyand, Luke Friedman, et al. 2023. Videoglue: Video general understanding evaluation of foundation models. *arXiv preprint arXiv:2307.03166*.
- Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. 2024. Robopoint: A vision-language model for spatial affordance prediction for robotics. In *CoRL*.
- 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 Wenhua Chen. 2024. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*.

- Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. 2023. When and why vision-language models behave like bags-of-words, and what to do about it? In *International Conference on Learning Representations*.
- Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.
- Rowan Zellers, Jiasen Lu, Ximing Lu, Youngjae Yu, Yanpeng Zhao, Mohammadreza Salehi, Aditya Kusupati, Jack Hessel, Ali Farhadi, and Yejin Choi. 2022. Merlot reserve: Multimodal neural script knowledge through vision and language and sound. In *CVPR*.
- Chunhui Zhang, Guanjie Huang, Li Liu, Shan Huang, Yinan Yang, Xiang Wan, Shiming Ge, and Dacheng Tao. 2023a. Webuav-3m: A benchmark for unveiling the power of million-scale deep uav tracking. *TPAMI*.
- Chunhui Zhang, Li Liu, Guanjie Huang, Hao Wen, Xi Zhou, and Yanfeng Wang. 2024a. Webuot-1m: Advancing deep underwater object tracking with a million-scale benchmark. In *NeurIPS*.
- Hang Zhang, Xin Li, and Lidong Bing. 2023b. Video-LLaMA: An instruction-tuned audio-visual language model for video understanding. In *EMNLP Demo Track*.
- Hao Zhang et al. 2023c. Lvbench: A benchmark for long-form video understanding. *arXiv preprint arXiv:2312.04817*.
- Hao Zhang et al. 2024b. Lvbench: An extreme long video understanding benchmark. *arXiv preprint arXiv:2406.08035*.
- Jieyu Zhang, Weikai Huang, Zixian Ma, Oscar Michel, Dong He, Tanmay Gupta, Wei-Chiu Ma, Ali Farhadi, Aniruddha Kembhavi, and Ranjay Krishna. 2024c. Task me anything. In *Thirty-Eighth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Jieyu Zhang, Le Xue, Linxin Song, Jun Wang, Weikai Huang, Manli Shu, An Yan, Zixian Ma, Juan Carlos

- Niebles, Silvio Savarese, et al. 2024d. Provision: Programmatically scaling vision-centric instruction data for multimodal language models. *arXiv preprint arXiv:2412.07012*.
- Libo Zhang, Junyuan Gao, Zhen Xiao, and Heng Fan. 2023d. Animaltrack: A benchmark for multi-animal tracking in the wild. *IJCV*.
- Pengchuan Zhang, Xiujuan Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5579–5588.
- Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. 2024e. Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*.
- Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. 2025. Llava-video: Video instruction tuning with synthetic data. *TMLR*.
- Long Zhao, Nitesh B Gundavarapu, Liangzhe Yuan, Hao Zhou, Shen Yan, Jennifer J Sun, Luke Friedman, Rui Qian, Tobias Weyand, Yue Zhao, et al. 2024. Videoprism: A foundational visual encoder for video understanding. *arXiv preprint arXiv:2402.13217*.
- Tiancheng Zhao, Tianqi Zhang, Mingwei Zhu, Haozhan Shen, Kyusong Lee, Xiaopeng Lu, and Jianwei Yin. 2022. VI-checklist: Evaluating pre-trained vision-language models with objects, attributes and relations. *arXiv preprint arXiv:2207.00221*.
- Zhiyuan Zhao, Bin Wang, Linke Ouyang, Xiaoyi Dong, Jiaqi Wang, and Conghui He. 2023. Beyond hallucinations: Enhancing lvlms through hallucination-aware direct preference optimization. *arXiv preprint arXiv:2311.16839*.
- Chenhao Zheng, Jieyu Zhang, Mohammadreza Salehi, Ziqi Gao, Vishnu Iyengar, Norimasa Kobori, Quan Kong, and Ranjay Krishna. 2025. One trajectory, one token: Grounded video tokenization via panoptic sub-object trajectory. *arXiv:2505.23617*.
- Chenhao Zheng, Jieyu Zhang, Jianing Zhang, Weikai Huang, Ashutosh Kumar, Quan Kong, Oncel Tuzel, Chun-Liang Li, and Ranjay Krishna. 2026. Trajtok: Learning trajectory tokens enables better video

- understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Guangze Zheng, Shijie Lin, Haobo Zuo, Changhong Fu, and Jia Pan. 2024. Nettrack: Tracking highly dynamic objects with a net. In *CVPR*.
- Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. 2024. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*.
- Junjie Zhou et al. 2025. Mlvu: Benchmarking multi-task long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- William Y Zhu, Keren Ye, Junjie Ke, Jiahui Yu, Leonidas Guibas, Peyman Milanfar, and Feng Yang. 2024. Artvlm: Attribute recognition through vision-based prefix language modeling. *arXiv preprint arXiv:2408.04102*.
- Yabin Zhu, Chenglong Li, Yao Liu, Xiao Wang, Jin Tang, Bin Luo, and Zhixiang Huang. 2023. Tiny object tracking: A large-scale dataset and a baseline. *TNNLS*.

Chapter A

Appendix: SugarCrepe

A.1 Implementation details

A.1.1 Hardware information

All experiments are run on a machine with an Intel(R) Xeon(R) CPU E5-2678 v3 with a 512G memory and two 48G NVIDIA RTX A6000 GPUs.

A.1.2 Dataset sources

We obtain all existing datasets from their original sources released by the authors. We refer readers to these sources for the dataset licenses. To the best of our knowledge, the data we use does not contain personally identifiable information or offensive content.

- CREPE [Ma et al., 2022]: We obtain CREPE dataset from its official repository ¹.
- ARO [Yuksekgonul et al., 2023]: We obtain ARO dataset from its official repository ².
- VL-CheckList [Zhao et al., 2022]: We obtain VL-CheckList dataset from its official repository ³.
- COCO [Lin et al., 2014]: We obtain COCO from its official project website ⁴.

¹<https://github.com/RAIVNLab/CREPE>

²<https://github.com/mertyg/vision-language-models-are-bows>

³<https://github.com/om-ai-lab/VL-CheckList>

⁴<https://cocodataset.org/>

A.1.3 Software configuration

Models. We detail the sources of the pretrained models we use in the paper, and the hyper-parameters used in training our own models.

- Vera model [Liu et al., 2023a]: We obtain pretrained Vera model released by its author ⁵.
- Grammar model [Morris et al., 2020]: We obtain the Grammar model released by the authors ⁶.
- All pretrained CLIP models: We obtain all pretrained CLIP models’ weights from OpenCLIP ⁷.
- NEGCLIP models: We obtain weights for pretrained NEGCLIP released by the authors ⁸. For training from scratch and finetuning, we train RN50 and ViT-B/32 based on OpenCLIP codebase and set hyper-parameters as the following: number of warmup steps is 1000, batch size is 256, learning rate is 1e-4, weight decay is 0.1, number of epochs is 30. We augment the original CLIP loss with hard negative captions following NEGCLIP [Yuksekgonul et al., 2023].

Evaluations. We base our evaluation framework on OpenCLIP [Ilharco et al., 2021]. We follow all default hyper-parameters used for evaluating models.

A.2 Vision-language compositionality benchmarks

We provide an overview of existing vision-language compositionality benchmarks below, with Table A.1 summarizing the dataset comparisons.

A.2.1 Image-to-text formulation

A majority of current benchmarks formulate the evaluation task as image-to-text retrieval problem. These benchmarks generate hard negative texts procedurally through rule-based templates, where each benchmark considers different types of hard negatives.

⁵<https://huggingface.co/liujch1998/vera>

⁶<https://huggingface.co/textattack/distilbert-base-uncased-CoLA>

⁷https://github.com/mlfoundations/open_clip

⁸<https://github.com/mertyg/vision-language-models-are-bows>

VL-Checklist [Zhao et al., 2022]. VL-CheckList aims at evaluating vision-language models’ understanding of different objects, attributes, and relationships. It contains REPLACE hard negatives generated by replacing atomic parts of the positive texts with other foils. VL-CheckList further breaks the hard negatives down into more granular categories based on the type of the replaced atomic part, i.e. , object, attribute, or relationship.

ARO [Yuksekgonul et al., 2023]. ARO focuses on models’ understanding of different relationships, attributes, and order information. It considers SWAP and SHUFFLE hard negatives. SWAP hard negatives are generated by swapping two words in the positive texts; on the other hand, SHUFFLE hard negatives are generated by shuffling words in the positive texts. ARO further divides SWAP hard negatives into attribute or relationship type.

CREPE [Ma et al., 2022]. CREPE is a large-scale evaluation benchmark that includes three types of hard negatives: REPLACE, SWAP and NEGATE. REPLACE and SWAP hard negatives are generated as in VL-CheckList and ARO. In addition, NEGATE hard negatives are generated by adding negation keywords (i.e. , *not* or *no*) to the original positive texts. The hard negatives are not further divided into fine-grained types (object, attribute, or relations).

A.2.2 Text-to-image formulation

Complementary to image-to-text formulation, compositionality can as well be evaluated by probing a model to select an image that best matches a given text description, against other hard negative images as distractors. Unlike hard negative texts, hard negative images are more difficult to obtain and thus current text-to-image compositionality benchmarks are smaller at scale.

Winoground [Thrush et al., 2022]. Winoground is a small dataset manually curated by human annotators. Each example in the dataset contains two images and two matching captions, where both captions contain identical words that appear in different orders. Note that Winoground can be used for either image-to-text or text-to-image retrieval. While the original intention for Winoground is to evaluate vision-language compositionality, recent work [Diwan et al., 2022] has pointed out that solving the tasks in Winoground requires not just compositional vision-language understanding, but additionally a suite of other abilities such as commonsense reasoning, or distinguishing visually difficult images.

Benchmark	Task Formulation	Scale	Hard Negative Text Type				
			SHUFFLE	REPLACE	SWAP	NEGATE	ADD
VL-CheckList [Zhao et al., 2022]	Image-to-Text	> 1000		✓			
ARO [Yuksekgonul et al., 2023]	Image-to-Text	> 1000	✓		✓		
CREPE [Ma et al., 2022]	Image-to-Text	> 1000		✓	✓	✓	
Winoground [Thrush et al., 2022]	Image-to-Text / Text-to-Image	400			✓		
Cola [Ray et al., 2023]	Text-to-Image	210			N/A		
SUGARCREPE	Image-to-Text	> 1000		✓	✓		✓

Table A.1: Summary on vision-language compositionality benchmarks. SUGARCREPE considers image-to-text formulation to enable larger scale evaluation set. In addition, SUGARCREPE considers a wide range of hard negative types. SHUFFLE and NEGATE are omitted as they introduce inevitable biases discussed in Sec. 2.3.1.

Cola [Ray et al., 2023]. Cola tests a vision-language model’s ability to select an image that correctly matches a given caption, against another distractor image with the same objects and attributes but in the wrong composition. The image pairs are mined from existing datasets. As a result, the final evaluation set is relatively small in size (210 examples in total).

We deem text-to-image evaluation as important as image-to-text evaluation. Future work can explore approaches to generate or mine compositional hard negative images at scale, as preliminarily explored in [Ray et al., 2023; Yuksekgonul et al., 2023].

A.3 SUGARCREPE

A.3.1 Taxonomy

Figure A.1 shows the taxonomy of SUGARCREPE. We first categorize the hard negatives based on their forms: REPLACE, SWAP, and ADD. We then further divide each type of hard negatives into finer-grained sub-categories based on the type (object, attribute, or relation) of the atomic concept altered. SUGARCREPE covers a total of 7 fine-grained hard negative types.

A.3.2 Hard negative generation procedure and templates

To generate hard negatives in SUGARCREPE, we come up with three different prompt templates for the three hard negative types considered: REPLACE, SWAP, and ADD. Each template consists of task instruction for

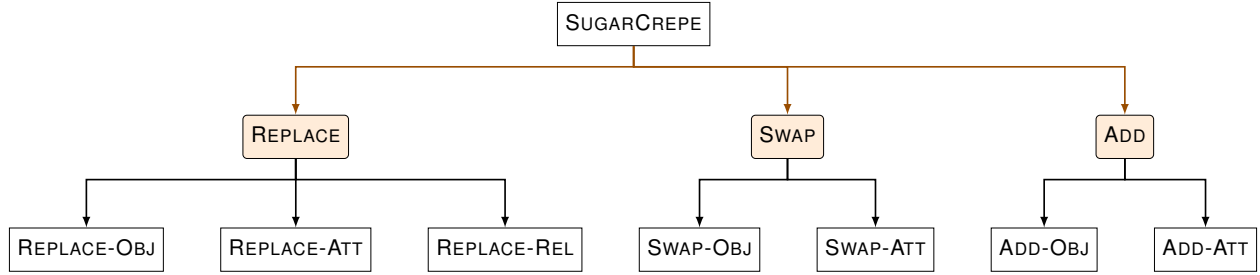


Figure A.1: Taxonomy of hard negatives considered in SUGARCREPE.

generating the corresponding type of hard negatives and several (7 or more) few-shot demonstrations. We describe the general generation procedure and example prompt templates below and refer readers to our dataset repository for the full prompts used ⁹.

Generating REPLACE hard negatives. To best leverage ChatGPT’s capabilities, we devise a three-step workflow to generate REPLACE hard negatives: (1) We prompt ChatGPT in locating the desired atomic concepts (e.g. , objects) in the sentence; (2) We prompt ChatGPT to generate a new concept to replace a randomly selected old concept; (3) We let ChatGPT compose a new sentence by replacing the old concept with the new one. For steps (1) and (3), we prompt ChatGPT with a temperature of 0.0 to get stable outputs. For step (2), however, we diversify the outputs by prompting ChatGPT with a higher temperature of 1.5. With this design, we are able to generate diverse REPLACE hard negatives. Figure A.2 shows the example templates and outputs for REPLACE hard negatives.

Generating SWAP hard negatives. To generate SWAP hard negatives, which do not require any new concepts, we simply prompt ChatGPT once with a temperature of 0.0. Unlike REPLACE, SWAP hard negatives are only possible when there are at least two atomic concepts of the same category, i.e. , either object or attribute. Thus, our prompt first queries ChatGPT whether it is possible to swap two atomic concepts in the input sentence to generate a new description. Only if the answer is yes, will ChatGPT then proceed to identify two swappable concepts and compose the corresponding new sentence by swapping the two concepts. Figure A.3 shows the example templates and outputs for SWAP hard negatives.

Generating ADD hard negatives. Similar to the REPLACE, we also employ a three-step prompting procedure to generate ADD hard negatives. The only difference in the procedure is that we prompt ChatGPT to add the generated new concept to the original caption, instead of using it to replace an old concept.

⁹<https://github.com/RAIVNLab/sugar-crepe>

Figure A.4 shows the example templates and outputs for ADD hard negatives.

A.3.3 Adversarial refinement

We detail the adversarial refinement procedure below. Given a text model M , we denote its output score for the positive and negative caption of i -th image as $M(p_i)$ and $M(n_i)$. If $M(p_i) > M(n_i)$, then the model could identify the correct caption for the i -th image without referring to it. For a test set to be unattackable given the text model M , the expectation of M 's identifying the correct caption should be as close to random guess as possible; in particular, we hope that $E_i[M(p_i) > M(n_i)] = 0.5$. To achieve this for both the grammar model M_1 and plausibility model M_2 , we first calculate the score difference $g_i^{(1)} = M_1(p_i) - M_1(n_i)$ and $g_i^{(2)} = M_2(p_i) - M_2(n_i)$, where the range of both $g^{(1)}$ and $g^{(2)}$ is $[-1, 1]$. Then we split the 2D space of the joint range of $g^{(1)}$ and $g^{(2)}$ into 100×100 equal grids, and for each pair of symmetric grids, e.g. , $\{(g^{(1)}, g^{(2)}) | g^{(1)} \in (0.02, 0.04], g^{(2)} \in (-0.04, 0.06]\}$ and $\{(g^{(1)}, g^{(2)}) | g^{(1)} \in (-0.02, -0.04], g^{(2)} \in (0.04, -0.06]\}$, we preserve the same number of data for both grids, therefore we ensure that for the resultant set, $E_i[M_1(p_i) > M_1(n_i)] = 0.5$ and $E_i[M_2(p_i) > M_2(n_i)] = 0.5$.

A.3.4 Dataset construction cost

We provide a high-level overview to the cost used to build SUGARCREPE by utilizing OpenAI's ChatGPT API for generating hard negatives. In building SUGARCREPE, we use approximately 40 API calls to generate hard negatives for each COCO test caption, including all different fine-grained types of hard negatives. This amounts to a total of $25,000 \times 40 = 1,000,000$ API calls to ChatGPT. With each API call costing around \$0.0005, it took roughly \$500 to build SUGARCREPE.

A.3.5 Dataset information

We host SUGARCREPE on Github ¹⁰. The data card [Pushkarna et al., 2022] for SUGARCREPE, containing detailed dataset documentation, is available at the dataset repository ¹¹. We provide a summary below.

Dataset documentation. SUGARCREPE is a benchmark for faithful vision-language compositionality evaluation. Given an image, a model is required to select the positive text that correctly describes the image,

¹⁰<https://github.com/RAIVNLab/sugar-crepe>

¹¹https://github.com/RAIVNLab/sugar-crepe/blob/main/data_card.pdf

against another hard negative text distractor that differs from the positive text only by small compositional changes. Each example consists of three fields:

- `filename`: The id to an image
- `caption`: Positive text correctly describing the image
- `negative_caption`: Hard negative text incorrectly describing the image

Maintenance plan. We are committed to maintain the dataset to address any technical issues. We actively monitor issues in the repository.

Licensing. We license our work using MIT License ¹². All the source data we use is publicly released by prior work [Lin et al., 2014].

A.4 Detailed evaluation results

A.4.1 Full evaluation results on existing benchmarks

We provide the full evaluation results over 17 pretrained CLIP models as well as 2 text-only models, Vera [Liu et al., 2023a] and the Grammar model [Morris et al., 2020], on existing compositionality benchmarks in Table A.2. We see that the text-only models, arguably without any vision-language compositionality, outperform most of the pretrained CLIP models, achieving state-of-the-art performances on many benchmark tasks. This implies that current benchmarks fail to faithfully reflect a model’s vision-language compositionality.

A.4.2 SUGARCREPE human evaluation

To compare the quality of the hard negatives generated in SUGARCREPE to those in current benchmarks (i.e., ARO+CREPE), we randomly sample 100 examples for each of the hard negative types: REPLACE, SWAP, and NEGATE / ADD. Each example is organized to consist of (1) the original positive text, (2) its hard negative in ARO+CREPE, and (3) its hard negative in SUGARCREPE. For each example, a human user rates whether the hard negative in ARO+CREPE or that in SUGARCREPE is better (or tie) in terms of commonsense and grammatical correctness, respectively. Note that we compare NEGATE in ARO+CREPE

¹²<https://github.com/RAIVNLab/sugar-crepe/blob/main/LICENSE>

Source	Model	CREPE			ARO				VL-Checklist		
		Atomic	Swap	Negate	VG-Relation	VG-Attribution	COCO-Order	Flickr30K-Order	Object	Attribute	Relation
Text-only model	Vera [Liu et al., 2023a]	43.70	70.80	66.15	61.71	82.59	59.81	63.52	82.48	73.99	85.72
	Grammar [Morris et al., 2020]	18.15	50.88	9.77	59.55	58.38	74.33	76.26	57.95	52.35	68.50
OpenAI [Radford et al., 2021]	RN50	26.47	28.32	31.25	53.87	63.37	44.89	52.46	86.85	68.30	75.95
	RN101	27.63	32.74	12.50	52.43	62.93	29.86	39.34	86.44	67.93	71.75
	RN50x4	26.24	28.32	9.51	51.59	62.27	29.39	34.56	87.23	68.74	73.81
	ViT-B-32	22.31	26.55	28.78	51.12	61.33	37.14	47.18	87.00	68.80	77.04
	RN50x16	26.36	29.65	9.38	52.13	62.71	29.95	34.26	86.95	69.34	76.83
	RN50x64	26.82	30.09	23.57	51.00	62.56	40.54	46.74	87.71	68.61	74.97
	ViT-L-14	26.36	25.66	24.74	53.34	61.50	36.11	45.08	87.86	68.27	75.89
LAION [Schuhmann et al., 2022]	ViT-H-14	23.70	25.22	16.54	50.33	62.93	25.79	30.96	85.39	68.46	71.13
	ViT-g-14	23.70	24.78	20.70	51.60	61.20	25.59	30.10	86.07	69.43	71.03
	ViT-bigG-14	23.58	24.78	17.97	51.61	61.89	25.24	30.22	84.66	67.80	66.48
	roberta-ViT-B-32	22.66	21.24	20.31	47.46	62.00	24.77	30.76	85.71	68.82	65.90
	xlm-roberta-base-ViT-B-32	21.16	20.80	12.76	47.93	59.73	23.85	30.32	86.06	70.41	63.01
	xlm-roberta-large-ViT-H-14	24.16	23.89	20.05	46.14	57.84	26.05	31.00	87.89	70.25	63.89
DataComp [Gadre et al., 2024]	small: ViT-B-32	13.64	27.88	14.84	50.83	50.17	13.35	14.02	68.72	58.80	57.00
	medium: ViT-B-32	16.42	20.35	11.33	50.45	54.04	16.44	16.26	78.43	63.53	62.94
	large: ViT-B-16	18.15	17.26	17.06	48.82	53.21	21.49	26.44	84.73	65.72	64.81
	x-large: ViT-L-14	21.62	22.57	16.28	48.54	60.03	23.19	29.52	86.66	67.01	67.93

Table A.2: Blind models (i.e., Vera and Grammar model) outperform all 17 existing pretrained CLIP models on nearly all existing benchmark tasks. This implies that current benchmarks fail to faithfully measure a model’s vision-language compositionality.

to ADD in SUGARCREPE, as both hard negatives are intended to probe a model’s understanding of the *existence or not* of an atomic concept. Table A.3 shows that hard negatives in SUGARCREPE are much more sensical and fluent than that in ARO+CREPE across all three different types. For instance, SUGARCREPE has 68% more sensical and 46% more fluent hard negatives than ARO+CREPE on SWAP.

Hard-negative Type	Evaluation	Human counts of better examples		
		ARO+CREPE	SUGARCREPE	Tie
REPLACE	Commonsense	11	29	60
	Grammar	4	33	63
SWAP	Commonsense	4	68	28
	Grammar	4	46	50
NEGATE / ADD	Commonsense	1	26	73
	Grammar	1	35	64

Table A.3: Human evaluation results on the comparisons between hard negatives in ARO+CREPE and SUGARCREPE. We report the counts (out of 100 sampled examples) that the human user considers better or tie, w.r.t. both commonsense and grammatical correctness.

A.4.3 Additional NEGCLIP results

In this section, we conduct preliminary experiments to answer whether models’ performances on SUGARCREPE would increase hugely if the models are trained with hard negatives generated through the same procedure as how we create SUGARCREPE. Since generating hard negatives for training with ChatGPT

would incur substantial cost, we create hard negatives using a proxy method. In particular, we start with template-generated hard negatives on the COCO training set and apply our adversarial refinement technique to remove the biases. We use this adversarially refined dataset for NEGCLIP training. We show the results in Table A.4. While we observe that the method improves over vanilla CLIP training without hard negatives, it performs similarly to NegCLIP and does not saturate the performance on SugarCrepe. This suggests that while the adversarial refinement mechanism prevents SugarCrepe from being attacked as an evaluation benchmark, leveraging the approach alone for training does not saturate the performance on SugarCrepe. Future work may characterize how LLMs could be used to generate better hard negatives for training to genuinely improve vision-language models’ compositionality.

Model	Hard negative	SUGARCREPE		
		REPLACE	SWAP	ADD
CLIP without hard negatives	N/A	69.54	60.33	67.63
NEGCLIP with template hard negatives	REPLACE	74.32	62.65	72.92
NEGCLIP with adversarial refined hard negatives	REPLACE	73.37	61.40	72.84
NEGCLIP with template hard negatives	SWAP	73.31	68.35	71.93
NEGCLIP with adversarial refined hard negatives	SWAP	72.07	65.13	69.68
NEGCLIP with template hard negatives	NEGATE	72.74	60.89	70.47
NEGCLIP with adversarial refined hard negatives	NEGATE	72.70	60.75	68.70

Table A.4: Model performances on SUGARCREPE when trained with hard negatives generated through similar procedure as how SUGARCREPE is created.

Given an input sentence describing a scene, your task is to:

1. Locate the noun words in the sentence.
2. Randomly pick one noun word.
3. Replace the selected noun word with a new noun word to make a new sentence.

The new sentence must meet the following three requirements:

1. The new sentence must be describing a scene that is as different as possible from the original scene.
2. The new sentence must be fluent and grammatically correct.
3. The new sentence must make logical sense.

Here are some examples:

Original sentence: A man is in a kitchen making pizzas.
Nouns: ["man", "kitchen", "pizzas"]
Selected noun: man
New noun: woman
New sentence: A woman is in a kitchen making pizzas.

Original sentence: a woman seated on wall and birds besides her
Nouns: ['woman', 'wall', 'birds']
Selected noun: wall
New noun: bench
New sentence: A woman seated on a bench and birds besides her.

(a) REPLACE-OBJ.

Given an input sentence describing a scene, your task is to:

1. Locate the adjective words describing objects in the sentence. If there are no adjective words, return an empty list.
2. Randomly pick one adjective word.
3. Replace the selected adjective word with a new adjective word to make a new sentence.

The new sentence must meet the following three requirements:

1. The new sentence must be describing a scene that is as different as possible from the original scene.
2. The new sentence must be fluent and grammatically correct.
3. The new sentence must make logical sense.

Here are some examples:

Original sentence: a blue bike parked on a side walk.
Adjectives: ["blue"]
Selected adjective: blue
New adjective: red
New sentence: a red bike parked on a side walk.

Original sentence: The kitchen is clean and ready for us to see.
Adjectives: ["clean", "ready"]
Selected adjective: clean
New adjective: dirty
New sentence: The kitchen is dirty and ready for us to see.

(b) REPLACE-ATT.

Given an input sentence describing a scene, your task is to:

1. Find any action or spatial relationships between two objects in the sentence. If there are no such relationships, return an empty list.
2. Randomly pick one relationship.
3. Replace the selected relationship with a new relationship to make a new sentence.

The new sentence must meet the following three requirements:

1. The new sentence must be describing a scene that is as different as possible from the original scene.
2. The new sentence must be fluent and grammatically correct.
3. The new sentence must make logical sense.

Here are some examples:

Original sentence: The dining table near the kitchen has a bowl of fruit on it.
Relationships: ["near", "on"]
Selected relationship: near
New relationship: far from
New sentence: The dining table far from the kitchen has a bowl of fruit on it.

Original sentence: A couple of buckets in a white room.
Relationships: ['in']
Selected relationship: in
New relationship: outside
New sentence: A couple of buckets outside a white room.

(c) REPLACE-REL.

Figure A.2: Example prompt templates (black) and outputs (green) from ChatGPT for REPLACE hard negatives.

Given an input sentence describing a scene, your task is to first locate two swappable noun phrases in the sentence, and then swap them to make a new sentence. The new sentence must meet the following three requirements: 1. The new sentence must be describing a different scene from the input sentence. 2. The new sentence must be fluent and grammatically correct. 3. The new sentence must make logical sense. To complete the task, you should: 1. Answer the question of whether generating such a new sentence is possible using Yes or No. 2. Output the swappable noun phrases. 3. Swap them to make a new sentence. Here are some examples: Input: A cat resting on a laptop next to a person. Is it possible to swap noun phrases in the input sentence to generate a new sentence that is different from the input sentence and makes logical sense? Yes Swappable noun phrases: laptop, person Output: A cat resting on a person next to a laptop. Input: A plate of donuts with a person in the background. Is it possible to swap noun phrases in the input sentence to generate a new sentence that is different from the input sentence and makes logical sense? Yes Swappable noun phrases: a plate of donuts, a person Output: A person with a plate of donuts in the background.	Given an input sentence describing a scene, your task is to first locate two swappable adjectives in the sentence describing different objects, and then swap them to make a new sentence. The new sentence must meet the following three requirements: 1. The new sentence must be describing a different scene from the input sentence. 2. The new sentence must be fluent and grammatically correct. 3. The new sentence must make logical sense. To complete the task, you should: 1. Answer the question of whether generating such a new sentence is possible using Yes or No. 2. Output the swappable adjectives. 3. Swap them to make a new sentence. Here are some examples: Input: A girl in a pink shirt holding a blue umbrella. Is it possible to swap attributes in the input sentence to generate a new sentence that is different from the input sentence and makes logical sense? Yes Swappable attributes: pink, blue Output: A girl in a blue shirt holding a pink umbrella. Input: A girl with a green shirt brushing her teeth with a blue toothbrush. Is it possible to swap attributes in the input sentence to generate a new sentence that is different from the input sentence and makes logical sense? Yes Swappable attributes: green, blue Output: A girl with a blue shirt brushing her teeth with a green toothbrush.
--	---

(a) SWAP-OBJ.

(b) SWAP-ATT.

Figure A.3: Example prompt templates (black) and outputs (green) from ChatGPT for SWAP hard negatives.

Given an input sentence describing a scene, your task is: 1. Find the objects in the sentence. 2. Randomly pick one object. 3. Generate a new object that's not in the sentence. 4. Add the new object next to the selected object to make a new sentence. The new sentence must meet the following three requirements: 1. The new sentence must describe a clearly new and different scene. 2. The new sentence must be fluent and grammatically correct. 3. The new sentence must make logical sense. Here are some examples: Original sentence: An elephant standing under the shade of a tree. Objects: ["elephant", "shade of a tree"] Selected object: elephant New object: squirrel New sentence: An elephant and a squirrel standing under the shade of a tree. Original sentence: A bench at the beach next to the sea Objects: ['bench', 'beach', 'sea'] Selected object: bench New object: umbrella New sentence: An umbrella and a bench at the beach next to the sea.	Given an input sentence describing a scene, your task is: 1. Find the objects in the sentence. 2. Randomly pick one object. 3. Generate a new plausible but uncommon attribute for this object that's not in the sentence. 4. Add the new attribute next to the selected object to make a new sentence. The new sentence must meet the following three requirements: 1. The new sentence must describe a clearly new and different scene. 2. The new sentence must be fluent and grammatically correct. 3. The new sentence must make logical sense. Here are some examples: Original sentence: A large white airplane and a person on a lot. Objects: ["airplane", "person"] Selected object: airplane New attribute: blue New sentence: A large white and blue airplane and a person on a lot. Original sentence: three people riding horses on a beach Objects: ['three people', 'horses', 'beach'] Selected object: three people New attribute: elderly New sentence: Three elderly people riding horses on a beach.
--	--

(a) ADD-OBJ.

(b) ADD-ATT.

Figure A.4: Example prompt templates (black) and outputs (green) from ChatGPT for ADD hard negatives.

Chapter B

Appendix: Task-Me-Anything

B.1 Details of Task Generation

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

B.1.1 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 [Deitke et al., 2023, 2024] and their associated annotations or the real images and scene graphs from GQA [Hudson and Manning, 2019; Krishna et al., 2017b].

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.1.2 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 [Northcutt et al., 2021], 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.

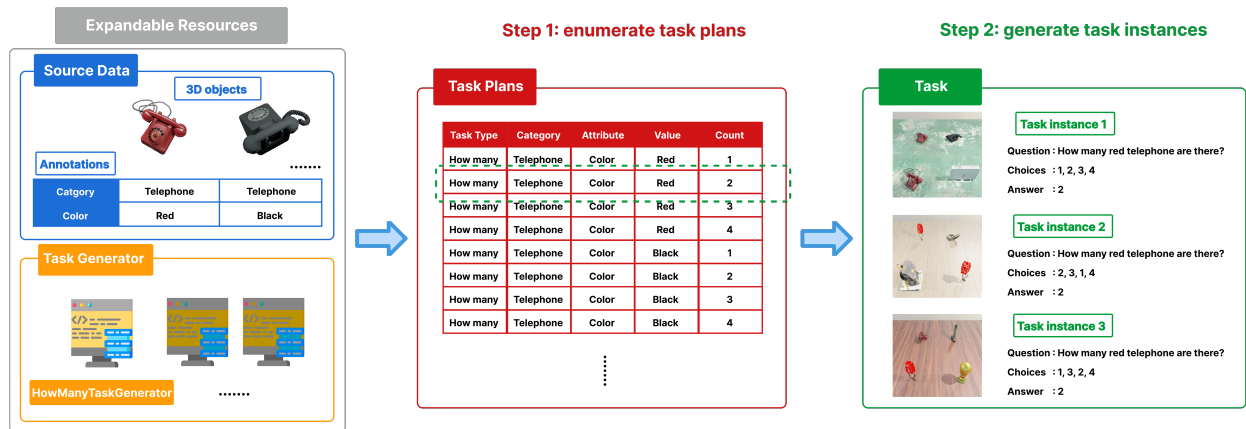


Figure B.1: An illustration of core concepts and the task generation process.

B.2 Details of Fine-grained User Query and Query Approximation Algorithms

With TASK-ME-ANYTHING, 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.

B.2.1 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.

B.2.2 Query execution

We provide an example of the conceptual query execution process in Figure B.2, 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.

Query Execution

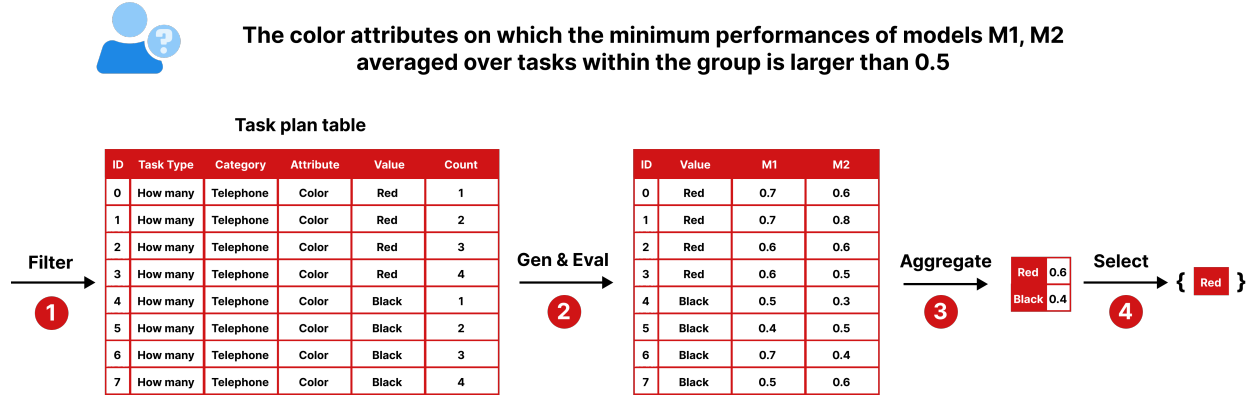


Figure B.2: An illustration of the query execution process.

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 [Han et al., 2022; Wang and Han, 2004; Han et al., 2001] 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.

B.2.3 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 regressor¹ because of its stable performance in our preliminary experiments, while any regression model is applicable.

¹https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html

B.3 Details of TASK-ME-ANYTHING 1.0

In this section, we introduce the task generators implemented in the first version of TASK-ME-ANYTHING. Inspired by the model cards for model reporting [Mitchell et al., 2019], 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 B.3.

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"
• Source Data. The source data and annotations it requires	– Answer Type. The answer type e.g. , integer number or object category
	– The model capability to evaluate. e.g. , counting
• 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 B.3: Summary of task generator card sections and suggested prompts for each.

B.3.1 Source data

3D objects with annotations. We start by selecting objects from Objaverse-LVIS, the subset of Objaverse 1.0 [Deitke et al., 2023] that has been annotated with LVIS [Gupta et al., 2019] 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 [Community, 2018], 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 [Krishna et al., 2017b] as part of our source data. In particular, we collect real images with scene graphs from the GQA dataset [Krishna et al., 2017b; Hudson and Manning, 2019] and real videos with scene graphs from the AGQA dataset [Ji et al., 2020; Sigurdsson et al., 2016].

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.

B.3.2 Task generators for different scenarios

2D sticker image

The first scenario of TASK-ME-ANYTHING 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 B.1.

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

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			

Table B.1: *2D sticker image*

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 [Johnson et al., 2017]. 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 B.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 B.3.

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 B.2: *3D tabletop scene* with images

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 B.4&B.5.

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 B.3: *3D tabletop scene* with videos

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 B.4: Real images 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			

Table B.5: Real videos with *Scene Graph*.

B.3.3 TASK-ME-ANYTHING-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 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);

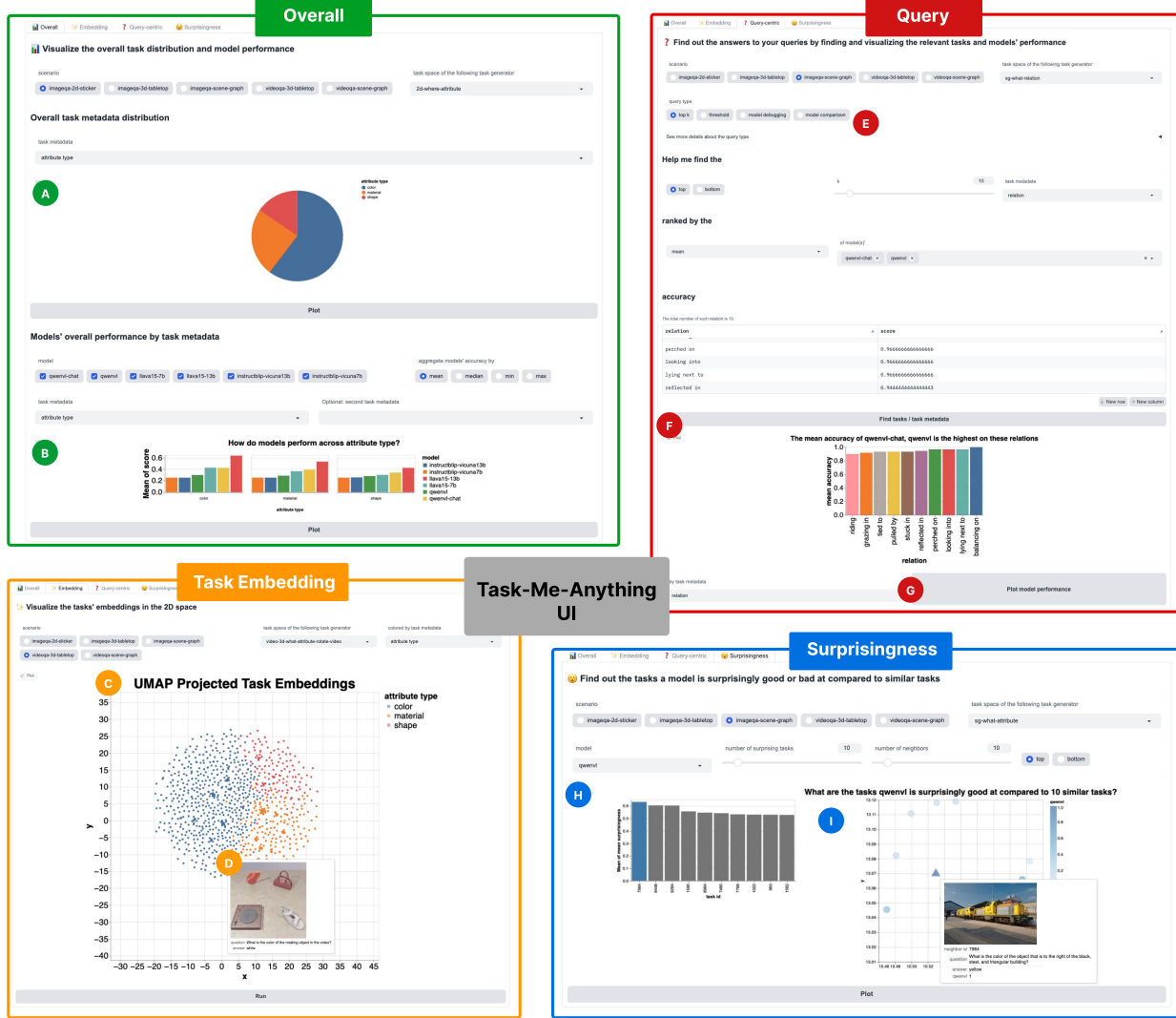


Figure B.4: TASK-ME-ANYTHING-UI Interface.

G3: Facilitate model debugging via discovery of surprising results.

To achieve these goals, we implemented a graphical user interface² with Gradio's [Abid et al., 2019] framework and used Altair [VanderPlas et al., 2018; Satyanarayan et al., 2017] 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:

²<https://huggingface.co/spaces/zixianma/TaskMeAnything-UI>

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 straightforward 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 2.5.2) 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 targeted 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

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

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.

B.4 Details of Model and Human Performance on Random Task Instances

In this section, we present the full results of our evaluation on TASK-ME-ANYTHING-RANDOM with 18 MLMs and human annotators.

B.4.1 Raw results of Figure 2.10

	<i>2D sticker image</i> (1,500)		<i>3D tabletop scene</i> (3,300)		<i>Scene Graph</i> (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
COGVL2-19B	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 B.6: TASK-ME-ANYTHING-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.

	<i>3D tabletop scene</i> (1,800)		<i>Scene Graph</i> (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

Table B.7: TASK-ME-ANYTHING-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.

B.4.2 A breakdown of Table B.6

	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 B.8: random-2D sticker image

	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 B.9: random-3D tabletop scene part 1

	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 B.10: random-3D tabletop scene part 2

	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

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

B.4.3 A breakdown of Table B.7

	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 B.12: random-3D tabletop scene

	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

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

B.5 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 TASK-ME-ANYTHING-DB, and study different query results approximation methods with this Oracle database to verify their effectiveness. We will release the TASK-ME-ANYTHING-DB for future studies of query results approximation or model performance prediction.

B.5.1 Experiment details

Setup. For image question answering tasks, We select 6 representative open-sourced large multimodal language models (MLMs) from 3 model families: INSTRUCTBLIP-7B and INSTRUCTBLIP-13B from INSTRUCTBLIP [Dai et al., 2024], QWEN-VL and QWEN-VL-CHAT from QWEN-VL [Bai et al., 2023], and LLAVA-7B and LLAVA-13B from LLAVA [Liu et al., 2024a]. 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 [Zhang et al., 2023b], VIDEO-CHATGPT-7B from VIDEO-CHATGPT [Maaz et al., 2024], CHAT-UNIVI-7B and CHAT-UNIVI-13B from CHAT-UNIVI [Jin et al., 2023], VIDEO-LLAVA-7B from VIDEO-LLAVA [Lin et al., 2023], and VIDEOCHAT2-7B from VIDEOCHAT2 [Li et al., 2024b]. We evaluate the models against a subset of tasks whose statistics can be found in Table B.14. 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.

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		

Table B.14: Statistics of evaluated tasks. For each task, we generate 15 task instances for evaluation.

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.

B.5.2 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 B.15, B.16, B.17, and B.18.

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

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 B.15: The performance of Top-K 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 B.16: 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	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 B.17: 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	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

Table B.18: The performance of Model debugging query results approximation algorithms with different budgets.

B.5.3 Query results approximation experiments in ImageQA

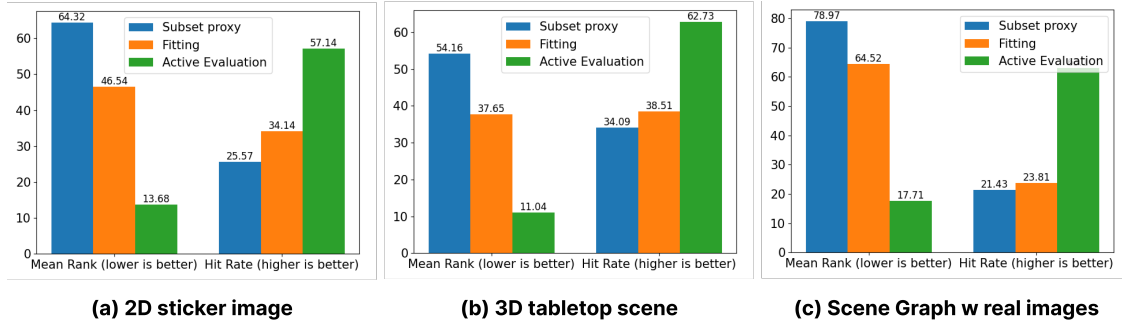


Figure B.5: Top-K Query. These three bar graphs display the performance of three query approximation methods in Top-K Query, measured by Mean Rank and Hit Rate.

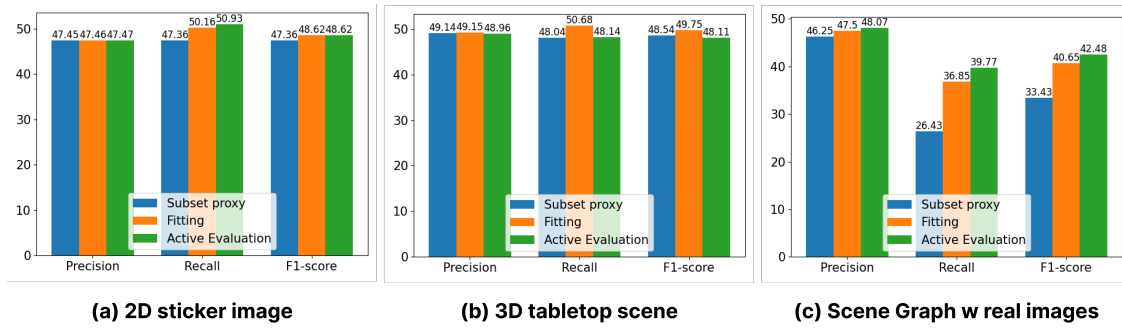


Figure B.6: Threshold Query. These three bar graphs display the performance of three query approximation methods in Threshold Query, measured by Precision, Recall, and F1-score.

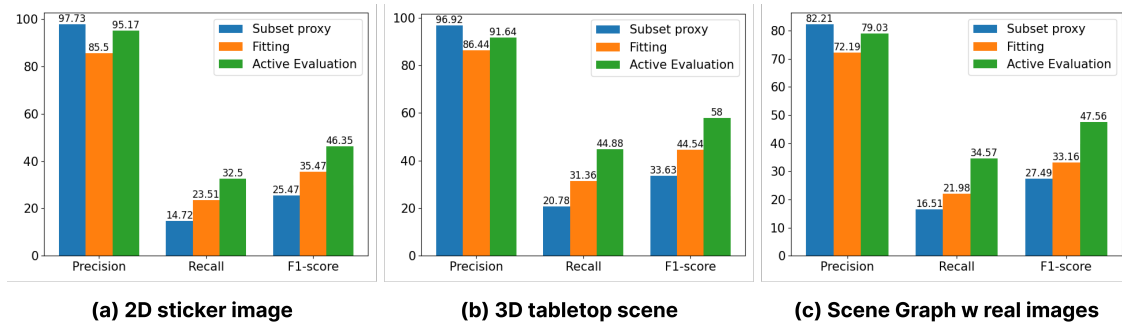


Figure B.7: Model Debugging Query. These three bar graphs display the performance of three query approximation methods in Model Debugging Query, measured by Precision, Recall, and F1-score.

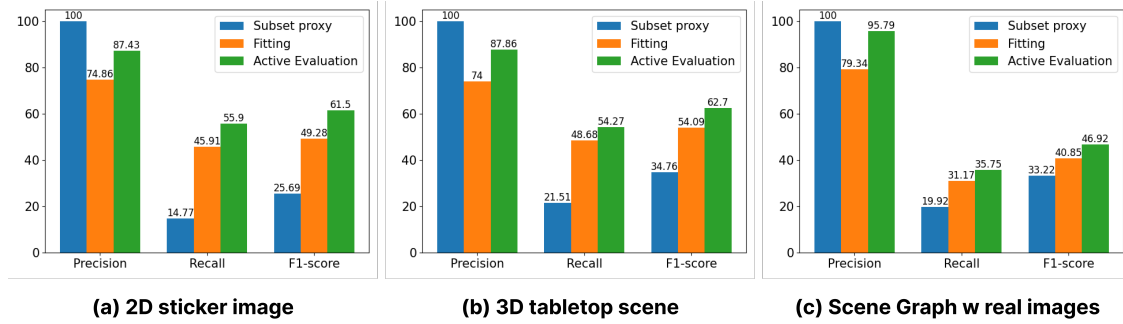


Figure B.8: Model Comparison Query. These three bar graphs display the performance of three query approximation methods in Model Comparison Query, measured by Precision, Recall, and F1-score.

B.5.4 Query results approximation experiments in VideoQA

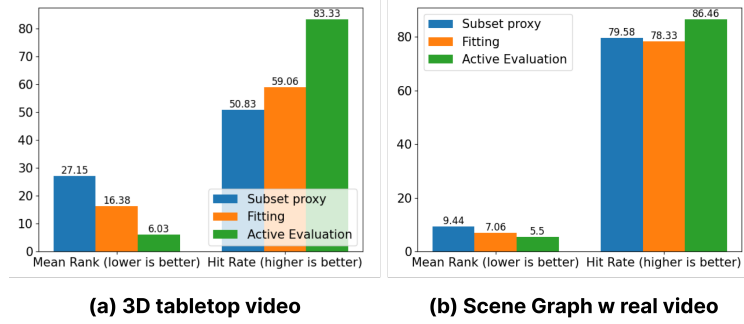


Figure B.9: Top-K Query in VideoQA. These three bar graphs display the performance of three query approximation methods in Top-K Query, measured by Mean Rank and Hit Rate.

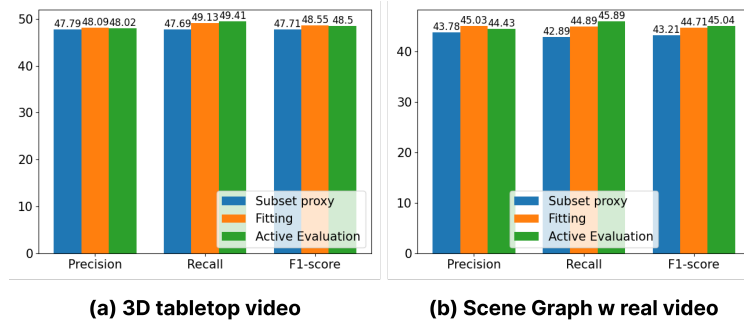


Figure B.10: Threshold Query in VideoQA. These three bar graphs display the performance of three query approximation methods in Threshold Query, measured by Precision, Recall, and F1-score.

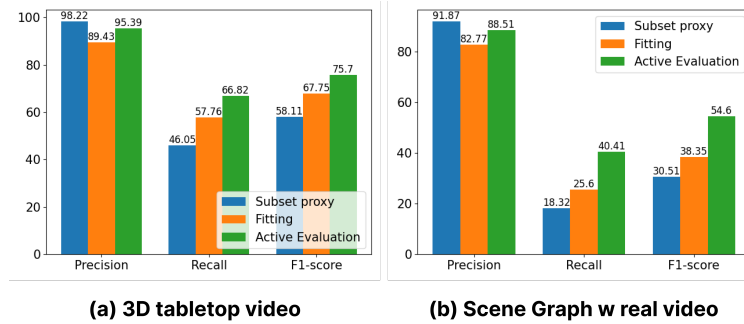


Figure B.11: Model Debugging Query in VideoQA. These three bar graphs display the performance of three query approximation methods in Model Debugging Query, measured by Precision, Recall, and F1-score.

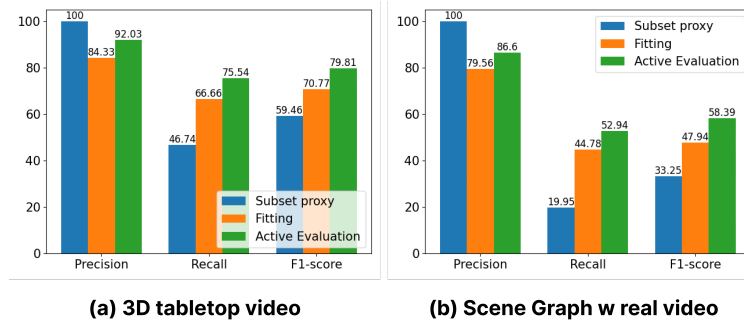


Figure B.12: Model Comparison Query in VideoQA. These three bar graphs display the performance of three query approximation methods in Model Comparison Query, measured by Precision, Recall, and F1-score.

B.6 Details of Analysis and Case Study

B.6.1 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 B.13) 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 B.14). 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 B.15); and among the actions VIDEO-LLAMA-2-13B performs the worst on, most involve "putting" or "throwing" something (Figure B.16).



Figure B.13: ImageQA: Best relations

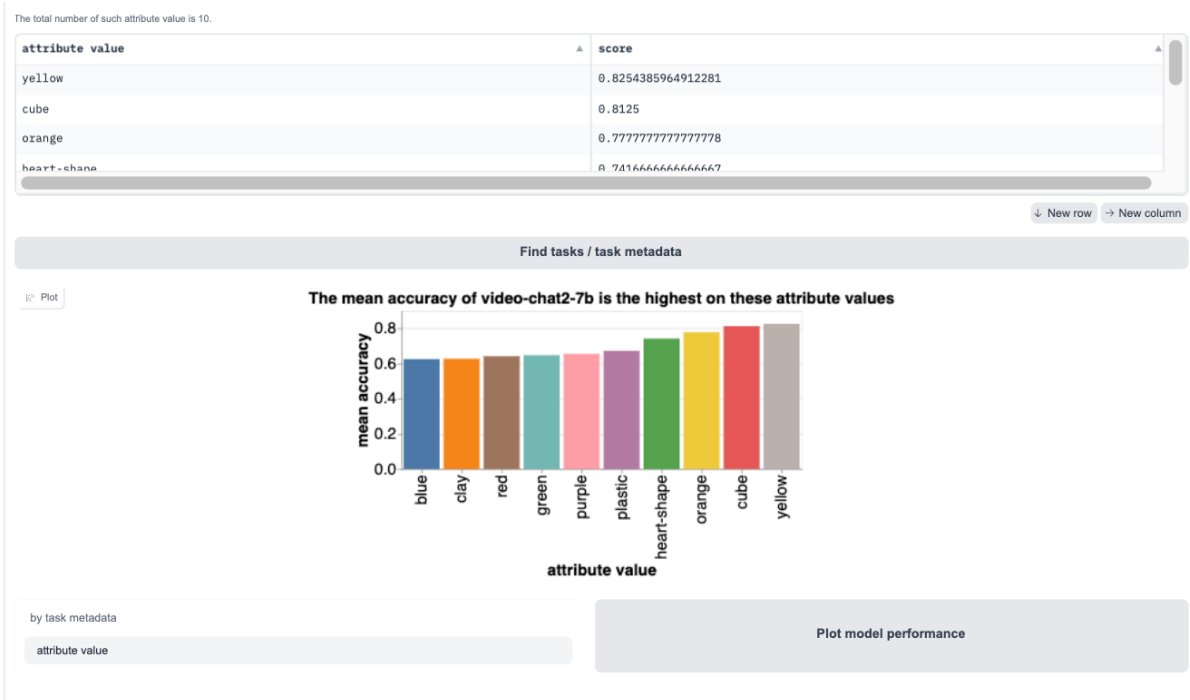


Figure B.14: VideoQA: Best attributes



Figure B.15: ImageQA: Worst positions



Figure B.16: VideoQA: Worst actions

B.6.2 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 2.17). 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 2.19).

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 B.17). 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 B.18 and B.19).



Figure B.17: INSTRUCTBLIP-7B vs. INSTRUCTBLIP-13B relations

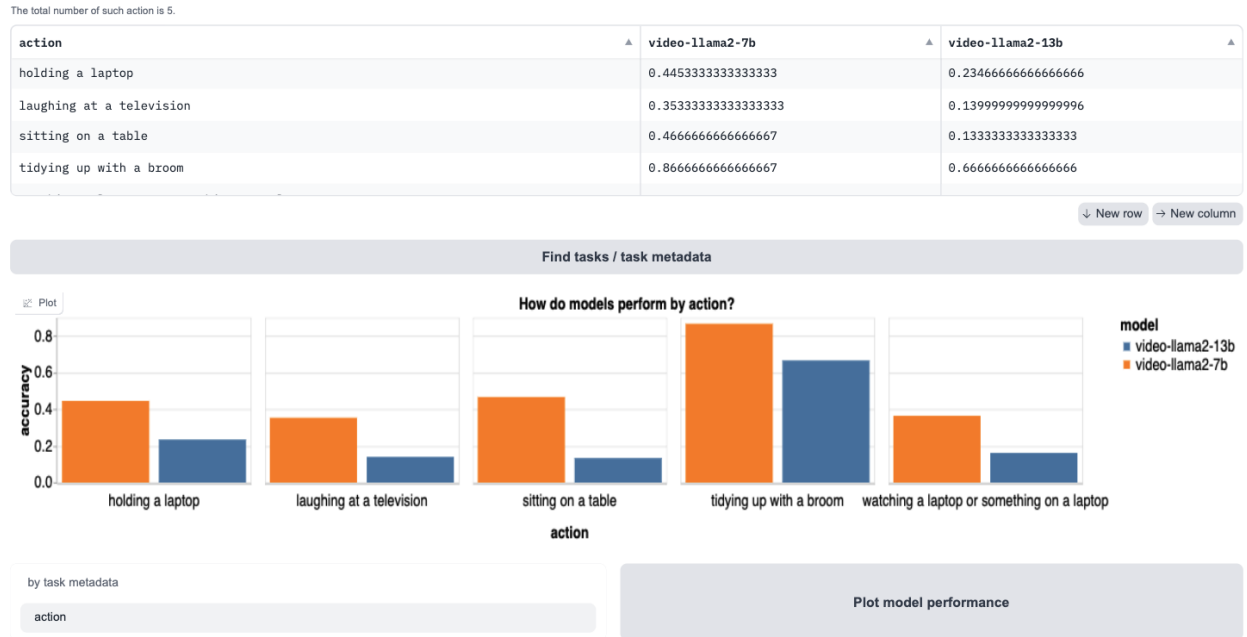


Figure B.18: VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B actions



Figure B.19: VIDEO-LLAMA-2-7B vs. VIDEO-LLAMA-2-13B objects

B.6.3 Do TASK-ME-ANYTHING yield results similar to existing benchmarks?

To check whether our TASK-ME-ANYTHING 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 [Acharya et al., 2019] (we selected 10,000 simple questions and 10,000 complex from the whole set) and 2D how-many and 3D how-many tasks in TASK-ME-ANYTHING-RANDOM. (Table B.19), the results demonstrate a notable correlation. For instance, the LLAVA-13B is the best-performing model in both TallyQA and how-many tasks in TASK-ME-ANYTHING-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 TASK-ME-ANYTHING can effectively reflect model performance in a manner similar to existing benchmark.

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

Table B.19: Models performance on TallyQA Counting benchmark and 2D how-many and 3D how-many in our TASK-ME-ANYTHING-RANDOM

Chapter C

Appendix: ProVision

C.1 Instruction Data Generator Statistics

We summarize the statistics of all 38 instruction data generators (24 single-image and 14 multi-image) in Table C.1. Each generator utilizes hundreds of pre-defined templates to ensure structural diversity in the generated instruction data.

C.2 Example Generated Instructions

We provide examples of the instruction data generated by our single-image and multi-image generators. Table C.2 shows examples from the 24 single-image instruction data generators, and Table C.3 shows examples from the 14 multi-image instruction data generators.

C.3 Results with Model-Generated Scene Graphs

We present detailed results when using instruction data generated from model-generated scene graphs (from the DataComp dataset) instead of manually-annotated scene graphs (from Visual Genome). Table C.4 shows the single-image instruction tuning results with **DC-S**, and Table C.5 shows the multi-image instruction tuning results with **DC-M**.

Task generator	# of templates	Vocabulary size
Single-image Generator		
ExistsObjectGenerator	175	27
MostObjectGenerator	675	35
LeastObjectGenerator	675	35
LeftMostObjectGenerator	675	35
RightMostObjectGenerator	675	35
TopMostObjectGenerator	675	35
BottomMostObjectGenerator	675	35
ExistsAttributeGenerator	175	27
AttributeBBoxGenerator	80	23
TypedAttributeBBoxGenerator	80	23
ExistsRelationGenerator	30	26
RelationBBoxGenerator	30	28
HeadRelationGenerator	100	27
SameObjectSegGenerator	250	32
DiffObjectSegGenerator	250	32
CloserPointGenerator	2025	35
FartherPointGenerator	1650	35
CloserObjectGenerator	2025	35
FartherObjectGenerator	1650	35
CloserToAnchorObjectGenerator	675	34
FartherToAnchorObjectGenerator	550	34
Multi-image Generator		
HasRelationMultiGenerator	10	18
HasNotRelationMultiGenerator	10	21
HasObjectMultiGenerator	15	14
HasNotObjectMultiGenerator	15	16
HasAttributedObjectMultiGenerator	15	15
HasNotAttributedObjectMultiGenerator	15	17
HasMostObjectMultiGenerator	60	19
HasLeastObjectMultiGenerator	60	19
CommonObjectMultiGenerator	135	28
CommonAttributeMultiGenerator	90	25
CountObjectMultiGenerator	900	32
CountAttributeObjectMultiGenerator	900	33
CompareRelationMultiGenerator	60	32
CompareAttributeMultiGenerator	60	29

Table C.1: Statistic of the number of templates each generator can create and the corresponding vocabulary size for each generator.

Task generator	Example question	Example answer
ExistsObjectGenerator	Tell me what is the number of stop sign that you see?	1
MostObjectGenerator	Determine from boat and sail, which object is the most commonly found?	boat
LeastObjectGenerator	Among door and drawer, what is the least frequent object?	door
LeftMostObjectGenerator	Can you tell among jeans, pants, and horses, which object is located on the far left?	pants
RightMostObjectGenerator	Can you tell among girl, hat, and sign, which object is positioned on the far right?	hat
TopMostObjectGenerator	Identify which object is located on the most upward, among cloud, floor, and water?	cloud
BottomMostObjectGenerator	Provide the extreme bottom object among dome, flags, and crosswalk.	crosswalk
ExistsAttributeGenerator	Tell me the quantity of long kites.	two
AttributeBBoxGenerator	Can you tell what are the attributes for the kite positioned at region (0.13, 0.26, 0.24, 0.47)?	blue
TypedAttributeBBoxGenerator	Provide the shape of the hat found at region (0.89, 0.47, 0.96, 0.51).	round
ExistsRelationGenerator	Can you tell what is the specific relationship between car and bus?	behind and to the left of
RelationBBoxGenerator	What kind of relationship exists between objects at (0.0, 0.76, 0.83, 1.0) and (0.79, 0.77, 0.99, 1.0)?	to the left of
HeadRelationGenerator	Which of glass, kitchen, straw, and wine is to the left of post?	straw
SameObjectSegGenerator	Can you tell the point that is in the same object as (0.83, 0.52) among (0.85, 0.54) and (0.81, 0.54)?	(0.85, 0.54)
DiffObjectSegGenerator	Identify in the list of (0.4, 0.57) and (0.45, 0.54), which point is in the part of different objects as (0.4, 0.46)?	(0.45, 0.54)
CloserPointGenerator	Determine in (0.94, 0.9) and (0.9, 0.23), what is closer point to the camera?	(0.9, 0.23)
FartherPointGenerator	Identify from (0.38, 0.26) and (0.38, 0.83), which point is positioned farther away in depth?	(0.38, 0.26)
CloserObjectGenerator	Identify what is the nearer object in depth, out of plants and olives?	olives
FartherObjectGenerator	Tell me out of wheel and man, which object is located farther to the camera?	man
CloserToAnchorObjectGenerator	In the list of ceiling and shorts, which object is located nearer to the logo in depth?	shorts
FartherToAnchorObjectGenerator	Among nose and steps, which object is positioned farther away to the dirt in depth?	nose
SceneGraphObjectQAGenerator	What is the leafy and small object that the word is to the right of?	tree
SceneGraphRelationQAGenerator	What is the relation from the object, which is behind the empty and wood shelf, to the object, which the large and lying dog is sitting on?	to the left of
SceneGraphAttributeQAGenerator	What is the color of the stone object that the green arrow is to the left of?	brown

Table C.2: Example QA for Single-Image Generator.

Task generator	Example question	Example answer
HasRelationMultiGenerator	Determine which image shows that door is to the left of man?	Image 1
HasNotRelationMultiGenerator	Tell me in which image tree isn't to the left of street light?	Image 1
HasObjectMultiGenerator	Which image shows buildings?	Image 0
HasNotObjectMultiGenerator	Tell me which image doesn't have mirror?	Image 0
HasAttributedObjectMultiGenerator	Tell me which image contains black tag?	Image 0
HasNotAttributedObjectMultiGenerator	Which image doesn't show striped mane?	Image 1
HasMostObjectMultiGenerator	Tell me which image shows the most window?	Image 1
HasLeastObjectMultiGenerator	Which image shows the least pole?	Image 0
CommonObjectMultiGenerator	Identify the objects that are seen in all of these images.	pot
CommonAttributeMultiGenerator	Identify what is common attribute of kite across these images?	flying
CountObjectMultiGenerator	Can you tell what is the number of coat in these images?	2
CountAttributeObjectMultiGenerator	Can you tell what is the number of black jacket in these images?	2
CompareRelationMultiGenerator	Determine the difference of the relation between window and windows across these images.	window is to the right of windows in Image 0, to the left of windows in Image 1.
CompareAttributeMultiGenerator	Determine what differences can be observed in the attributes of kite across these images?	kite is blue in Image 0, yellow and flying in Image 1.

Table C.3: Example QA for Multi-Image Generator.

	Data Ratio	CVB-2D	CVB-3D	SEED	MMB	MME	QBench2	MMMU	RealWorldQA	Avg.
LLaVA-1.5 instruction data		58.0	61.0	66.8	66.7	63.2	46.4	36.2	54.3	56.6
Replacement	5%	58.0	62.0	66.8	67.8	63.5	46.9	36.3	54.4	57.0
	10%	59.0	66.0	66.9	67.6	61.3	48.1	35.4	55.6	57.5
	20%	58.0	66.0	67.0	68.2	61.2	47.1	36.0	56.6	57.5
	50%	60.0	70.0	66.2	66.6	64.0	49.9	37.4	55.4	58.7
Augmentation	5%	54.0	62.0	66.7	66.8	63.4	46.0	36.9	55.7	56.4
	10%	58.0	66.0	67.4	66.8	64.5	45.7	35.6	56.3	57.5
	20%	57.0	68.0	66.7	66.1	62.9	49.8	37.3	55.0	57.9
	50%	60.0	69.0	67.2	67.2	64.3	48.6	37.9	58.2	59.0

Table C.4: Raw results of Figure 3.4a, Results of instruction tuning LLaVA-1.5-7B with **DC-S**, i.e. , single-image instruction data generated from DataComp images with our automatic scene graph generation pipeline.

		Multi-image benchmark		Single-image benchmark						
	Data Ratio	Mantis-Eval	MMT	SEED	MMB	MME	QBench2	MMMU	RealWorldQA	Avg.
Mantis instruction data		54.4	52.9	68.1	72.8	58.5	70.1	44.3	51.5	59.1
Replacement	5%	52.5	52.3	68.7	73.1	58.7	70.2	43.7	51.2	58.8
	10%	60.8	52.0	68.0	73.5	58.5	69.9	42.3	51.4	59.6
	20%	55.8	51.8	68.7	73.9	59.9	68.0	42.9	52.0	59.1
	50%	54.4	51.8	67.2	71.1	59.5	65.9	41.4	52.6	58.0
Augmentation	5%	59.4	52.5	68.4	73.2	58.1	70.1	43.3	49.4	59.3
	10%	59.9	52.7	68.6	73.1	57.8	70.4	42.2	51.6	59.5
	20%	58.5	52.5	68.8	72.4	60.5	70.5	43.0	52.8	59.9
	50%	59.0	52.7	68.6	73.1	59.0	69.9	43.7	51.4	59.7

Table C.5: Raw results of Figure 3.4b, Results of instruction tuning Mantis-SigLIP-8B with **DC-M**, i.e. , multi-image instruction data generated from DataComp images with our automatic scene graph generation pipeline.

Chapter D

Appendix: TrajViT and TrajTok

D.1 Segmenter Training Details

In the **TrajAdapter** and **TrajVLM** settings, we pretrain the trajectory segmenter once and reuse its weights for initialization during downstream probing and VLM training. This section provides full details of the dataset construction, annotation pipeline, filtering criteria, and training configuration for our segmenter training.

D.1.1 Dataset Construction

Sources. We construct a video & image corpus for segmenter training by combining: Panda (video) [Chen et al., 2024d], CC12M (image) [Changpinyo et al., 2021], CC3M (image) [Sharma et al., 2018], and a subset of DataComp-50M (image) [Gadre et al., 2024]. All samples are annotated with pseudo panoptic trajectory masks using the TrajViT trajectory-generation pipeline, followed with data filtering. We describe the details below.

Annotation Pipeline. We adopt the same annotation process as in the TrajViT paper [Zheng et al., 2025]. In summary, the pipeline consists of four steps.

1. Sample frames and detect keyframes based on feature changes in colorspace and Luminance Histogram.
2. Generate panoptic object masks in the key frames using DirectSAM [Chen et al., 2024a] model.
3. Track objects across frames via SAM2 [Ravi et al., 2025].

4. Merge instance masks between clips using heuristics like IOU overlaps to form long-term trajectories.

The pipeline uses external models like DirectSAM and SAM2 [Chen et al., 2024a; Ravi et al., 2025]. For images, only spatial segmentation steps are applied.

Quality Filtering. We apply two filtering criteria to remove low-quality pseudo labels:

- **Coverage filter:** remove samples where the union of all trajectory masks covers less than 80% of pixels.
- **Object-count filter:** remove samples containing fewer than 10 detected objects.

After filtering, we retain roughly 2.5M images and 2.0M videos for segmenter pretraining.

D.1.2 Training Configuration

The segmenter is trained on the filtered dataset. Different from TrajViT2 where all modules are trained from scratch, we initialize the ConvNext-small patch encoder from DINOv3’s weights, which helps in the generalization performance of produced segments. Other modules are initialized from scratch. We train the model for 20 epochs with 8 A100 GPUs. We use the base learning rate of 1×10^{-3} , and adopt a linear decay learning-rate schedule with warm-up. Additional hyperparameters are summarized in Table D.1.

D.2 More Qualitative Examples of the Segmenter

We show examples of generated trajectories from our training set in Figure D.1. Overall, the segmenter exhibits strong semantic grouping ability, consistently discovering object-level regions that are sufficiently accurate for downstream understanding tasks. From the perspective of pixel-level segmentation quality, however, the lightweight design and low output resolution introduce several expected limitations: the model occasionally misses very small objects, may over-merge background regions, and produces imprecise object boundaries. These imperfections, while noticeable visually, do not hinder its effectiveness as a trajectory proposal module, as our downstream tasks primarily rely on correct semantic grouping rather than pixel-perfect masks.

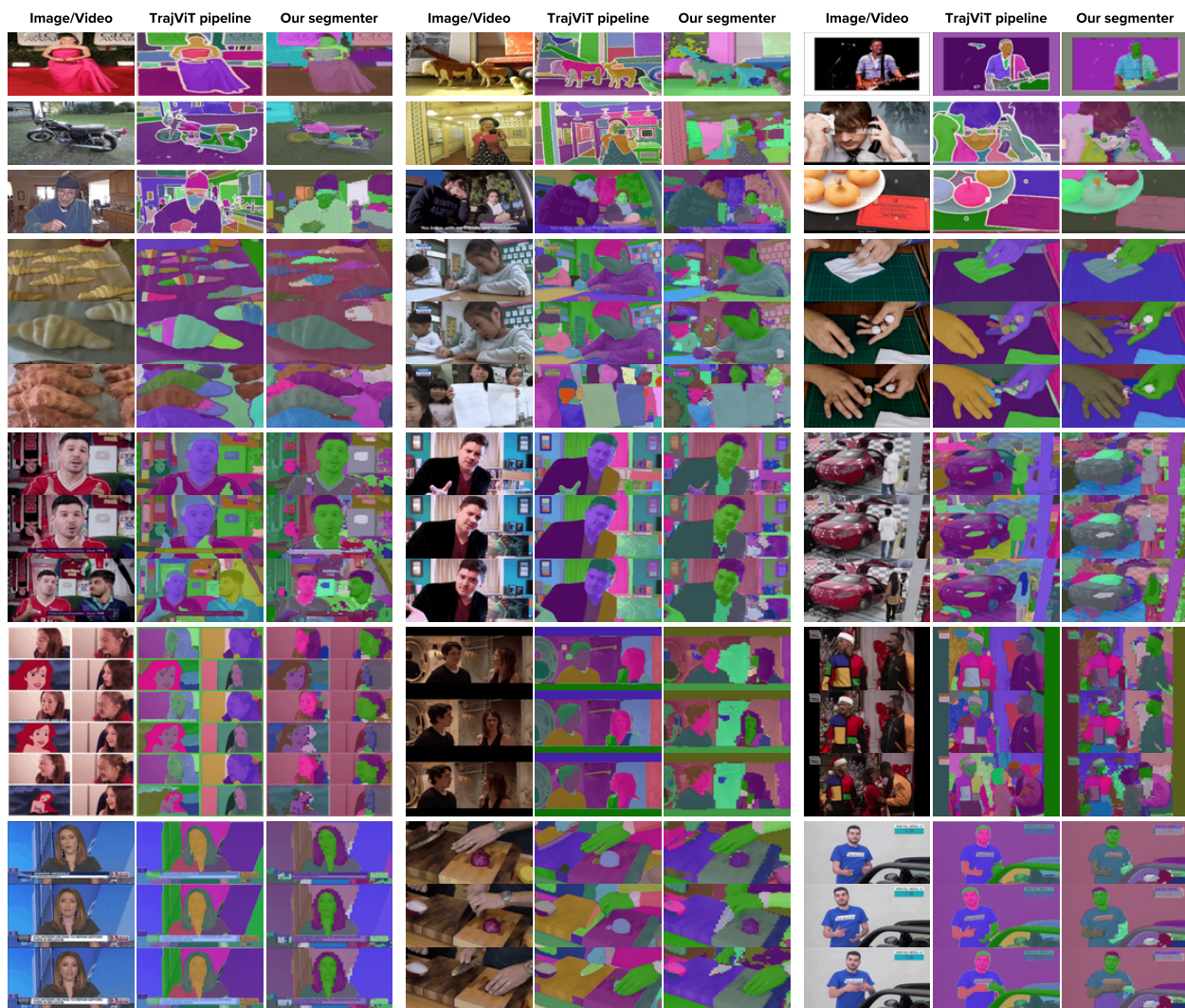


Figure D.1: Qualitative Examples of the trajectory masks produced by our segmenter.

Hyperparameter	Value
Resolution	224
Frame sampling	uniform 16 frames (videos only)
Optimizer	AdamW
Base LR	1×10^{-3}
Weight decay	0.01
Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
Batch size	video-128, image-512
Training epochs	20
LR schedule	linear decay
Warm up epochs	1
Warm up schedule	linear warm-up
Random crop scale	(0.2, 1.0)
Random crop ratio	(3/4, 4/3)
Horizontal flip probability	0.5
Color jitter probability	0.8
Gaussian blur probability	0.5
Grayscale probability	0.2

Table D.1: Hyperparameters used for segmenter pretraining.

D.3 Quantitative Evaluation of the Segmenter

Although the proposed segmenter in the main paper is intentionally lightweight—prioritizing semantic grouping over pixel-level precision—we additionally study how well it can perform on the standard panoptic video segmentation task when its capacity is scaled up. This experiment is conducted purely for analysis and is *not* used by any model in the main paper.

Scaling up the segmenter. We keep the same training dataset as described in Sec. D.1.1, but increase the segmenter capacity in two ways: (1) replacing the ConvNeXt-Tiny patch encoder with a ConvNeXt-Large backbone and expanding the Perceiver stack from 2 layers to 4 layers, and (2) producing full-resolution

predictions by adding a pixel decoder identical to the one used in SAM [Ravi et al., 2025], applied on top of the downsampled patch features. The input and output resolution are both set to 512×512 .

Benchmark and competitors. We evaluate on the ViPEntitySeg [Miao et al., 2022] benchmark and compare against two state-of-the-art video panoptic segmentation systems: EntitySAM [Ye et al., 2025] and SAM 2.1 [Ravi et al., 2025]. We report VEQ-SQ, VEQ-RQ, and STQ-EN following the benchmark protocol.

Results. Table D.2 shows that the scaled-up version of our segmenter achieves competitive performance, surpassing EntitySAM in VEQ-SQ and improving VEQ-RQ relative to SAM 2.1. While its STQ-EN score is slightly lower than EntitySAM, these results demonstrate that our grouping-centric design can approach state-of-the-art performance when augmented with a strong visual backbone and a full-resolution decoder, confirming our segmenter design is reasonable.

VIPSeg Benchmark	VEQ-SQ	VEQ-RQ	STQ-EN
EntitySAM	84.7	64.5	43.3
SAM 2.1	83.1	36.7	41.7
Ours (scaled-up)	85.5	45.1	40.2

Table D.2: Evaluation of a scaled-up version of our segmenter on the ViPEntitySeg benchmark. This model is *not* used in any main-paper experiments; it serves only as an isolated study of segmentation quality under increased capacity.

D.4 Training Details for TrajViT2

For all TrajViT2 experiments and baseline models, we optimize using the AdamW optimizer with a base learning rate of 10^{-4} , weight decay of 10^{-2} , and mixed-precision training. We use a cosine annealing schedule with a linear warm-up of one epoch. The contrastive batch size is 128 for video clips and 1024 for images. All models are trained for 20 epochs using 8 NVIDIA A100 GPUs. During training, we apply standard video augmentation including random ColorJitter, Grayscale, Gaussian blur, horizontal flip, and resized cropping. At evaluation, we use only a single resizing operation for consistency. All models adopt a ViT-Large transformer and operate on 224-resolution inputs with 16 uniformly sampled frames.

D.5 Training Details for TrajAdapter

For all TrajAdapter experiments, we follow the standard protocol for probing pretrained video encoders. We use the AdamW optimizer with a learning rate of 1×10^{-4} and weight decay of 0.5. The pretrained backbone is kept frozen, while the trajectory encoder and probing head are updated. Before classification, video features are layer-normalized. We train with a batch size of 128 for 10 epochs. This configuration is used for all TrajAdapter experiments on both Kinetics-400 and Something-Something-V2 probing tasks.

D.6 Training Details for TrajVLM

Data sources. TrajVLM is trained using a two-stage procedure. For the pretraining stage, we closely follow the Molmo training paradigm [Deitke et al., 2025] and use the same PixMo captioning split to align visual representations with the language model. For the instruction-tuning stage, we adopt the mixture of public academic VideoQA datasets and synthetic QA pairs curated in Molmo-2 [Clark et al., 2025]. These datasets span a wide range of reasoning skills—including temporal grounding, causal inference, long-horizon understanding, and multi-step procedural reasoning. In total, the mixture contains approximately 5 million training examples.

Training hyperparameters. The training hyperparameters of the first stage follow Molmo [Deitke et al., 2025]. We summarize the training hyperparameters for the second stage in Table D.3.

Hyperparameter	Value
Max video frames	128
Total training steps	10,000
Training stages	Pretraining + Instruction tuning
Sequence length	8192
Global batch size	32
Device batch size	4
Number of GPUs	$8 \times \text{A100 (80GB)}$
Precision	bfloat16 (AMP)
Optimizer	AdamW
LLM learning rate	1×10^{-5}
ViT learning rate	5×10^{-6}
Connector learning rate	5×10^{-6}
Learning rate warmup	200 steps
LR schedule	Multimodal cosine decay
Weight decay	0.0 (LLM / ViT / connector)
Adam momentum	$\beta_1 = 0.9, \beta_2 = 0.95$
Adam ϵ	1×10^{-6}
Gradient clipping	1.0

Table D.3: Key hyperparameters used for training TrajVLM. Values reflect the shared configuration across all VLM experiments.

Chapter E

Appendix: Molmo2

E.1 Model Architecture and Training Details

We provide the complete model architecture and training specifications for all Molmo2 variants in Table E.1.

E.2 Image Benchmark Results

We evaluate Molmo2 on a comprehensive suite of single-image benchmarks. Table E.2 compares Molmo2 against both proprietary and open-weight models across 16 image understanding benchmarks.

E.3 NLP Benchmark Results

To verify that Molmo2 maintains strong language capabilities, we evaluate on standard NLP benchmarks including MMLU, GSM8K, ARC-C, and MBPP+ in Table E.3.

E.4 Additional Video Counting and Pointing Results

We provide a detailed breakdown of video counting and pointing performance across different query categories and count ranges in Tables E.4, E.5, and E.6.

		4B	7B	8B
Image Encoder	Params		380m	
	Dim		1152	
	MLP Dim		4304	
	Act.		GELU	
	Heads		16	
	KV Heads		16	
	Layers		27	
	Image Size		384×384	
	Patch Size		14	
	Dropout		0.0	
V/L Connector	Params	57m	80m	88m
	Image Pool Size		2×2	
	Video Pool Size		3×3	
	Pool Dim		1152	
	Pool Heads		16	
	MLP Dim	9728	100352	12288
	Act.		SwiGLU	
	Dropout		0.0	
LLM	Params	4.0b	7.3m	8.2m
	Embed	151936	100352	151936
	Dim	2560	4096	4096
	MLP Dim	9728	11008	12288
	Act.		SwiGLU	
	Heads		32	
	KV Heads	8	32	8
	Layers	36	32	36
	Theta	1m	0.5m	1m
	Dropout		0.1	
Pre-Train	Warmup ViT		2000	
	Warmup Con.		200	
	Warmup LLM		2000	
	LR ViT		6e-6	
	LR Con.		2e-4	
	LR LLM		2e-4	
	Cosine Decay		10%	
	Eps.		1e-6	
	Betas		0.9, 0.95	
	Batch Size		128	
	Sequence Length		2560	
	Steps		32k	
Fine-Tune	Warmup ViT		200	
	Warmup Con.		200	
	Warmup LLM		200	
	LR ViT		5e-6	
	LR Con.		5e-6	
	LR LLM		1e-5	
	Cosine Decay		10%	
	Eps.		1e-6	
	Betas		0.9, 0.95	
	Batch Size		128	
	Sequence Length		16384	
	Steps		25k	

Table E.1: Model and training hyper-parameters, Molmo2-7B is a version of Molmo2 with the OLMo 2 language model.

Model	A12D	ChartQA	DocVQA	InfoQA	TextVQA	VQA v2.0	RWQA	MMMU	MathVista	CountBench	PixMoCount	MuirBench	MMIU	Blink	Img QA avg.	Multimlg QA avg.	Average
<i>API call only</i>																	
GPT-5	89.5	83.8	88.9	83.0	78.7	79.7	80.8	81.8	82.7	90.8	67.2	78.6	71.0	66.5	82.5	72.1	80.2
GPT-5 mini	86.7	82.1	86.7	82.2	79.1	72.1	77.0	78.7	79.2	87.1	74.4	71.4	64.5	68.7	80.5	68.2	77.8
Gemini 2.5 Pro	94.3	77.8	91.5	82.0	70.3	67.1	77.4	79.6	84.6	90.8	73.8	74.5	68.9	73.7	80.8	72.4	79.0
Gemini 2.5 Flash	95.9	76.8	91.1	80.9	73.0	69.4	74.5	79.0	81.2	86.7	63.9	73.5	61.2	70.2	79.3	68.3	76.9
Claude Sonnet 4.5	91.5	80.2	91.7	65.9	67.2	77.0	61.1	77.8	73.1	87.3	58.3	59.6	54.1	64.8	75.6	59.5	72.1
<i>Open weights only</i>																	
InternVL3.5-4B	82.6	86.0	92.4	78.0	77.9	78.1	66.3	66.6	77.1	82.2	62.4	53.1	49.2	58.1	77.2	53.5	72.1
InternVL3.5-8B	84.0	86.7	92.3	79.1	78.2	79.5	67.5	73.4	78.4	79.6	61.9	55.8	49.4	59.5	78.2	54.9	73.2
Qwen3-VL-4B	84.1	85.0	95.3	80.3	81.0	81.7	70.9	67.4	73.7	85.5	58.0	63.8	43.2	65.8	78.4	57.6	74.0
Qwen3-VL-8B	85.7	85.2	96.1	83.1	82.8	82.3	71.5	69.6	77.2	90.4	65.0	64.4	35.3	69.1	80.8	56.3	75.6
Keye-VL-1.5-8B	89.5	85.0	93.4	74.9	81.5	79.3	73.5	71.4	81.2	81.6	57.4	51.2	50.3	54.9	79.0	52.1	73.2
GLM-4.1V-9B	87.9	70.0	93.3	80.3	79.6	68.3	70.7	68.0	80.7	88.0	60.7	74.7	62.4	65.1	77.0	67.4	75.0
MiniCPM-V-4.5-8B	86.5	87.4	94.7	73.4	82.2	64.1	72.1	67.7	79.9	83.9	62.8	53.3	46.5	42.0	77.7	47.3	71.2
Eagle2.5-8B	84.5	87.5	94.1	80.4	83.7	82.4	76.7	55.8	67.8	90.2	66.9	61.8	48.4	45.8	79.1	52.0	73.3
<i>Open models</i>																	
PLM-3B	90.9	84.3	93.8	74.6	84.3	84.4	72.4	41.2	59.1	87.1	63.0	25.7	40.6	55.4	75.9	40.6	68.3
PLM-8B	92.7	85.5	94.6	80.0	86.5	85.6	75.0	46.1	59.9	91.8	68.0	23.5	27.4	56.0	78.7	35.7	69.5
<i>Molmo2 family: Open weights, Open data (no distillation), Open code</i>																	
Molmo2-4B	95.5	85.8	92.4	79.2	85.6	86.8	74.6	49.7	57.4	94.3	89.1	61.2	54.3	57.0	80.9	57.5	75.9
Molmo2-8B	96.2	86.8	93.9	81.3	86.4	87.3	77.5	53.8	61.0	93.7	90.7	64.8	53.7	54.0	82.6	57.5	77.2
Molmo2-7B	93.6	83.7	92.0	76.5	85.4	86.9	75.4	43.9	48.9	93.9	87.4	55.9	49.8	57.1	78.9	54.3	73.6

Table E.2: Image benchmark results for a range of proprietary APIs, open-weight baselines, and our Molmo2 family across image understanding and counting benchmarks. The result of the best-performing open-weight model is in **bold**.

Model	MMLU	GSM8K	ARC-C	MBPP+
Qwen3 4B	72.2	87.8	83.3	59.5
Qwen3 8B	76.8	89.8	88.3	62.2
Molmo2 4B	71.7	87.0	85.6	52.9
Molmo2 8B	76.8	89.1	90.3	50.3

Table E.3: Results on selective NLP benchmarks, including MMLU for general knowledge QA, GSM8K for math, ARC-C for reasoning, and MBPP+ for coding tasks.

E.5 Tracking Dataset Statistics

Table E.7 provides detailed statistics of the Molmo2-VideoTrack training dataset and the Molmo2-Track evaluation benchmark.

E.6 Pre-training Ablation

We ablate the effect of adding pointing data during pre-training on downstream tasks in Table E.8.

Model	Query Catogery			Avg.
	Action/Event	Animal	Object	
<i>API call only</i>				
GPT-5	46.6	75.5	29.8	50.6
GPT-5 mini	36.2	63.3	25.1	41.5
Gemini 2.5 Pro	53.4	63.3	30.0	48.9
Gemini 2.5 Flash	36.2	63.3	27.7	42.4
Claude Sonnet 4.5	26.3	53.1	24.3	34.6
<i>Open weights only</i>				
QwenVL3-4B	39.7	59.2	19.5	39.4
QwenVL3-8B	43.1	75.5	22.5	47.0
<i>Molmo2 family: Open weights, Open data, Open code</i>				
Molmo2-4B	39.7	63.3	25.4	42.8
Molmo2-8B	43.1	71.4	25.4	46.6

Table E.4: Molmo2-VideoCount accuracy by query category.

E.7 Long Context Training

Table E.9 shows the effect of post-training with longer context (384 frames with context packing) compared to the baseline (128 frames) on long video benchmarks.

E.8 Image-Only Ablation

E.9 Additional Video Model Ablations

Model	Object Count						Avg.
	0–5	5–10	10–15	15–20	20–25	25–60	
<i>API call only</i>							
GPT-5	64.4	34.1	31.3	16.2	11.1	10.5	27.9
GPT-5 mini	55.7	28.2	25.0	10.8	6.3	10.5	22.8
Gemini Pro 2.5	61.5	31.3	31.5	15.7	17.5	13.0	28.4
Gemini Flash 2.5	56.9	31.0	27.5	19.2	9.8	3.5	24.6
Claude Sonnet 4.5	48.0	24.7	20.3	14.9	15.9	5.4	21.5
<i>Open weights only</i>							
QwenVL3-4B	56.9	17.6	21.3	2.7	3.2	0.0	16.9
QwenVL3-8B	63.8	30.6	15.0	6.8	6.3	0.0	20.4
<i>Molmo2 family: Open weights, Open data, Open code</i>							
Molmo2-4B	54.6	25.9	22.5	17.6	12.7	10.5	24.0
Molmo2-8B	58.6	30.6	16.2	21.6	6.3	12.3	24.3

Table E.5: Molmo2-VideoCount accuracy by object count.

Model	F1	Precision	Recall	Valid
<i>API call only</i>				
GPT-5	2.9	2.9	2.9	100.0
GPT-5 mini	3.6	3.6	3.6	100.0
Gemini Pro 2.5	17.0	17.3	16.9	98.5
Gemini Flash 2.5	8.4	8.1	9.5	90.5
Claude Sonnet 4.5	2.9	2.9	2.9	98.5
<i>Open weights only</i>				
QwenVL3-4B	0.0	0.0	0.0	85.4
QwenVL3-8B	1.5	1.5	1.5	92.0
<i>Molmo2 family: Open weights, Open data, Open code</i>				
Molmo2-4B	28.4	28.6	28.7	100.0
Molmo2-8B	32.9	32.3	35.2	100.0

Table E.6: Molmo2-VideoPoint full results.

Data Source	Type (Ann.)	# Clips	# Tracks	# Queries	Avg # Obj/Q
VIPSeg	General (Segm)	675	2,562	3,497	2.00
SAM-V	General (Segm)	1,090	2,282	2,537	1.43
MOSEv2	General (Segm)	463	1,107	1,168	2.08
MOSE	General (Segm)	337	863	880	1.91
TeamTrack	Sports (Bbox)	154	899	1,158	2.13
SoccerNet	Sports (Bbox)	56	445	561	2.70
SportsTrack	Sports (Bbox)	56	397	481	2.69
BDD100K	Auto. Driving (Bbox)	450	1,812	1,894	3.18
AnimalTrack	Animals (Bbox)	97	429	526	3.26
BFT	Animals (Bbox)	30	204	334	2.07
UAV-MOTD	UAV (Bbox)	142	426	437	3.43
SeaDrones	UAV (Bbox)	79	343	371	2.10
MOT20	Pedestrians (Bbox)	147	605	645	2.88
DanceTrack	Dancers (Bbox)	88	436	569	3.14
Total	All	3,612	12,501	15,058	2.28

(a) Statistics for the Molmo2-VideoTrack dataset.

Data Source	Type (Ann.)	# Clips	# Tracks	# Queries	Avg # Obj/Q
SAM-V	General (Segm)	29	89	123	1.24
SportsMOT	Sports (Bbox)	42	143	145	3.34
APTV2	Animals (Bbox)	211	401	486	1.52
PersonPath	Pedestrians (Bbox)	78	222	226	2.06
DanceTrack	Dancers (Bbox)	41	124	132	4.49
Total	All	401	979	1,112	2.19

(b) Statistics for the Molmo2-Track benchmark

Table E.7: Distribution of tracking dataset for Molmo2-VideoTrack (train) and Molmo2-Track (benchmark). We report the number of unique video clips, unique tracks, total queries, and average number of objects per query (Avg # Obj/Q) for each dataset. Type indicates video category; Ann. indicates original tracking annotation format (Segm: segmentation masks, Bbox: bounding boxes).

Pretrain	Vid. QA	Vid. Cap	Img. QA	Img. Point
With pointing	66.8	31.8	80.9	73.0
No pointing	65.9	31.3	80.1	71.8

Table E.8: Pre-training ablations. Columns show the average of our 12 video benchmarks, using validation sets for EgoSchema, PerceptionText, and MLVU, video captioning F1, the average of the 11 image benchmarks using validation sets for InfoQA, DocQA, ChartQA, VQA v2, and AI2D, and the average score in PointBench.

Model	Video-MME	Video-MME-Sub	LongVideoBench	MLVU	LVBench	VideoEvalPro	Short QA avg	Long QA avg
128 frames	68.8	74.3	65.9	74.5	49.6	54.3	69.3	64.0
384 frames w/ CP	67.8	74.7	66.2	75.0	50.3	55.9	68.5	65.0

Table E.9: Molmo2-4B with long context post-training. Post-training our model with a longer context length helps improve performance on benchmarks with long videos (>3 minutes).

Model	11 Average
Molmo2-4B	80.7
Image-Only	79.8

Table E.10: Image Specialization. Scores on the 11 image benchmark average from. Molmo2 outperforms the specialized version showing positive video to image transfer.

Data	Vid. QA	Vid. Cap. F1
Video-Only	64.8	32.6
Video pool size 3x3 to 4x4	64.3	28.5
No time tokens	<u>64.5</u>	<u>29.3</u>

Table E.11: Additional model ablation. Increasing pool size and removing time tokens hurts the model performance.