

Author Contributions

Christopher Clark, Jieyu Zhang, Zixian Ma, JaeSung Park, Rohun Tripathi, Sangho Lee and Mohammadreza Salehi collectively contributed to dataset construction, model training, and conducted numerous exploratory experiments for this project.

Christopher Clark led the project and focused on video modeling and training strategies, including experiments with the SFT mixture, the pre-training approach, and video modeling. He also wrote much of the core training code and implemented the packing and message tree systems.

Jieyu Zhang co-led the data effort on video datasets. He collected and filtered raw videos for Molmo2 video caption, video QA, and video pointing datasets, and contributed to the curation of these datasets. He helped the integration of other training/evaluation datasets and ran evaluations for many baseline models. He also helped add subtitle understanding to the model and ablations of the video SFT/caption models.

Zixian Ma co-led the data effort on video datasets. She designed human data collection interfaces and implemented them with help from Yinuo Yang. She collected the Molmo2-Cap, Molmo2-AskModelAnything, and Molmo2-VideoPoint datasets via Prolific. She led the training ablations on video counting and pointing and helped integrate academic training datasets. She ran the human preference and NLP evaluations.

Jae Sung Park led the effort to add tracking capability to Molmo2 as points. Together with Zhongzheng Ren and Vincent Shao, he designed the Molmo2-Track human annotation collection, curated existing academic tracking datasets for training, and built the pipeline to extract accurate point tracks. He introduced auxiliary grounding and single-point tracking objectives and performed ablations on mixtures of video tracking tasks. He and Zhongzheng Ren designed tracking evaluations across diverse VLMs and segmentation models.

Mohammadreza Salehi led the long-context post-training and co-led sourcing videos for training. He also contributed to training dataset construction, training on a mixture of images and videos, and evaluation of Molmo and API models.

Rohun Tripathi primarily worked on efficient modeling strategies. He developed learned and training free solutions to token allocation for different frames, with and without the input query. He implemented the initial training pipeline and details such as 3D position encoding and time tokens. He helped with training/evaluation set integrations, with a focus on long video understanding.

Sangho Lee led improvements to image modeling and training strategies and extended them to the multi-image setting. He also supported and directly conducted extensive ablation studies to develop effective training strategies for video modeling. In addition, he implemented the Hugging

Face model and processor code and vLLM integrations.

Chris Dongjoo Kim led the data effort for multi-image datasets. In collaboration with Weikai Huang and Sangho Lee, he curated the MultiImageQA dataset. He also held full responsibility for the multi-image pointing capability, including dataset curation algorithms and model training.

Yue Yang led data curation for text-rich multi-image datasets, synthetically generating diverse question-answer pairs grounded in images such as charts, tables, and documents.

Zhongzheng Ren, Yinuo Yang, Vincent Shao, Weikai Huang, and Ziqi Gao all made significant dataset contributions.

Jitesh Jain, Jianrui Zhang, and George Stoica contributed to research discussions throughout the project and did exploratory experiments based on Molmo2.

Taira Anderson managed the project.

Winson Han designed the figures in this report.

Ali Farhadi advised the project.

Ranjay Krishna was the PI for the project.

Appendix

The appendix includes the following sections:

- §A - Model details
- §B - Training details
- §C - Evaluation details
- §D - Additional results
- §E - Test time scaling and SlowFast encoding
- §F - Data details
- §G - Data examples
- §H - Related Works
- §I - Limitations
- §J - Qualitative results

A. Model details

We present additional details about image encoding, hyperparameters, and implementation choices.

Image crops. Our method of encoding images largely follows Molmo [29], including the use of overlapping crops. Unlike Molmo, we do not pad crops with black. Instead, we resize them to 378 (even if that means changing the aspect ratio), following how SigLIP 2 [139] was trained. If the number of image patches is not evenly divisible by the pooling size, the bottom and far-right image patches are pooled with a reduced number of patches.

Video frames. We use torchcodec* to extract frames from videos. We extract frames at S fps and the last frame. If that leads to more than F frames, we instead extract frames uniformly, including the first and last frames. For tracking, during training, we always sample videos at S fps and trim both videos and point tracks to a maximum of F frames instead. This ensures that points, which are annotated for S fps, remain aligned with the sampled frames. We include the last frame since it is typically what is shown when the video ends and, therefore, can have special importance to users. Frames are extracted based on timestamps (instead of frame indices) to handle variable fps videos.

Formatting. Videos and image tokens are always inserted first, right after the BOS token. We insert different start and end special tokens for videos, tokens from a multi-crop image, and tokens for the low-resolution single-crop version of the image. Frames are interleaved with text timestamps written as seconds to one decimal point, and multi-images are interleaved with “Image 1”, “Image 2”, *etc.*, labels. Text is added after the image/video tokens following the Qwen3 [169] prompt template without thinking tokens.

Pointing. Our pointing format provides points in an HTML-like format, with the coordinates stored in a compact string. For each frame or image with points, the string contains an image index (for image input, starting at 1) or a frame timestamp (for video, shown in seconds with one

decimal point), followed by a list of point coordinates. The points each have an *object index*, which is unique for each distinct object being pointed at, and x and y coordinates that are normalized to be between 0 and 1000. Object indices are sequential, starting at 1. The object indices both facilitate counting, because the final object index represents the total count, and enable tracking by identifying repeating objects. Points are sorted by time/frame index and then by x and y coordinates. Values are space-separated, with semi-columns indicating a new frame/image. We elect to use this format over a format like JSON since it dramatically reduces the number of tokens needed to represent points.

An example output for a pointing and tracking task are shown below (new lines added for clarity):

```
<points coords="1 1 555 169;2 3 649
154 4 709 162;5 5 758 175 6 808 183 7
852 187">
Inline text
</points>
```

```
<tracks coords="0.0 1 635 522;0.5 1
606 490 2 511 124;1.0 2 515 164;1.5 2
520 168">
Inline text
</tracks>
```

Where image indices and frame timestamps are in blue, object indices are in purple, and x and y coordinates are in green. The first example points to an object in images 1, 2, and 5. The second one tracks two different objects through several frames. The “Inline text” is used to describe what is being pointed at.

Hyperparameters. Hyperparameters for the Molmo2 models are shown in Table 8. The connector MLP uses the same intermediate dimension as the LLM, so its size depends on the LLM; otherwise, they are the same across all models. All models use the SigLIP 2 So400m/14 384px ViT [139].

Implementation. Our implementation uses PyTorch with Fully Sharded Data Parallel (FSDP) 2 [185]. We use PyTorch’s Scaled Dot Product Attention (SDPA), not FlashAttention [27, 28], since it does not support custom attention masks. We use `torch.compile` to improve throughput and ensure that the shapes in the LLM and ViT are static so the model can be statically compiled, which we find essential for maximizing throughput.

To improve throughput, we also utilize PyTorch’s Automatic Mixed Precision (AMP) module*, which enables most operations to run in half-precision with bfloat16 numbers. Computations for layer normalization [9] and Rotary

*<https://pytorch.org/blog/torchcodec/>

*<https://pytorch.org/docs/stable/report/amp.html>

		4B	7B	8B
Image Encoder	Params		380m	
	Dim		1152	
	MLP Dim		4304	
	Act.		GELU	
	Heads		16	
	KV Heads		16	
	Layers		27	
	Image Size		384384	
	Patch Size		14	
	Dropout		0.0	
V/L Connector	Params	57m	80m	88m
	Image Pool Size		22	
	Video Pool Size		33	
	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		
SFT	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		30k		

Table 8. **Model and training hyper-parameters**, Molmo2-O-7B is a version of Molmo2 with OLMo 3 [112]. Long-context post-training used the same parameters as SFT

Position Embedding (RoPE) [131] are still carried out in full precision.

When computing gradients, each GPU computes a gradient on a small mini-batch of examples, after which the gradients are averaged across all devices. We always compute the per-device gradient by dividing the total loss on that device by the *average* number of loss tokens across all devices, not the number of loss tokens on that particular device. This avoids a subtle bias that effectively up-weights examples with a small number of loss tokens (*e.g.*, with

short responses)* [51].

During fine-tuning, mixing is done within each batch so that the batches contain examples from a variety of datasets. We truncate examples that are longer than the max sequence length. This occurs in $< 0.1\%$ of cases, usually due to videos with both subtitles and a large number of annotations. We find training to be stable, without loss spikes or NaNs.

B. Training details

In this section, we provide additional details about packing, the data mixture, and other components of how Molmo2 was trained.

Packing. Our packing algorithm keeps a pool of $M = 48$ examples that have already been preprocessed and converted into a tokenized representation. If the pool is not full, examples are drawn from the training mixture and added to the pool. When the pool is full, we run a dynamic programming solver to find the optimal subset of examples that maximizes $T + I * w_i$ subject to $T \leq 16384$ and $I \leq 128$, where T is the total number of text tokens in the selected subset, I is the total number of crops, and $w_i = 30$ is a hyperparameter. During long context training, we instead use a max of 384 images and 36864 tokens. The selected examples are yielded as a single packed sequence and removed from the pool. In practice, we run the solver on a quantized version of the problem by rounding the number of tokens to the nearest multiple of 32.

Increasing M quickly leads to diminishing returns in terms of packing efficiency. We do not observe any gains from using more than 48. The algorithm is usually robust to w_i , but we observe that in some settings, if w_i is too low, the pool can become filled with examples with 128 crops, which usually cannot be packed with anything else, thereby reducing efficiency.

Implementation-wise, we add this logic into torch’s *DataLoader* so that each data-worker runs this algorithm independently. This makes the algorithm easy to use, but it does add some unnecessary overhead when there are many data workers. This could be addressed in future work through a deeper integration into torch’s data-loading logic. In practice, we find that packing still does not slow down the training speed. Loading and extracting frames from videos remains, by far, the most costly part of data loading.

Pre-training. During pre-training, we use response-only dropout, *i.e.*, residual dropout on just the output tokens, of 0.1, length conditioning, and both the caption and transcript, following Molmo [29].

SFT. The full list of datasets in our SFT mixture is shown in Table 9, and visualized in Figure 4. During SFT we use regular residual dropout of 0.1.

*<https://unsloth.ai/blog/gradient>

name	rate	visual	anno.	ex.
IMAGE QA	22.7	2.7m	32m	2.4m
PixMo-Clocks	1.9	800k	800k	800k
Llava-665k-Multi	1.5	280k	2.5m	160k
TallyQA	1.4	130k	250k	130k
CoSyn-chart	1.3	120k	1.1m	120k
NLVR2	1.1	100k	86k	86k
VQA v2	1.1	83k	440k	83k
CoSyn-doc	1.0	71k	610k	71k
A-OKVQA	1.0	33k	34k	34k
CoSyn-math	1.0	67k	67k	67k
CoSyn-table	0.8	47k	420k	47k
DocVQA	0.7	10k	39k	39k
CoSyn-diagram	0.7	35k	300k	35k
TextQA	0.7	22k	35k	35k
Molmo2-SynMultiImageQA-chart	0.7	100k	330k	33k
ChartQA	0.6	18k	28k	28k
Molmo2-SynMultiImageQA-doc	0.6	88k	270k	28k
ST-VQA	0.6	18k	25k	25k
InfographicVQA	0.6	4.4k	24k	24k
TabWMP	0.6	23k	23k	23k
PlotQA	0.5	160k	20m	160k
AI2D	0.5	6.2k	15k	15k
Molmo2-SynMultiImageQA-diagram	0.5	45k	150k	15k
Molmo2-SynMultiImageQA-table	0.4	47k	140k	14k
CoSyn-music	0.4	12k	82k	12k
DVQA	0.4	200k	2.3m	200k
FigureQA	0.4	100k	1.3m	100k
OK-VQA	0.4	9k	9k	9k
CoSyn-chemical	0.4	8.9k	55k	8.9k
Spot-the-Difference	0.3	15k	14k	7.5k
ScienceQA	0.3	6.2k	6.2k	6.2k
Molmo2-SynMultiImageQA-music	0.3	12k	46k	4.7k
Molmo2-SynMultiImageQA-chemical	0.2	8k	23k	2.4k
IMAGE POINTING	9.1	510k	5.5m	1.1m
PixMo-Points	4.6	220k	4.6m	530k
Molmo2-MultiImagePoint	2.0	180k	470k	470k
PixMo-Count	1.2	37k	74k	74k
CoSyn-point	1.2	68k	320k	68k
CAPTIONS/LONG QA	13.6	1.2m	1.6m	1.2m
Molmo2-Cap	3.4	100k	280k	100k
PixMo-CapQa	3.1	190k	270k	190k
PixMo-Cap	2.3	710k	710k	710k
PixMo-AskModelAnything	1.9	71k	160k	71k
Molmo2-MultiImageQA	1.5	98k	73k	45k
Molmo2-AskModelAnything	1.5	43k	130k	43k
NLP	9.1	0	980k	980k
Tulu	9.1	0	980k	980k
VIDEO POINTING	13.6	260k	500k	370k
Molmo2-VideoPoint	10.9	250k	450k	330k
AcademicVideoPoint-MeViS	1.2	1.6k	20k	20k
AcademicVideoPoint-ReVOS	0.7	3.4k	11k	11k
AcademicVideoPoint-LV-VIS	0.7	3.1k	11k	11k
AcademicVideoPoint-OVIS	0.05	600	880	880
AcademicVideoPoint-BURST	0.04	310	680	680
AcademicVideoPoint-Ref-DAVIS17	0.03	58	450	450
name	rate	visual	anno.	ex.
VIDEO QA	18.2	2.3m	4.7m	2.4m
Molmo2-CapQA	1.6	190k	950k	190k
Molmo2-SubtitleQA	1.2	100k	470k	100k
Video Localized Narratives	1.1	53k	180k	56k
TGIF	0.9	63k	210k	63k
TVQA	0.9	120k	120k	120k
Paxion	0.9	440k	440k	440k
Moments In Time	0.9	710k	710k	710k
Kinetics	0.9	420k	420k	420k
LLaVA Academic	0.9	11k	62k	31k
Ego4D	0.9	53k	53k	53k
EPIC KITCHENS	0.7	37k	37k	37k
COIN	0.7	7.8k	30k	30k
How2QA	0.6	25k	35k	25k
ActivityNet	0.5	12k	46k	21k
FunQA	0.5	3.1k	200k	21k
CLEVRER	0.5	10k	130k	20k
STAR	0.5	3k	91k	19k
YouCook2	0.4	1.2k	18k	10k
SUTD-TrafficQA	0.4	10k	56k	10k
CinePile	0.4	9.2k	300k	9.2k
Charades STA	0.4	5.3k	12k	9.2k
QVHighlights	0.3	6.8k	7k	7k
MotionBench	0.3	5k	5k	5k
Countix	0.2	3.9k	4.4k	4.4k
NEX-T-QA	0.2	3.9k	34k	3.9k
Sports-QA	0.2	3.6k	56k	3.6k
IntentQA	0.2	3.2k	24k	3.2k
NewsVideoQA	0.2	2.9k	8.4k	2.9k
RoadTextVQA	0.2	2.6k	8.4k	2.6k
PerceptionTest	0.2	2k	7.4k	2k
CamaeraBench	0.1	1.4k	1.4k	1.4k
Social IQ 2	0.1	0.79k	5k	0.79k
VIDEO TRACKING	13.6	130k	800k	800k
Molmo2-VideoTrack	4.6	8k	220k	220k
AcademicVideoTrack-MeViS	2.0	1.7k	150k	150k
AcademicVideoTrack-ViCaS	1.2	15k	130k	130k
AcademicVideoTrack-ReVOS	1.2	0.7k	82k	82k
AcademicVideoTrack-TrackingNet	1.1	29k	29k	29k
AcademicVideoTrack-Ref-Youtube-VOS	0.9	3.5k	26k	26k
AcademicVideoTrack-VastTrack	0.8	46k	93k	93k
AcademicVideoTrack-LV-VIS	0.8	3.1k	38k	38k
AcademicVideoTrack-GOT-10k	0.4	9.2k	18k	18k
AcademicVideoTrack-WebUAV	0.2	3.2k	6.3k	6.3k
AcademicVideoTrack-BURST	0.07	0.28k	2.9k	2.9k
AcademicVideoTrack-LaSOT	0.06	1.1k	2.2k	2.2k
AcademicVideoTrack-TNL2K	0.06	0.88k	1.8k	1.8k
AcademicVideoTrack-WebUOT	0.05	0.84k	1.5k	1.5k
AcademicVideoTrack-LVOS V2	0.05	0.42k	1.2k	1.2k
AcademicVideoTrack-lasot	0.03	0.22k	0.45k	0.45k
AcademicVideoTrack-UW-COT220	0.03	0.21k	0.4k	0.4k
AcademicVideoTrack-LVOS V1	0.02	0.12k	0.3k	0.3k
AcademicVideoTrack-TNL2K	0.02	0.15k	0.29k	0.29k
AcademicVideoTrack-Ref-DAVIS17	0.02	0.06k	1.1k	1.1k
AcademicVideoTrack-YouTube-VIS	0.02	1.2k	1.4k	1.4k
AcademicVideoTrack-MoCA-Video	0.01	0.13k	0.4k	0.4k

Table 9. **Full dataset list.** Columns show sampling rates, the number of videos or images, the number of annotations, and the number of training examples built after formatting the data into message trees.

Prompting. We use the human-written questions with long-form answers from PixMo-AskModelAnything, PixMo-CapQA, and Molmo2-AskModelAnything directly. For captioning, all multiple-choice questions, and our various grounding tasks, we use prompt templates to generate a variety of ways to prompt the model for the target output. The remaining short-answer or captioning academic datasets typically have answer styles that are poorly suited for user-facing behaviors, either because they are too terse or have

other idiosyncratic quirks due to how the data was collected. For these datasets, we prompt the model with style tags (*e.g.* "short_video_answer:") so that Molmo2 adopts those answer styles only if specifically prompted to do so.

Hyperparameters. Hyperparameters for AdamW [67] are in Table 8. Following Molmo [73], during pre-training, we use a high learning rate for the connector and a long warmup for the ViT and LLM so that the first steps of training mostly train the connector. We use a cosine learning rate that de-

Model	Pre-train			SFT			Long-Context		
	GPU_s	time	GPU hr.	GPU_s	time	GPU hr.	GPU_s	time	GPU hr.
4B	32	15.2	490	128	58.8	7.5k	128	25.3	3.2k
7B	64	11.3	720	128	59.3	7.6k	128	25.7	3.3k
8B	64	12.1	780	128	63.0	8.1k	128	26.0	3.3k

Table 10. **Training times.** Training was done with Nvidia H100 GPUs.

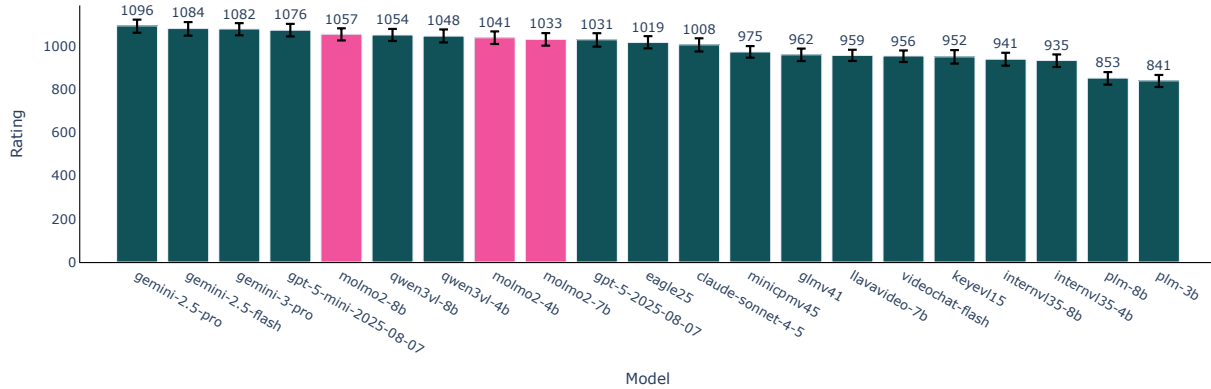


Figure 5. Elo ratings with confidence intervals

Model	Overall		Captioning		QA	
	Score	Rank	Score	Rank	Score	Rank
API call only						
GPT-5 [114]	1031	10	1136	2	1019	11
GPT-5 mini [114]	1076	4	1086	5	1075	4
Gemini 3 Pro [45]	1082	3	1126	3	1076	3
Gemini 2.5 Pro [25]	1096	1	1148	1	1090	1
Gemini 2.5 Flash [25]	1084	2	1109	4	1082	2
Claude Sonnet 4.5 [5]	1008	12	1009	10	1008	12
Open weights only						
InternVL3.5-4B [149]	935	19	817	19	947	19
InternVL3.5-8B [149]	941	18	855	18	951	17
Qwen3-VL-4B [10]	1048	7	1052	7	1049	6
Qwen3-VL-8B [10]	1054	6	1105	5	1048	7
Keye-VL-1.5-8B [170]	952	17	957	15	950	18
GLM-4.1V-9B [137]	962	14	1013	9	956	15
MiniCPM-V-4.5-8B [176]	975	13	978	14	975	13
Eagle2.5-8B [17]	1019	11	987	13	1022	10
Open models						
PLM-3B [22]	841	21	880	17	836	21
PLM-8B [22]	853	20	761	21	863	20
LLaVA-Video-7B [184]	959	15	981	14	955	16
VideoChat-Flash-7B [79]	956	16	932	16	959	14
Molmo2 family: Open weights, Open data, Open code						
Molmo2-4B	1041	8	1004	11	1045	8
Molmo2-8B	1057	5	1049	8	1059	5
Molmo2-O-7B	1033	9	1019	9	1034	9

Table 11. **Human evaluation results.** Scores updated using bootstrap Elo medians from overall, captioning, and QA evaluations.

generating extremely long output (see Section I).

Counting and Pointing. For the video counting evaluation, we preprocess 2 fps videos and clip them to random intervals under 63 seconds. In addition to exact accuracy and close accuracy, we also track models’ counting accuracy by

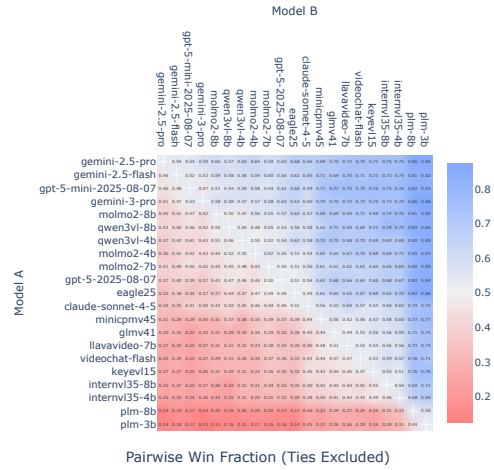


Figure 6. Pairwise win rates across all model pairs in human preference evaluation.

query category and by object count (Table 12). We find that Molmo2-8B performs the best on Action/Event and Object counting, just behind Gemini 2.5 Pro and GPT-5. Molmo2-8B also performs competitively on Animal counting, trailing slightly behind GPT-5 and Qwen3-VL-8B. Importantly, Molmo2 achieves similar accuracies to Qwen3-VL on low-count (0-10) queries while performing substantially better on high-count cases (10-60). Notably, Qwen3-VL obtains

0% accuracy in the 25-60 range, whereas Molmo2 exceeds 10%, placing it just behind Gemini 2.5 Pro.

For the video pointing evaluation, we use 2 fps videos with a maximum of 384 frames along with ground truth points and masks at 2 fps. For metrics, we compute recall, precision, F1, and valid accuracy (*i.e.*, the percentage of predictions that are parsed correctly), reporting all metrics in Table 3. In contrast to the counting task, Qwen3-VL struggles to perform meaningful pointing: Qwen3-VL-8B achieves only 1.5 F1, indicating that it rarely produces correct points. Even the strongest proprietary model shows a significant gap relative to ours: Gemini 3 and 2.5 Pro reach 20.0 and 13.0 F1, whereas Molmo2-4B and Molmo2-8B achieve 39.9 and 38.4 F1, respectively. This highlights a substantial performance advantage of Molmo2 on fine-grained spatio-temporal localization.

To evaluate the performance of baseline models on counting and pointing, we adopt the following setups. For both counting and pointing, we feed the entire videos to Gemini and Qwen3-VL models and use their default setup for video preprocessing. For GPT and Claude models, we feed the video frames to them using the same max frames and fps in our models' video preprocessing. As for the prompt, we use a general counting prompt followed by a brief format instruction across all models: "How many {label} are there? Output the integer number of the count only. The answer is:". For pointing, we first try prompting baseline models with our pointing format, but find that they struggle to follow the instruction and produce sensible outputs. We then carefully review various cookbooks for the baseline models where available, and design prompts with the HH:MM:SS format for timestamps and the bounding box format (which we then calculate the center's coordinates and use those for evaluation). We present the prompts used in video pointing evaluation for models with video and image inputs in prompt 1 and 2, respectively.

```
You are a video-analysis assistant that
  points to unique target objects in the
  video at 2FPS.
```

Goal:

```
Point to the timestamp and spatial
  coordinates of target objects, actions,
  or events in the input video.
- timestamp (as a string in 'HH:MM:SS'
  format, where the second can be to the
  closest 0.5 seconds e.g. '00:01:23.5')
- x_min, y_min, x_max, y_max (integer
  coordinates normalized to a 0-1000 scale
  )
```

Rules (strict):

```
- For actions/events spanning some time,
  pick the most representative / clear
```

```
timestamp.
```

- Each instance should be a separate spatial-temporal point in "results".
- Do NOT point to the same object more than once.
- Return only valid JSON, without markdown code blocks, explanations, or extra text .

Output format (strict JSON):

```
{
  "results": [
    {
      "timestamp": <str>, 'HH:MM:SS' format
      "x_min": <int>,
      "y_min": <int>,
      "x_max": <int>,
      "y_max": <int>
    },
    ...
  ]
}
```

Target: {label}

Listing 1. Video pointing prompt for baselines with video inputs

```
You are a video-analysis assistant that
  points to unique target objects in the
  video, represented as a sequence of
  image frames at 2FPS.
```

Goal:

```
Point to the timestamp and spatial
  coordinates of target objects, actions,
  or events in the input video frames at
  0.5 second intervals.
- timestamp (as a string in 'HH:MM:SS'
  format, where the second can be to the
  closest 0.5 seconds e.g. '00:01:23.5')
- x_min, y_min, x_max, y_max (integer
  coordinates normalized to a 0-1000 scale
  )
```

Rules (strict):

- For actions/events spanning some time, pick the most representative / clear timestamp.
- Each instance should be a separate spatial-temporal point in "results".
- Do NOT point to the same object more than once.
- Return only valid JSON, without markdown code blocks, explanations, or extra text .

Output format (strict JSON):

```
{
  "results": [
```

Model	Query Category				Object Count							
	Action/Event	Animal	Object	Avg.	0–5	5–10	10–15	15–20	20–25	25–60	Avg.	
<i>API call only</i>												
GPT-5 [114]	46.6	75.5	29.8	50.6	64.4	34.1	31.3	16.2	11.1	10.5	27.9	
GPT-5 mini [114]	36.2	63.3	25.1	41.5	55.7	28.2	25.0	10.8	6.3	10.5	22.8	
Gemini 3 Pro [45]	58.6	75.5	29.7	54.6	69.5	34.1	24.1	16.2	14.3	12.5	28.5	
Gemini 2.5 Pro [25]	53.4	63.3	30.0	48.9	61.5	31.3	31.5	15.7	17.5	13.0	28.4	
Gemini 2.5 Flash [25]	36.2	63.3	27.7	42.4	56.9	31.0	27.5	19.2	9.8	3.5	24.6	
Claude Sonnet 4.5 [5]	26.3	53.1	24.3	34.6	48.0	24.7	20.3	14.9	15.9	5.4	21.5	
<i>Open weights only</i>												
Qwen3-VL-4B [10]	39.7	59.2	19.5	39.4	56.9	17.6	21.3	2.7	3.2	0.0	16.9	
Qwen3-VL-8B [10]	43.1	75.5	22.5	47.0	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	51.7	59.2	29.1	46.7	58.0	31.8	30.0	24.3	9.5	12.3	27.7	
Molmo2-8B	50.0	69.4	29.6	49.7	64.4	32.9	26.3	25.7	7.9	7.0	27.4	
Molmo2-O-7B	50.0	63.3	27.5	46.9	60.9	32.9	27.5	16.2	6.3	8.8	25.4	

Table 12. **Molmo2-VideoCount** accuracy by query category (left) and by object count (right).

```

{
  "timestamp": <str>, 'HH:MM:SS' format
  "x_min": <int>,
  "y_min": <int>,
  "x_max": <int>,
  "y_max": <int>
},
...
]
}

Target: {label}

```

Listing 2. Video pointing prompt for baselines with image inputs

Tracking. We expand on the tracking results from Table 4 with comprehensive benchmark splits, additional baselines, and an identity-aware tracking metric (HOTA).

We report three complementary metrics. $\mathcal{J}\&\mathcal{F}$ is the standard segmentation quality metric, averaging the Jaccard index \mathcal{J} (mask IoU) and boundary F-score \mathcal{F} (contour alignment) over all objects and frames. **Point F1** captures frame-wise detection performance at 1 fps, where a predicted point is correct if it falls within the ground-truth mask. Since Point F1 is insensitive to identity swaps when the number of objects remains constant, we additionally report **HOTA** [97] ($\text{HOTA} = \sqrt{\text{DetA} \times \text{AssA}}$), which jointly scores detection accuracy (DetA) and association accuracy (AssA). We adapt HOTA from its original bounding-box formulation to point-based tracking by defining similarity as binary: a predicted point matches a ground-truth object if it falls within its segmentation mask. DetA measures whether points land in correct masks, while AssA measures whether consistent object IDs are maintained over time and penalizes identity switches. Since baseline models do not output stable track IDs, HOTA is only reported for Molmo2.

Segmentation metrics are computed at the original video frame rate, while point-based metrics are evaluated at 1 fps. Specialized open segmentation models output a single foreground mask per frame and are evaluated on segmentation quality only. For models that produce discrete points per object (e.g., VLMs), we additionally report point-based metrics. API models and generic VLMs cannot produce accurate point tracks (as shown in the video pointing task, Table 3), but their grounding improves substantially when prompted to output bounding boxes. Thus, for these models we predict bounding boxes at 1-second intervals, use them to prompt SAM 2 to generate segmentation masks, and take box centers as points. Molmo2, instead, predicts discrete point tracks with explicit IDs that are directly fed to SAM 2 to obtain segmentation masks.

Table 13 presents results across all academic benchmarks and their splits. Molmo2 substantially outperforms both API-based and open-source VLMs, suggesting existing VLMs are not well-suited for object tracking. Specialized open models that directly generate segmentation also fall behind, indicating difficulty in effectively grounding object semantics despite being specifically trained for tracking. The most directly comparable baseline is Video-Molmo [3], another video language model trained for point grounding in videos. While specialized models perform on par or outperform our model on Ref-Davis, which involves single objects with simple text queries, Molmo2 excels in more complex scenarios: multi-object tracking in MeViS [31] and reasoning-intensive tasks in ReasonVOS [166].

Table 14 reports per-domain results on our Molmo2-Track benchmark. Overall, Molmo2 outperforms other VLMs and even specialized open video models. API-based and open-source VLMs, including Molmo and Video-

Model	MeViS [31]			MeViS [31]			Ref-YT-VOS [127]			Ref-Davis [66]			ReasonVOS [11]		
	valid			valid-u			valid			valid			test		
	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA
API call only															
GPT-5 [114]	23.4	26.5	17.3	14.0	30.9	21.0	18.4	25.2	17.0	11.6	24.7	13.6	10.7		
GPT-5 mini [114]	15.7	15.4	8.5	6.8	16.2	7.4	6.2	8.4	3.4	2.3	14.6	4.2	3.4		
Gemini 3 Pro [45]	42.5	51.1	42.3	36.0	55.0	49.1	45.5	66.6	60.8	55.7	52.6	48.5	42.1		
Gemini 2.5 Pro [25]	40.7	52.8	41.2	35.0	45.1	44.5	40.5	45.6	62.7	56.6	44.0	50.2	42.4		
Gemini 2.5 Flash [25]	27.6	31.8	24.0	19.9	36.0	32.8	30.0	31.6	36.7	30.0	26.5	25.8	21.0		
Open weights only															
Qwen3-VL-4B [169]	29.7	30.6	23.3	18.7	32.1	29.0	26.5	44.4	33.1	26.9	26.5	17.0	13.5		
Qwen3-VL-8B [169]	35.1	34.4	30.1	23.8	48.3	42.1	37.6	41.0	41.6	33.2	24.9	22.3	17.5		
Specialized open models															
VideoLISA [11]	44.4	53.2	–	–	63.7	–	–	68.8	–	–	47.5	–	–		
VideoGLaMM [121]	45.2	50.6	–	–	66.8	–	–	69.5	–	–	33.9	–	–		
Sa2VA-8B [177]	46.9	57.0	–	–	70.7	–	–	<u>75.2</u>	–	–	55.5	–	–		
Sa2VA-Qwen3-VL-4B [177]	36.7	57.1	–	–	68.1	–	–	76.0	–	–	50.0	–	–		
Molmo [29] + SAM 2 [122]	46.9	51.5	53.8	–	64.6	71.1	–	65.2	74.5	–	45.7	50.3	–		
VideoMolmo-7B [3]	53.9	57.0	59.4	–	67.3	73.7	–	72.5	75.4	–	51.1	50.3	–		
Molmo2 family: Open weights, Open data (no distillation), Open code															
Molmo2-4B	63.3	<u>70.0</u>	<u>75.5</u>	<u>72.4</u>	<u>70.2</u>	80.4	78.8	73.5	83.1	81.1	61.9	66.5	64.0		
Molmo2-8B	<u>62.3</u>	70.8	<u>75.9</u>	72.6	<u>70.2</u>	<u>78.7</u>	<u>77.3</u>	72.7	<u>81.3</u>	<u>78.7</u>	65.8	70.8	68.6		
Molmo2-O-7B	58.4	69.7	76.1	72.3	67.9	77.7	76.1	70.4	79.2	76.0	<u>62.6</u>	<u>67.5</u>	<u>65.1</u>		

Table 13. **Tracking Results on Academic Benchmark.** $\mathcal{J}\&\mathcal{F}$ is reported for specialized segmentation or points-to-segmentation models. F1 is the point accuracy measured for VLMs that can generate points per frame. HOTA [97] is the tracking accuracy that accounts for association accuracy for models that provide tracking IDs.

Model	Animals			Person			Sports			Dancers			Misc			Overall		
	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA
API call only																		
GPT-5 [114]	41.4	20.6	20.3	16.5	4.5	4.2	14.4	2.0	2.5	33.8	11.7	11.5	14.6	2.2	1.6	23.5	7.5	7.5
GPT-5 mini [114]	21.7	7.8	8.0	8.6	1.6	1.5	10.7	0.6	0.8	15.6	2.1	2.0	13.5	0.6	0.4	12.7	2.1	2.1
Gemini 3 Pro [25]	70.4	62.3	60.0	44.5	30.7	29.2	23.4	10.3	8.8	55.6	44.3	37.8	35.3	18.3	14.4	44.6	32.2	29.1
Gemini 2.5 Pro [25]	69.3	56.8	53.2	50.0	33.6	31.9	29.7	10.8	8.9	55.9	39.4	32.2	34.7	17.6	18.3	47.9	31.2	27.8
Gemini 2.5 Flash [25]	58.0	46.6	44.4	38.9	21.4	20.1	13.2	6.2	5.5	48.0	29.0	25.1	21.9	5.7	4.6	36.2	21.8	19.8
Open weights only																		
Qwen3-VL-4B [169]	57.2	11.5	12.3	35.1	12.0	11.2	3.8	0.4	0.4	34.6	6.9	5.7	17.5	6.2	4.2	28.5	7.2	6.7
Qwen3-VL-8B [169]	63.8	52.3	50.2	35.4	20.3	18.9	5.2	1.7	1.4	31.3	19.0	16.7	16.3	6.2	4.2	28.7	18.0	16.5
Specialized open video models																		
VideoLISA [11]	67.8	–	–	35.8	–	–	32.9	–	–	53.6	–	–	25.8	–	–	43.3	–	–
VideoGLaMM [121]	63.9	–	–	26.2	–	–	34.3	–	–	46.0	–	–	22.3	–	–	37.9	–	–
Sa2VA-8B [177]	74.3	–	–	45.5	–	–	30.7	–	–	53.3	–	–	49.1	–	–	46.9	–	–
Sa2VA-Qwen3-VL-4B [177]	73.3	–	–	48.6	–	–	31.6	–	–	50.1	–	–	31.4	–	–	46.7	–	–
SAM 3 [16]	41.1	–	–	35.2	–	–	43.3	–	–	29.2	–	–	36.8	–	–	36.3	–	–
Molmo [29] + SAM 2 [122]	71.8	76.0	–	52.7	7.0	–	52.8	2.6	–	51.7	7.55	–	40.9	<u>37.5</u>	–	54.2	14.0	–
VideoMolmo-7B [3]	68.4	69.5	–	<u>51.1</u>	6.3	–	43.2	2.1	–	53.8	7.2	–	39.9	30.8	–	51.3	12.7	–
Molmo2 family: Open weights, Open data (no distillation), Open code																		
Molmo2-4B	81.0	83.0	83.7	43.7	48.3	<u>47.7</u>	<u>59.7</u>	<u>53.1</u>	<u>54.3</u>	60.4	64.4	64.4	43.1	35.1	<u>31.3</u>	56.7	57.5	57.6
Molmo2-8B	<u>80.1</u>	<u>82.0</u>	<u>83.0</u>	43.1	<u>47.9</u>	48.0	59.8	53.3	54.8	<u>59.9</u>	<u>63.9</u>	<u>63.5</u>	41.6	31.5	29.7	<u>56.2</u>	<u>57.1</u>	<u>57.5</u>
Molmo2-O-7B	<u>80.1</u>	81.9	82.8	41.5	45.5	45.4	54.1	47.6	48.6	57.7	61.0	60.3	<u>45.0</u>	37.6	34.7	53.7	54.2	54.2

Table 14. **Tracking results on Molmo2-Track by video domain.** Overall is the accuracy across all samples.

Molmo [3], struggle to maintain consistent object identity throughout videos, as reflected in their low F1 and HOTA scores. Interestingly, some baselines achieve high $\mathcal{J}\&\mathcal{F}$ in cluttered scenes (Pedestrians, Sports, Dancers) by generating large, coarse masks covering entire groups rather than precisely localizing individuals—this inflates region overlap while failing to track specific objects, as shown by substantially lower F1 and HOTA. This highlights the importance of our point-based F1 and identity-aware HOTA metrics, which more directly measure a model’s ability to precisely ground and track the correct objects.

D. Additional results

In this section, we present several additional evaluations.

D.1. Image results

We present image and multi-image benchmark results in Table 15. We follow the evaluation protocol from Molmo [29] and report the same 11-benchmark average for single-image benchmarks. As with videos, we collect results for all models by testing them ourselves if needed.

Generally, Molmo2 robustly outperforms previous open-data models. Molmo2 is a bit behind the best open-weight model on OCR-heavy benchmarks (such as DocVQA or InfoQA) but performs well on general QA tasks, including state-of-the-art performance on VQA v2.0 and RealWorldQA (RWQA). Counting is also a strength, most notably on the challenging PixMo-Count test set. However, Molmo2 is behind on open-weight reasoning benchmarks (MathVista, MMMU), possibly due to the lack of multi-modal reasoning training data. On multi-image tasks, Molmo2 performs competitively with most open-weight models, with the exception of GLM-4.1V-9B, which is notably ahead of all other models.

We evaluate image pointing on Point-Bench [20], results are in Table 16. Molmo2 surpasses all other models on the Point-Bench leaderboard* and the recent dedicated pointing model Poivre [171]. We attribute the gain on pointing compared to Molmo to the improved vision encoder, pointing pre-training, and token-weighting.

D.2. Additional model ablations

Model ablations. For additional model ablations (Table 17), we increase the video pool size from 3x3 to 4x4, which slightly lowers QA performance and causes a notable drop in captioning quality. We believe that most of the video benchmarks are relatively high-level and do not require understanding small details, which is why there is a relatively small drop on these tasks when increasing the pooling size.

* An older version of this report include higher scores that were the result of an evaluation bug.

* As of 12/15/25

This illustrates the importance of tracking the captioning metric in addition to the other benchmarks, which requires a much more fine-grained understanding of the video. Similarly, removing time tokens diminishes both metrics, indicating that temporal information and appropriate pooling are important for maintaining strong overall model performance.

Long context SFT. We compare the Molmo2-4B performance before and after long-context post-training in Table 18. We find that long-context post-training significantly improves model performance on long video QA benchmarks, while the video caption performance drops and performance on short video QA benchmarks and image QA benchmarks do not significantly change.

D.3. NLP Benchmarks

We evaluate Molmo2 on selective NLP benchmarks covering general knowledge QA, math, reasoning, and coding tasks and report their results compared to the base language models Qwen3 in Table 19. We run evaluations for all models following OLMo 3’s evaluation protocol, except for OLMo3-7B-Instruct’s MMLU and MBPP+ numbers, which we take directly from OLMo3’s model card. We find that Molmo2 achieves comparable numbers on the general knowledge QA and math benchmarks, MMLU and GSM8K, but suffers from some drops in coding on the MBPP+ coding benchmark [8]. Interestingly, both Molmo2-4B and Molmo2-8B perform slightly better than their respective base language models in the ARC Challenge multiple-choice evaluation.

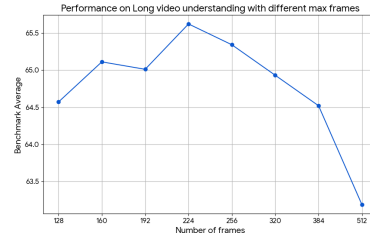


Figure 7. Long video benchmark results with different max frames, the average of our six long video benchmarks.

E. Test time scaling with 128-frame model

In this section, we consider whether it is possible to scale the number of frames past 128 during inference without long-context training. We also test an approach using SlowFast [164] to provide the model with a mix of high and low-resolution frames during inference, or during both training and inference.

Increasing max frames. At test time, we scale the maximum number of frames for better long video understanding. We evaluate Molmo2-8B after the SFT stage, but before

Model	A12D test [65]	ChartQA test [102]	DocVQA test [104]	InfoQA test [105]	TextVQA val [130]	VQA v2.0 val [47]	RWQA [158]	MMMU val [179]	MathVista testmini [96]	CountBench [13]	PixMoCount test [29]	MuirBench [142]	MMU [106]	Blink val [41]	Img QA avg.	Multiling QA avg.	Average
API call only																	
GPT-5 [114]	97.1	89.6	88.9	83.0	78.7	79.7	80.8	81.8	82.7	90.8	67.2	78.6	71.0	66.5	83.7	72.1	81.2
GPT-5 mini [114]	95.8	88.2	86.7	82.2	79.1	72.1	77.0	78.7	79.2	87.1	74.4	71.4	64.5	68.7	81.9	68.2	78.9
Gemini 3 Pro [45]	98.7	93.7	87.1	86.9	74.1	74.1	73.6	85.2	89.1	96.1	90.0	86.1	72.1	87.4	86.2	81.9	85.3
Gemini 2.5 Pro [25]	94.3	82.7	91.5	82.0	70.3	67.1	77.4	79.6	84.6	90.8	73.8	74.5	68.9	73.7	81.3	72.4	79.4
Gemini 2.5 Flash [25]	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 [5]	91.5	88.1	91.7	65.9	67.2	77.0	61.1	77.8	73.1	87.3	58.3	59.6	54.1	64.8	76.3	59.5	72.7
Open weights only																	
InternVL3.5-4B [149]	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 [149]	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 [169]	84.1	84.6	<u>95.3</u>	80.3	81.0	81.7	70.9	67.4	73.7	85.5	58.0	63.8	43.2	<u>65.8</u>	78.4	57.6	73.9
Qwen3-VL-8B [169]	85.7	<u>89.6</u>	96.1	83.1	82.8	82.3	71.5	69.6	77.2	90.4	65.0	<u>64.4</u>	35.3	69.1	<u>81.2</u>	56.3	<u>75.9</u>
Keye-VL-1.5-8B [170]	89.5	94.1	93.4	74.9	81.5	79.3	73.5	<u>71.4</u>	81.2	81.6	57.4	51.2	50.3	54.9	79.8	52.1	73.9
GLM-4.1V-9B [137]	87.9	70.0	93.3	80.3	79.6	68.3	70.7	68.0	<u>80.7</u>	88.0	60.7	74.7	62.4	65.1	77.0	67.4	75.0
MiniCPM-V-4.5-8B [176]	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 [17]	84.5	87.5	94.1	<u>80.4</u>	83.7	82.4	<u>76.7</u>	55.8	67.8	90.2	90.2	61.8	48.4	45.8	<u>81.2</u>	52.0	75.0
Open models																	
PLM-3B [22]	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 [22]	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
Molmo1 family: Open weights, Open data (no distillation), Open code																	
MolmoE-1B [29]	86.4	78.0	77.7	53.9	78.8	83.9	60.4	34.9	34.0	87.2	79.6	-	-	-	68.6	-	-
Molmo-7B-O [29]	90.7	80.4	90.8	70.0	80.4	85.3	67.5	39.3	44.5	89.0	83.3	-	-	-	74.6	-	-
Molmo-7B-D [29]	93.2	84.1	92.2	72.6	81.7	85.6	70.7	45.3	51.6	88.5	84.8	-	-	-	77.3	-	-
Molmo-72B [29]	96.3	87.3	93.5	81.9	83.1	86.5	75.2	54.1	58.6	91.2	85.2	-	-	-	<u>81.2</u>	-	-
Molmo2 family: Open weights, Open data (no distillation), Open code																	
Molmo2-4B	95.6	86.1	87.8	78.6	85.0	<u>86.6</u>	75.4	50.9	56.7	<u>93.9</u>	88.1	60.5	<u>55.5</u>	57.5	80.4	<u>57.8</u>	75.6
Molmo2-8B	<u>95.8</u>	86.0	93.2	80.1	<u>85.7</u>	87.0	77.6	53.0	58.9	93.7	<u>88.5</u>	63.7	54.2	51.3	81.7	56.4	76.3
Molmo2-O-7B	93.7	84.9	90.4	77.9	84.7	<u>86.6</u>	73.6	45.8	54.2	95.1	88.9	58.4	51.7	50.5	79.7	53.5	74.1

Table 15. **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**. The Molmo1 models do not support multi-image input, so those evaluations are left blank.

long-context training, with 160, 192, 224, 256, 320, and 512 max frames and report the average on the val sets of our six long video understanding benchmarks in Figure 7. Molmo2 has the best performance with 224 frames for long video benchmarks. For short video understanding benchmarks, the average is 69.8 for 128 frames and 69.7 for all other settings as shown in Table 20.

Keeping Vision tokens fixed. However, increasing the maximum number of frames also increases the number of vision tokens fed into the model, which raises compute cost and may not be feasible on GPUs with limited memory. With the default setting of max 128 frames, the maximum number of vision tokens is $83 * 128 \sim 10.6k$. We therefore evaluate alternative test-time strategies that keep the num-

ber of max vision tokens close to $10.6k$. Specifically, we evaluate different pooling strategies in the vision-language connector - 4×4 pooling with 216 frames and 5×5 pooling with 332 frames. The 5×5 pooling setting improves long video understanding by accessing more frames; however, both settings regress on short video understanding (Table 20).

SlowFast encoding. Since we find that our model can generalize to different pooling sizes at test time, we further explore a SlowFast video strategy [164]. We build on the interleaved SlowFast variant used in [129, 165, 170], which dynamically allocates computational resources across frames by varying their spatial pooling in the Molmo2 connector, with each frame represented exactly once – either in the

Model	Affordance	Spatial	Reasoning	Steerability	Counting	Average
Human	92.3	83.6	87.8	86.3	95.6	89.1
<i>API call only</i>						
Gemini-Robotics-ER-1.5 [1]	69.7	69.7	60.1	67.5	<u>68.5</u>	67.1
Gemini-2.5-Pro [25]	72.7	70.3	71.0	41.0	59.2	62.8
<i>Open weights only</i>						
Poivre-7B [171]	-	-	-	-	-	67.5
Qwen2.5-VL-32B-Instruct [168]	76.8	60.0	54.4	<u>46.5</u>	57.1	59.0
Qwen2.5-VL-72B-Instruct [168]	76.8	60.0	54.4	<u>46.5</u>	57.1	59.0
Qwen3VL [169]	81.3	65.6	60.6	<u>23.5</u>	61.2	58.5
Qwen3-VL-235B-A22B-Instruct [169]	-	-	-	-	-	58.3
<i>Open models</i>						
VisionReasoner-7B [92]	-	-	-	-	-	64.7
<i>Molmo1 family: Open weights, Open data (no distillation), Open code</i>						
Molmo-7B-D [29]	82.8	67.7	70.5	28.5	58.7	61.6
Molmo-72B [29]	87.9	70.3	69.4	37.0	54.6	63.8
Molmo-7B-O [29]	<u>84.9</u>	63.1	63.2	45.5	59.7	63.3
<i>Molmo2 family: Open weights, Open data (no distillation), Open code</i>						
Molmo2-4B	82.3	71.8	72.0	41.0	<u>71.4</u>	<u>67.7</u>
Molmo2-8B	84.8	<u>71.3</u>	<u>71.5</u>	44.5	<u>71.4</u>	68.7
Molmo2-O-7B	81.8	69.7	69.4	39.0	72.4	66.5

Table 16. **Point-Bench results**, baseline scores taken from the Point-Bench leaderboard. Qwen3-VL-235B-A22B-Instruct and VisionReasoner-7B scores were taken from their evaluation in Poivre [171], which did not include sub-category scores.

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 17. **Additional model ablation**. Increasing pool size and removing time tokens hurts the model performance.

Long-Context	Short Vid.	Long Vid.	Video Cap.	Image
Yes	69.4	67.4	39.9	80.6
No	69.6	64.4	42.3	80.5

Table 18. **Long-context SFT ablation**. Columns show the average of our 12 video benchmarks divided by short/long video benchmarks, using validation sets for EgoSchema, Perception-Text, and MLVU, video captioning F1, the average of the 11 image benchmarks using validation sets for InfoQA, DocQA, ChartQA, VQA v2, and AI2D.

slow or the fast pathway. Frames are categorized as slow or fast based on a periodicity parameter p : every p -th frame is designated as a slow frame, while the remaining frames are fast frames. We refer to this approach as *Slowfast-periodic*. Note that $p = 1$ reduces to the default setting. Slow frames use the default pooling size of 3×3 , whereas fast frames use 9×9 pooling. We use four different periodicities $p \in \{1, 2, 3, 4\}$ with corresponding max frames $M \in \{128, 224, 300, 368\}$. The max frame M for each pe-

riodicity is chosen such that the maximum number of vision tokens input to the LLM is approximately $10.6k$. $10.6k$ is the maximum number of vision tokens used in the default setup of Molmo2. When processing a video with SlowFast encoding, after we sample F_t frames, p is selected to maximize the tokens in the slow pathway. For example, when $F_t \leq 128$, we use $p = 1$ and all the frames are in the slow pathway, or when $128 < F_t \leq 224$, we use $p = 2$ and every other frame is in the slow pathway. In practice, that leads to stepwise changes in selected p as the number of frames ranges from 1 to 368.

We explore two strategies to score the frames’ relevance for inclusion in the slow pathway. First, we embed both the query and all the frames using SigLIP 2 [139] and calculate per frame cosine similarity scores. Second, we calculate the average of the absolute similarity difference of the embedded frames with their neighboring frames. In either strategy, we use the per-frame score to select the relevant frames for the slow pathway. Our formulation when selecting F_s slow pathway frames from F_t sampled frames is to include both frames that globally have the highest scores and frames that have high scores in their local neighborhoods. To select locally high scoring frames, we first select $F_s/2$ frames by choosing the single highest scoring frame from temporally ordered groups of size $F_t \div F_s/2$. To select globally relevant frames, we select the remaining $F_s/2$ frames that have the highest scores from all the remaining

Model	MMLU [50]	GSM8K [24]	ARC-C [23]	MBPP+ [8]
Qwen3-4B [169]	72.2	87.8	83.3	59.5
Qwen3-8B [169]	76.8	89.8	88.3	62.2
OLMo3-7B-Instruct [112]	69.1	90.1	72.2	60.2
Molmo2-4B	72.2	86.6	89.3	56.2
Molmo2-8B	76.6	89.7	89.6	57.5
Molmo2-O-7B	64.1	89.0	79.9	55.7

Table 19. **Results on selective NLP benchmarks**, including MMLU for general knowledge QA, GSM8K for math, ARC-C for reasoning, and MBPP+ for coding tasks.

Model	VTok	Video-MME	Video-MME-Sub	LongVideoBench	MLVU	LVBench	VideoEvalPro	Short QA avg	Long QA avg
128 frames	10.6k	68.8	74.3	65.9	74.5	49.6	54.3	69.8	64.6
pool4, 216 frames	11k	68.9	75.0	64.3	75.7	48.9	54.9	68.8	64.6
pool5, 332 frames	10.6k	69.1	74.2	64.2	76.5	50.6	56.9	68.4	65.2
128 frames + SF-periodic	10.7k	68.1	74.5	64.2	74.5	48.3	53.5	69.6	63.9
128 frames + SF-diff	10.7k	68.4	74.1	64.7	75.7	48.7	54.8	69.6	64.4
128 frames + SF-query	10.7k	68.9	73.9	66.6	76.2	51.5	57.2	69.6	65.7
128 frames + SF-tr-0.1	10.7k	69.1	74.3	65.4	75.0	48.6	54.3	69.8	64.4
128 frames + SF-tr-0.1 + SF-query	10.7k	68.9	74.3	65.5	75.4	51.5	57.1	69.8	65.5
224 frames	18.6k	69.2	74.6	66.1	76.4	50.7	56.7	69.7	65.6

Table 20. **Molmo2-8B with test time scaling / SlowFast (SF) encoding** SF-query boosts long video understanding and matches using 224 frames while using $\sim 43\%$ fewer visual tokens. Training without SF and then using SF-query marginally beats training with SF-tr-0.1 on long video understanding tasks. All SlowFast models use a max of 368 frames. VTok denotes max vision tokens. SF-tr-0.1 denotes using SlowFast 10% of the time in training.

frames. Additionally, we don’t use score based selection and use Slowfast-periodic when the frames per second F_r is high. This follows the intuition that frame selection is useful when selecting amongst sparser frames for long videos with multiple scenes, but not for shorter videos that get densely sampled and tend to have only one scene. In practice, we fall back to Slowfast-periodic when $F_r \geq 2$.

With Slowfast-periodic, the model regresses on the long video understanding, contrary to the finding in [164]. Using the frame difference improves over using periodic sampling, but still lags behind the default setting. However, using the query to select frames for the slow pathway achieves the best performance. It provides a boost to long video understanding with minor regression in short video understanding. It closes the gap to the optimal setting of using 224 frames while having $\sim 43\%$ fewer visual tokens (Table 20).

Training with SlowFast. Due to the improvement on long video understanding tasks using SlowFast encoding in the training-free regime, we explore training with SlowFast. We report results for training in a combined single stage starting from the image captioner. We keep the max frames the same 128 and sample using the SlowFast setup with a probability P_{sf} while randomly sampling different $p \in 2, 4, 8$. We use the default sampling with a probability if $1 - P_{sf}$ and use $P_{sf} = 0.1$. When training with a SlowFast setup, we randomize the slow frames. Concretely, to select F_s frames from F_t sampled frames, 1 frame in ordered groups of size $F_t \div F_s$ is selected randomly. Even

though the max frames is not increased, the goal is to familiarize the video model with the SlowFast encoding similar to score-based Slow frame selection, but without increasing the training cost by requiring the use of more frames. At test time, we evaluate with and without the query based SlowFast setup described above. Surprisingly, training without SF and then using the query to select Slow frames beats training with SF 10% of the time as shown in Table 20. This suggests Molmo2 can frame using 9×9 pooling even though such frames were not seen during training.

F. Dataset details

In this section, we provide additional details about our data collection methodology.

F.1. Dataset statistics

Pointing. We report the statistics on the Molmo2-VideoPoint training and validation sets. Overall, the Molmo2-VideoPoint dataset contains diverse pointing queries across seven categories (Figure 8). There are more queries in Action/Event, Object, and Referring expression, as we expect these to be harder for the model to learn. We also see that the distribution is skewed towards low-count examples with 0 to 5 counts (Figure 8 and 10). We mitigate this bias by upsampling medium- and high-count examples during training, and plan to collect more high-count examples in the future. Similarly, the distribution of frames anno-

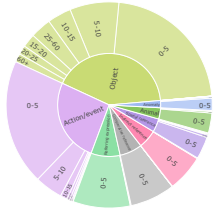


Figure 8. The distribution of categories and counts across pointing queries in **Molmo2-VideoPoint**.

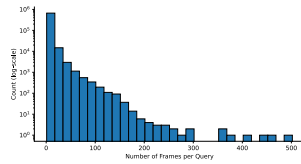


Figure 9. The distribution of annotated frame count per query in **Molmo2-VideoPoint**.

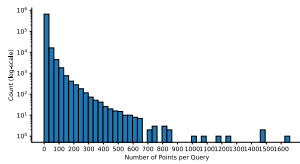


Figure 10. The distribution of annotated point count per query in **Molmo2-VideoPoint**.

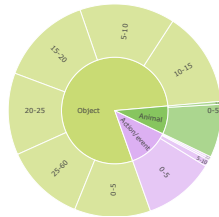


Figure 11. The distribution of categories and counts across queries in the **Molmo2-VideoCount** evaluation.

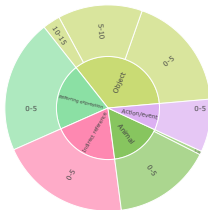


Figure 12. The distribution of categories and counts across queries in the **Molmo2-VideoPoint** evaluation.

tated per query is also heavily skewed to the left (Figure 9).

For the validation sets used in Molmo2-VideoCount and Molmo2-VideoPoint evaluations, we carefully build them by (1) collecting double annotations on some queries and selecting high-confidence examples where two different annotators provide the same answer; and (2) sampling queries across diverse categories and counts (Figure 11 and 12). For video counting, we mostly sample queries from the object category, as there are significantly more high-count examples in this category than in others (Figure 11). For video pointing evaluation, we intentionally pick queries in the more difficult categories – referring expression and indirect reference (Figure 12) – orthogonal to the ones in the counting evaluation, so that we have a comprehensive evaluation of our model’s counting and pointing capabilities.

Tracking. We report statistics on the videos and text queries in Molmo2-VideoTrack and the Molmo2-Track benchmark.

The two datasets have a total of 8k video clips, with 6.6k for training and 1.3k for evaluation. Both datasets provide segmentation masks, text queries, and metadata for each video. On average, there are 6.08 annotated objects per video, and the videos are up to 2 minutes long, with most being around 10-30 seconds. The distribution of video durations is shown in Figure 13.

Our dataset contains a total of 29k diverse text queries covering a wide variety of categories, bringing an average of 1.33 text queries per video. The distribution of categories is detailed in Figure 16 and Figure 17. Multi-object tracking is a primary focus in the tracking capabilities of Molmo2, so we strived to find text queries that describe many objects within a video. The dataset has an average of 3.31 objects described per text query, with many queries describing far more than that. The distribution is shown in Figure 14. Each text query is on average 8.21 words long, but there is a wide range. The exact distribution across all text queries is shown in Figure 15.

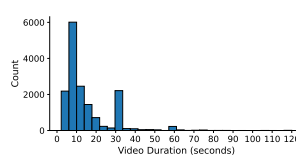


Figure 13. Distribution of video clip duration in **Molmo2-VideoTrack** and **Molmo2-Track**.

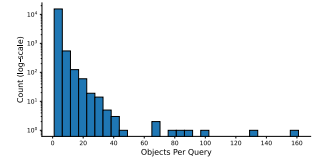


Figure 14. Distribution of objects described by text queries in **Molmo2-VideoTrack** and **Molmo2-Track**.

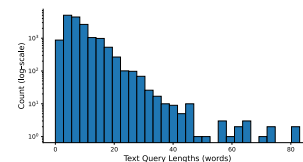


Figure 15. Distribution of text query lengths in **Molmo2-VideoTrack** and **Molmo2-Track**.

F.2. Data collection

Here, we detail how we collect videos and synthesize annotations for most of Molmo2 video datasets.

Video collection for Molmo2-Cap. We first source videos less than 3 minutes from multiple large-scale datasets [147, 153, 180, 184] and YouTube videos searched with keywords used in MetaCLIP [162] to form a pool of over 10M videos.

Then, we perform one step of filtering based on the informativeness of the video: we first discard the audio track and uniformly sample the video at 1 fps; Then the sampled frames are encoded using H.264; The total size of the resulting encoded stream (in bits) is divided by the product of

the video duration and spatial resolution ($\text{duration} \times W \times H$) to obtain a normalized video informativeness score. After collecting scores for all videos in the pool, we discard those whose score falls below (mean - 1 standard deviation), effectively removing videos with unusually low visual or temporal diversity.

After this filtering, we conduct a diversity-based sampling to obtain a final set of videos for human annotation: for each remaining video, we uniformly sample 5 frames and apply SAM 2 [122] to segment each frame, computing the average number of segments as a proxy for visual complexity. We further use Molmo to caption each sampled frame and follow MetaCLIP’s processing pipeline to extract a set of keywords that characterize its semantic content. To select a diverse subset, we perform a greedy sampling procedure that aims to maximize the entropy of both the segment-count distribution and the keyword distribution. At each step, we score all candidate videos using a two-stage ranking: (1) we compute a “what-if” entropy gain for the keyword distribution if the candidate were selected, and rank candidates accordingly; (2) we compute a density-based score that favors videos contributing to underrepresented segment-count regions. The final score is obtained by summing the two ranks, and we select the top-ranked candidate. For efficiency, we approximate this process by scanning the pool in chunks of 1,000 candidates at a time, rather than evaluating the entire pool at each iteration. This procedure yields a video subset that is both semantically diverse and visually varied, providing a strong foundation for high-quality human annotations. Finally, we set the sampling ratio to be 1% and obtained around 100k videos.

Video and synthetic annotation collection for Molmo2-CapQA, -SubtitleQA, -VideoPoint, and -AskModelAnything. We first source 500k videos with Creative Commons license from YT-Temporal [180] and YouTube keyword search. Then we use a video captioner trained on Molmo2-Cap to caption these videos. In particular, we segment each video into multiple scenes and caption each scene instead of the entire video to encourage detailed descriptions. Since model-generated captions can sometimes be low-quality, we apply a heuristic rule-based filter to remove captions with repetition patterns. The final set of videos and synthetic captions is used to curate Molmo2-CapQA, -SubtitleQA, and -VideoPoint datasets.

For Molmo2-CapQA and Molmo2-SubtitleQA, we prompt an LLM to generate both the question and the answer. For Molmo2-VideoPoint, we prompt an LLM to generate the queries and solicit human answers. For Molmo2-AskModelAnything, we elicit questions from human annotators and generate the corresponding answers using an LLM with human feedback.

F.3. Data annotation

Molmo2-Cap. To obtain clips for the first-stage captioning, we develop an algorithm to split a video into clips of variable lengths between 10 and 30 seconds based on their information density so that a more informative clip has a shorter duration. This algorithm minimizes the highest information density of a video clip across all clips. Overall, videos are split into 4-5 clips on average. We then deploy the video-description task to online crowdworkers (see Figure 21 for the task interface). For each full video, workers are first shown a sequence of shorter clips split by our algorithm from the original video with audio muted. At the top of the interface, we provide instructions to guide their descriptions. For each clip, workers verbally describe what is happening on the screen, and their speech is automatically converted to text via real-time transcription. They then edit the transcript to correct recognition errors before submitting it. After completing all clips, workers are asked to provide a comprehensive description of the full video (see Figure 22).

Molmo2-VideoPoint. For each video, we design several visual questions that require workers to answer using evidence from a single or several frames (see Figure 23 for the task interface). Crowdworkers first watch the full video clip without audio. For each question, they capture screenshots from the video at the moments when the relevant content is visible. On the screenshot, workers annotate points on object instances that satisfy the question, and we record both the video timestamp and the (x, y) coordinates of all points. Then they answer the corresponding questions in a required format. Workers could mark a question as Unanswerable (e.g., if the content is missing or ambiguous) or flag that they are unsure about their answer. This process is repeated for all questions associated with the video.

To collect annotations for anomaly identification queries in Molmo2-VideoPoint, we first need to construct a dataset of generative videos exhibiting visual defects. We begin by leveraging two publicly available datasets: the ViBe dataset [124] and the Broken Video Detection Dataset [85]. The Broken Video Detection Dataset provides high-quality, frame-level annotations of defective regions, allowing us to directly incorporate its pixel-accurate defect masks. From the ViBe dataset, we selectively retain only videos labeled as Vanishing Subject, Physical Incongruity, or Temporal Dystopia. These categories correspond to defects intrinsic to the generated video itself rather than issues arising from ill-posed or misleading prompts, ensuring our dataset focuses on model-induced visual failures. To complement these sources with realistic user prompts, we sample 2,000 human-written prompts from the VidProM dataset [148]. For each prompt, we generate videos using 10 T2V models and manually filter the outputs to retain only those containing clear and salient defects. This step introduces diversity

in both content and failure types and reflects real-world usage patterns of contemporary text-to-video systems. In total, our final training set for generative video anomaly pointing consists of 10k videos, covering a broad range of defective generations produced by around 25 T2V models.

Molmo2-VideoTrack. Directly reusing the Molmo2-VideoPoint annotation strategy for tracking is infeasible, as it would require point annotations on every sampled frame. One could use off-the-shelf tracking models, such as CoTracker [63] or SAM 2 [122], with point prompts; however, we found them to yield incomplete or unstable trajectories and are therefore not reliable sources for generating accurate training data for tracking. We thus resort to existing human-annotated tracks and focus on expanding coverage to video domains and object categories underrepresented in standard training datasets.

As our base pool, we use a set of videos in video object segmentation (VOS) datasets: SAM-V [122], VIPSeg [108], MOSE [32], and MOSEv2 [33], which are not as densely supported in existing academic video track datasets. We discard videos that are shorter than 3 seconds or that contain fewer than three object tracks. We additionally decontaminate videos in MOSE [32] with respect to the MeViS validation set [31]; we sample 8 frames per video, extract CLIP ViT-L/14 features [119], and remove any videos whose maximum pairwise frame similarity exceeds 0.95. We then extract points from segmentation masks by computing an alpha-weighted score that combines centroid distance and distance to mask boundaries, which keeps the points near the center while minimizing flickering.

We further extend our pool with datasets that provide video object tracks in the form of bounding boxes. These datasets span diverse domains and challenging multi-object scenarios with occlusion, including pedestrians, dancers, autonomous vehicles, animals, athletes, and UAV footage. Unlike in segmentation tracks, naively sampling a (center) point from a bounding box does not guarantee that the point lies on the object. Thus, we convert each bounding-box track into a segmentation task to obtain reliable point tracks. We prompt SAM 2 with the first available bounding box for an object to generate a mask tracklet and propagate this segmentation through the rest of the video. We re-prompt SAM 2 with a new box if the predicted mask has low IoU with the ground truth bounding box or if more than 20% of the mask is outside the bounding box. We filter out object tracks whose predicted segmentation masks have an average IoU below a threshold 0.5 across all frames. We then apply the same point-sampling procedure on these generated segmentation masks to obtain point tracks. This process is depicted in the first panel of Figure 18, and the annotator interface for this step is shown in Figure 19.

Text descriptions for these tracks are acquired with hu-

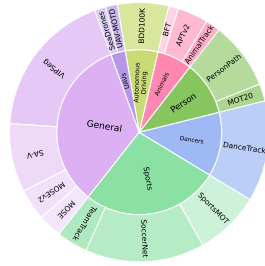


Figure 16. Molmo2-VideoTrack dataset



Figure 17. Molmo2-Track benchmark

man annotators. The annotation procedure is illustrated in the second panel of Figure 18, where human annotators are given a video and its list of object tracks and are asked to select one or more objects to write text queries for. The query should describe the selected objects only. The process is repeated N times per video, while ensuring that the set of selected objects is unique for each query. A separate validation round performs quality checks on the annotated text queries. After this filtering, we retain approximately 70% of the queries on average. This process yields both our training set and the Molmo2-Track benchmark. The annotator interface for validation is shown in Figure 20.

Table 21 summarizes the dataset statistics, and Figures 16 and 17 break down the distribution of queries and objects per semantic category for both training data and Molmo2-Track. The segmentation datasets provide general object tracking across diverse categories, while the bounding-box datasets contribute domain-specific tracking scenarios. Together, these complementary data sources yield a large-scale and diverse corpus for object tracking.

Academic-VideoTrack. We additionally construct an Academic-VideoTrack dataset by aggregating existing academic VOS datasets and bounding-box tracking datasets with referring expressions. Similar to the bounding-box processing for Molmo2-VideoTrack, we convert bounding-box tracks into segmentation mask tracklets by running them through the same pipeline (bounding-box-prompted SAM 2 followed by propagation and IoU-based filtering).

We also accommodate datasets with non-exhaustive labels, where objects mentioned in the text queries lack corresponding tracks despite appearing in the video. Since these missing objects cannot be used directly for general multi-object tracking, we repurpose them for the “single-point” task (Section 3), where the model receives a single point on the target object with the associated query and generates its track. This allows us to augment non-exhaustive tracking datasets to our training data and have the model be exposed to diverse, challenging tracking scenarios.

Table 9 shows the detailed composition of the Academic-

Data Source	Type (Ann.)	# Clips	# Tracks	# Queries	Avg # Obj/Q
VIPSeg	General (Segm)	675	2,150	5,466	2.65
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)	610	4109	4420	6.60
SportsMOT	Sports (Bbox)	396	2,150	2,420	4.48
BDD100K	Auto. Driving (Bbox)	450	1,810	1892	3.10
APTv2	Animals (Bbox)	401	1,051	1,132	2.68
AnimalTrack	Animals (Bbox)	52	413	542	3.59
BFT	Animals (Bbox)	30	214	364	2.38
UAV-MOTD	UAV (Bbox)	142	426	437	3.43
SeaDrones	UAV (Bbox)	79	368	408	2.25
MOT20	Person (Bbox)	147	603	643	2.68
PersonPath	Person (Bbox)	1,146	2,383	2,502	1.86
DanceTrack	Dancers (Bbox)	704	3,199	3,735	4.07
Total	All	6,624	25,437	29,704	3.38

(a) Statistics for the Molmo2-VideoTrack dataset.

Data Source	Type (Ann.)	# Clips	# Tracks	# Queries	Avg # Obj/Q
APTv2	Animals (Bbox)	188	331	332	1.57
PersonPath	Person (Bbox)	487	958	992	1.58
SportsMOT	Sports (Bbox)	323	825	838	4.03
DanceTrack	Dancers (Bbox)	360	885	905	3.11
SAM-V	Misc (Segm)	28	63	80	1.21
Total	All	1,386	3,062	3,147	2.66

(b) Statistics for the Molmo2-Track benchmark

Table 21. **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).

VideoTrack dataset used for training.

Molmo2-AskModelAnything. For each video, we first ask crowdworkers to watch the clip without audio and write questions in English that require non-trivial visual reasoning, such as temporal understanding, reading on-screen text details, or identifying fine-grained visual details. We discourage questions that were too vague, too easy or low-level, subjective with no clear ground-truth answer, dependent on unverifiable information such as names or identities, or simple counting questions, which we do not collect for this task. We then feed the full video caption together with the worker’s question into a backend language model, which produces an initial answer. Workers are then instructed to slightly edit the question to form a valid query and to carefully edit the model answer to form a final answer. Once they are satisfied, they submit the final Q&A pair, which we used as our annotation (see Figure 24 for the task interface).

Molmo2-MultiImagePoint. To improve multi-image

grounding, we construct a dataset of over 470k pointing and counting examples by applying soft clustering over images in PixMo-Points. Image sets are formed using a combination of single-token and sentence-level label embedding similarities, producing sets of 2–5 semantically related images (mean set size: 3.24). For each image set, we first normalize all human-provided labels via lowercasing, punctuation, and whitespace normalization, and synonym consolidation. We then use a large language model to resolve these normalized labels into a single canonical description that is semantically consistent across the set. This canonical label defines the shared entity or concept to be pointed to and counted across all images in the set. During training, we stochastically sample from the original (pre-canonicalized) human annotations rather than always using the canonical label, thereby preserving lexical diversity and improving robustness to annotation variability.

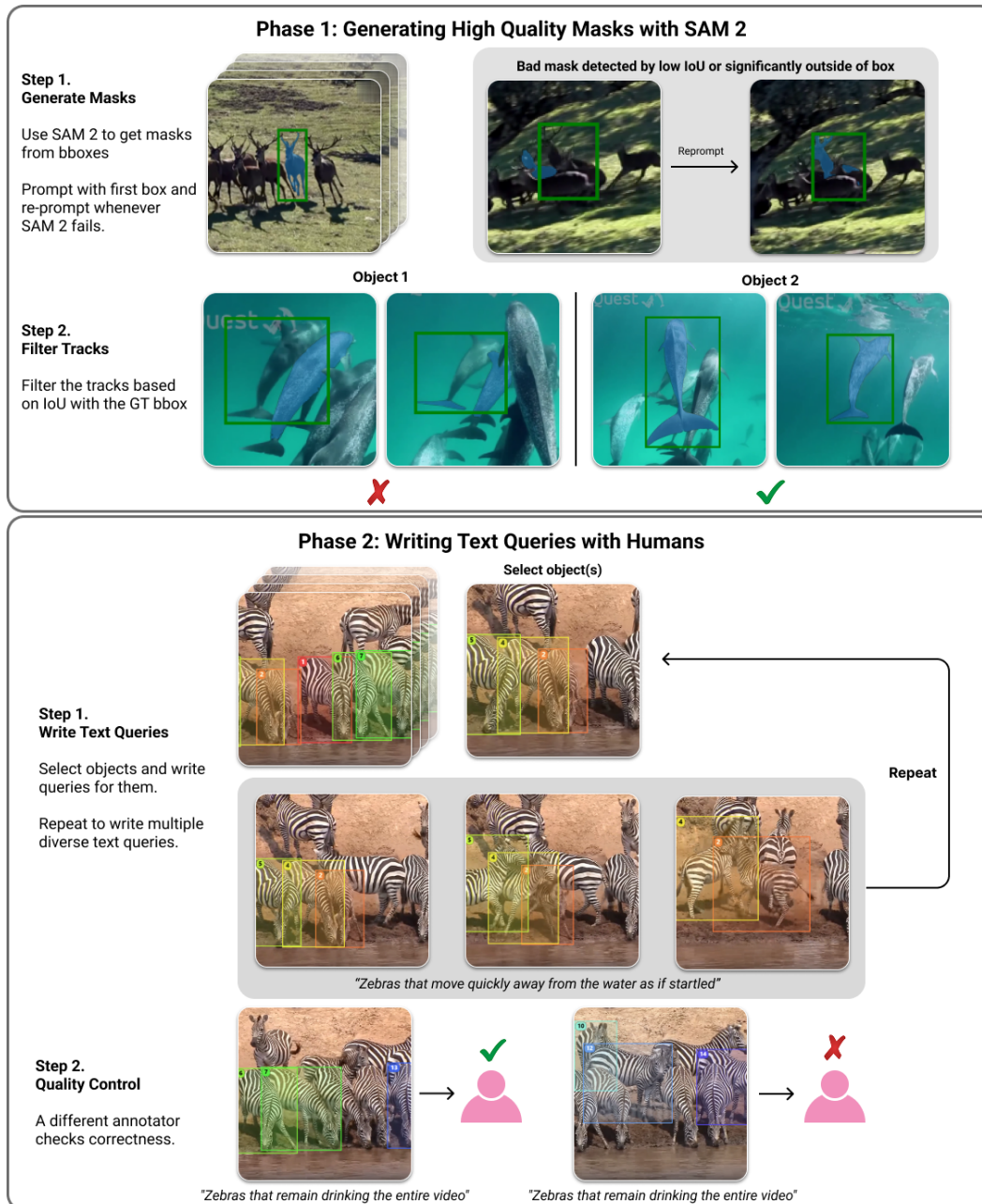


Figure 18. Overview of the annotation pipeline for **Molmo2-VideoTrack** and the **Molmo2-Track** benchmark.

G. Data examples

Here, we present qualitative examples from the Molmo2 datasets. For datasets, we show **randomly** selected examples. Prompts are in bold, and the target output text is below. Videos are shown using a small number of sampled frames. Examples can be found in:

- Molmo2-Cap: Figure 25
- Molmo2-AskModelAnything: Figure 26
- Molmo2-CapQA: Figure 27

- Molmo2-SubtitleQA: Figure 28
- Molmo2-VideoPoint: Figure 29
- Molmo2-VideoTrack: Figure 30
- Molmo2-MultiImageQA: Figure 31
- Molmo2-SynMultiImageQA: Figure 32
- Molmo2-MultiImagePoint: Figure 33

H. Related works

Multimodal LLMs. Multimodal LLM models have become popular in the last few years for image understanding and grounding tasks [29, 70, 136]. A common strategy for multimodal LLMs is to use CLIP-style image encoders and align image embeddings with the LLM input space via a connector module [29, 90]. Video LLMs also commonly extend the CLIP-style image encoding and use image embedders to individually embed each frame in a video [17, 22, 99]. Some have explored using pretrained video encoders in combination with per-frame encoding or encoding 2 frames together [137, 146, 190], but using video encoders with more frames lags behind using image encoders (such as SigLIP 2 [139]). However, when encoding each frame of a video individually, the number of visual tokens increases linearly with the frame sampling rate and the length of the video. This leads to a high compute cost and has led to a rise in works exploring efficient video encodings [79, 129, 149, 164, 170].

The best performing video LLMs [5, 25, 114] are closed-source proprietary models. While they are very capable, not much is known about how these models are trained and what data they use. By contrast, while some open weight models have been released [17, 146, 149, 170, 190], most don't release their training recipes or don't release their training data. A few projects do release all the training details and data [22, 184], but use biased data generated by proprietary VLMs (such as GPT4 and LLaMA3 [4, 18]). Hence, there is a need for a fully open SoTA training pipeline for Video LLMs that does not use previously trained multimodal LLMs to generate data.

Video-language instruction tuning datasets. The popularity of Video LLMs has also led to an increase in methods to develop instruction-tuning data for them. The current dominant paradigm involves generating synthetic instruction data by first segmenting videos into clips, generating descriptive captions for each clip, and then using a powerful LLM to synthesize video-level captions and QA pairs [17, 19, 22, 184]. However, a critical limitation of these approaches is their reliance on closed-source Video-Language Models (VLMs) for the initial clip captioning step. This introduces an inherent, often proprietary, bias into the generated data, as the underlying VLM's training data and biases are inaccessible to the research community.

Our Molmo2-CapQA dataset is generated through a similar pipeline but utilizes a video captioner trained on our fully open Molmo2-Cap to generate video captions. We segment each video into multiple scenes, caption each scene, and then provide these to an LLM along with the video metadata to generate 1M QA pairs. Another strategy used for generating QA pairs is to have annotators work with an LLM provided with an image caption when gener-

ating QA pairs [29], and we extend the same to video data to generate our Molmo2-AskModelAnything.

Video tracking. Early video tracking focused on bounding boxes for a closed set of objects [30, 110]. Since then, the field has branched into specific subtasks, including track any point (TAP) [35, 63] and tracking object segmentations [6, 53]. Object segmentations improved accuracy and granularity, but tracking was still limited to a closed set of objects. Moving beyond a closed set of objects to an open vocabulary has led to a rise in language-guided video object segmentation (VOS) [166]. A variety of new specialized models have been trained to track object [3, 11, 81]. Unlike Molmo2, these models are specialized and do not support other capabilities.

Previous methods, like Ref-VOS [12] and MeVis [31], support the language-guided VOS task by augmenting existing tracking datasets with complex referring expressions. However, we noticed a lack of language prompts referring to multiple objects or diverse actions. For our Molmo2-VideoTrack dataset, we similarly add to existing datasets by asking annotators to craft non-trivial text queries that apply to object tracks, with a focus on queries that describe multiple objects. For segmentation masks, we source videos and tracks from diverse open-source segmentation tracks [12, 33, 108, 122] and use a data pipeline to produce masks from bounding-box tracks [30, 37, 44, 126, 133, 140, 144, 174, 183, 186].

Video pointing. Multimodal LLMs that support point grounding in an image have recently become quite common [1, 10, 20, 25, 29, 171]. The training data used in these works is collected using automated object detectors, using existing referring expression datasets [68, 88, 175] or through manual human annotation [29]. We extend the human annotation pipeline approach to videos by adding a frame-selection phase. We also propose generating some queries through an LLM based on the caption to ensure the queries are complex and diverse.

I. Limitations

Here we discuss some of the limitations of the Molmo2 models.

Closed image ViT. Even with OLMo 3 as the LLM, our models still utilize a closed-data SigLIP 2 image encoder [139]. We chose to use SigLIP 2 because there are currently no competitive open-data encoders. We call upon the open-source community to explore such alternatives in future work.

Use of closed LLMs. We use closed text-only LLMs for data generation, as is common practice [89]. This reduces the transparency of our data collection pipeline. However, we believe that future open LLMs will become sufficiently proficient to be used in place of closed ones to reproduce

this dataset in a fully open manner. It is still important that we avoid using closed *VLMs*, which would create a circular dependency (training our *VLMs* would require first building a *VLM* to generate the training data) and therefore cannot lead to a fully open system in the same way.

Video grounding repeating points. For both video tracking and pointing, we sometimes observe that the model produces degenerate outputs, such as a long line of points on one frame or the same point for every frame. This is particularly common when pointing to high-frequency objects or on long videos, so this could likely be mitigated by sourcing more training data to better cover these cases. We also observe the issue is less common in specialized models, so we hypothesize that there might be some interference between the tasks in the joint training mixture which leads to this behavior.

Video grounding. Video grounding is less consistent than image grounding. Our metrics reflect this, with none of the models we tested reaching more than 40% on either our counting or pointing metrics, while image models often achieve 70-90% on image grounding metrics like Point-Bench.

We believe this is partly due to the inherent complexity of the task. Video grounding typically requires looking at much more visual content, and pointing at more things, than image grounding. Video grounding also requires re-identification, meaning understanding whether two objects in two different frames are the same object or not, which can be challenging. We also think that the lower resolution typically used when processing long videos, and the fact that the vision encoders are often not pre-trained on videos, could be contributing factors.

Long video grounding. Grounding has limited support for long (3 minutes+) videos because our grounding training is limited to that length. Handling longer videos is complicated by the fact that we would have to lower the fps when sampling frames to < 2 . This would result in our annotations, which are always at 2 fps, not being aligned with the selected frames. A possible solution is to customize how frames are sampled in these cases to ensure that all grounding annotations are selected.

Point tracking. Molmo2's generated tracks will sometimes change the location of its output point on the target object. This is likely because our tracking data generation pipeline does not always ensure that the point is consistently placed within the target object for every frame. Future improvements in generating points from bounding box or segment mask data could mitigate this issue.

Captioning. We observe that Molmo2 can sometimes generate repeating text when generating a very long video caption using greedy decoding. This is a known issue with

LLMs [52], including Qwen3*. However, we also think that the limited captioning training data contributed, as well as the high length of the captions (we observe that this typically occurs after generating thousands of tokens). We do not observe this behavior for other tasks.

J. Qualitative results

We show qualitative examples from Molmo2-8B. Each figure shows a query, the response from the model, and selected frames from the input video. The returned points are annotated with pink dots. Successful examples are shown in Figure 34 and Figure 35. We also show some failure cases in Figure 36.


*<https://huggingface.co/Qwen/Qwen3-4B> Best Practices

← Back to Videos (You will lose your progress for this video)

Task Instructions (click to expand)

UI Quick Guide: Write query text below video, use +/- buttons on right to select objects → click "Save Query" → repeat 3 times → click "Submit Annotations"

video: v_gONyhvBy0QY_c003_0 Masks Shown Click to stretch video Video Loop OFF




Precise Time: 11.01s

[10/11] Critical Rules - Check Before Saving Query (click to collapse)

- Select only the **SUBJECT** (not every object mentioned)
"person making a phone call" → select [3] person only, **NOT** [7] phone
- Query must **ONLY** match selected objects - use "ALL" text

Video With All Masks (click to hide masks)



Click to open video with boxes in fullscreen in a new tab

Precise Time: 11.01s

Selectable Objects (click for explanation)

Show All	Hide All					
<input type="checkbox"/>	<input type="checkbox"/>	[1] person (Start: 26.80)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[2] person (Start: 0.00)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[3] person (Start: 0.00)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[4] person (Start: 0.00)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[5] person (Start: 0.00)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[6] person (Start: 0.00)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[7] person (Start: 0.00)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[8] person (Start: 0.00)	✓	✖	+	×
<input type="checkbox"/>	<input type="checkbox"/>	[9] person (Start: 0.00)	✓	✖	+	×

Figure 19. Crowdworkers annotating object text queries.

Read the instructions below before starting the validation.

Validation Instructions (click to show)

CRITICAL: A query must **APPLY ONLY** to the selected objects (i.e. Are the selected objects the only ones that match the text?). If other objects in the video (**whether HAS MASK OR NOT**) could match the query, also mark it as **"Has Issues - Missing Objects"**.

CRITICAL: Selected objects should **ONLY** be the **SUBJECT** of the query (e.g. Person riding horses → Person [1],[2],[3]). If selected objects include other objects in the query but are **not** the **subject**, mark it as **"Has Issues - Wrong Query"**.

[10/23] Click to see good and bad examples (new tab)

Reference Video Masks ON Click back to original ratio Loop ON

[10/23] Please check if semicolon (;) is used correctly to separate multiple queries for the SAME object set. It should NOT be used as additional instruction to exclude objects. See the new examples linked above.

Queries to Validate (3)

Query 1: "people dancing"


Query 2: "people wearing pink shirts"

Query 3: "projection screens"

Overall Feedback (Optional)

Any general comments about the annotation quality or patterns you noticed...

Validated: 0/3 queries Submit Validation



Precise Time In Seconds: 2.16

Figure 20. Crowdworkers validating object text queries.


Instructions

You will watch a few clips from the video **WITHOUT THE AUDIO** and talk about them. We have listed some questions below for you to talk about. You can focus on the most relevant ones or talk about anything else interesting.

- Who or what are the main **subjects**? (e.g. people, animals, or other objects, ...)
- What do they **look** like? (e.g. age, clothing, color, texture, size, shape, material, ...)
- What are they **doing**? (e.g. events, activities, sports, ...)
- What are their **counts**? (e.g. how many are there?)
- How do things **change**?
- What is the main **setting**? Does it change?
- What other **details** (e.g. texts) are noticeable?


← Back to Videos (You will lose your progress)

Clip 1 [0:00:0:10] (DO NOT TURN ON THE AUDIO)



Check this box if the video clip is hard to talk about without the audio.

Clip 2 [0:10:0:30] (DO NOT TURN ON THE AUDIO)



Editing time: 10s

Correct mistakes in your transcript here

This video clip starts with a branded intro for FurnitureCare Network, which is part of the Castelan Group. You see a light grey background, with a shiny blue globe-style icon with white curved lines on the left, and the words "FurnitureCare Network" in blue next to it. Up in the top-right corner there's a smaller CASTELAN GROUP logo, with a little stepped blue line above the word "Castelan."

Then, along the bottom, a row of rounded blue icons slides in. Each one has a simple white outline of a different household item — a chair, a bed, an armchair, a rolled-up carpet, and a table — showing the different types of furniture the network deals with.

Restart Recording

Submit

Word Count: 0 / 50


Talk about clip 2 for at least 20 seconds.

Start Recording

Submit

Figure 21. Video clip captioning interface. Crowdworkers are instructed to annotate captions for video clips in sequence.

Watch the entire video below [DO NOT TURN ON AUDIO].



I finished watching the video. Proceed to the next step.

Word Count: 323 / 150

Editing time: 10s

Correct mistakes in your transcript here

So, the video is basically an advert for an at-home fabric stain-removal service. It starts on a clean white background with blue text that says "Stain Removal from Fabric", next to a blue square icon with the outline of an armchair. Along the bottom it explains that the stains are removed in your home. The same screen pops up again, but this time it adds a call to action: "Call 0870 320 0063."

Then it cuts to a man driving a branded service van, so you get the idea that a technician is on the way to someone's house. After that, you see a close-up of a pale woven fabric with a round reddish stain on it. A hand starts scrubbing the stain with a brush, showing how the cleaning process works.

Next, there's a close-up of a hand pressing a folded white cloth or paper towel firmly onto the fabric to blot up the moisture. A green spray bottle is used to spray cleaning solution onto the area, and you can see the stain fading. Then they use a small extraction tool—like a little handheld wet vac—to pull the liquid and residue out of the fibres. Finally, a motorised round brush goes quickly back and forth over the spot to lift and smooth the fibres, and the carpet or upholstery ends up looking clean and fresh.

After that, the video moves on to other kinds of repairs. You see a close-up of the edge of a wooden table with light scratches and white marks, and a hand working over it with a small pad or cloth, as if polishing the damage away. Then there's damaged upholstery: two hands holding a pale leather or vinyl surface with a tear in it. Another shot shows a repair tool or applicator working on a similar rip in fabric, making it clear that they don't just deal with stains—they can fix tears and scuffs as well.

Restart Recording

Submit transcript and finish.

Figure 22. Video captioning interface. Crowdworkers are instructed to annotate captions for complete videos.

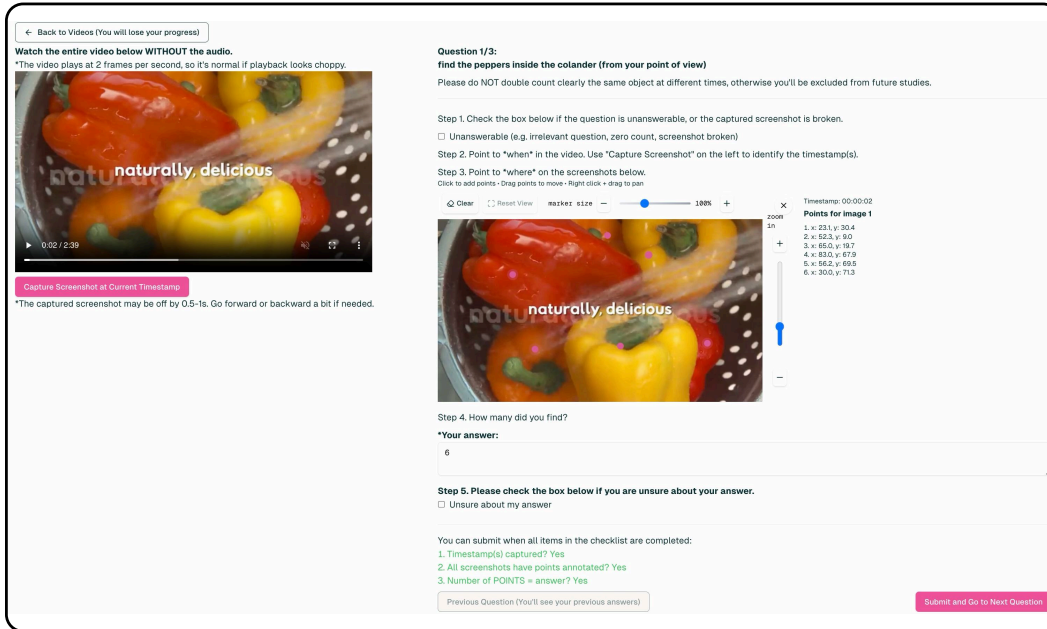


Figure 23. Video pointing interface. Crowdworkers are instructed to annotate points for object instances to answer visual questions.

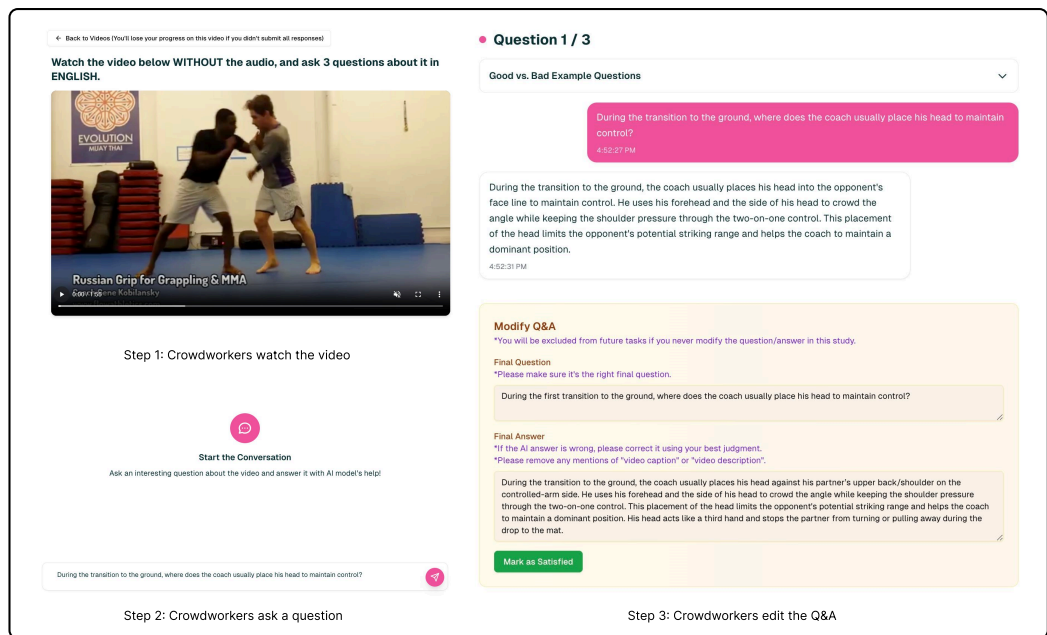


Figure 24. AskModelAnything interface. Crowdworkers are instructed to ask model non-trivial visual questions and finalize Q&A.



Figure 25. Random examples from Molmo2-Cap. Prompts are generated from our captioning prompt templates.

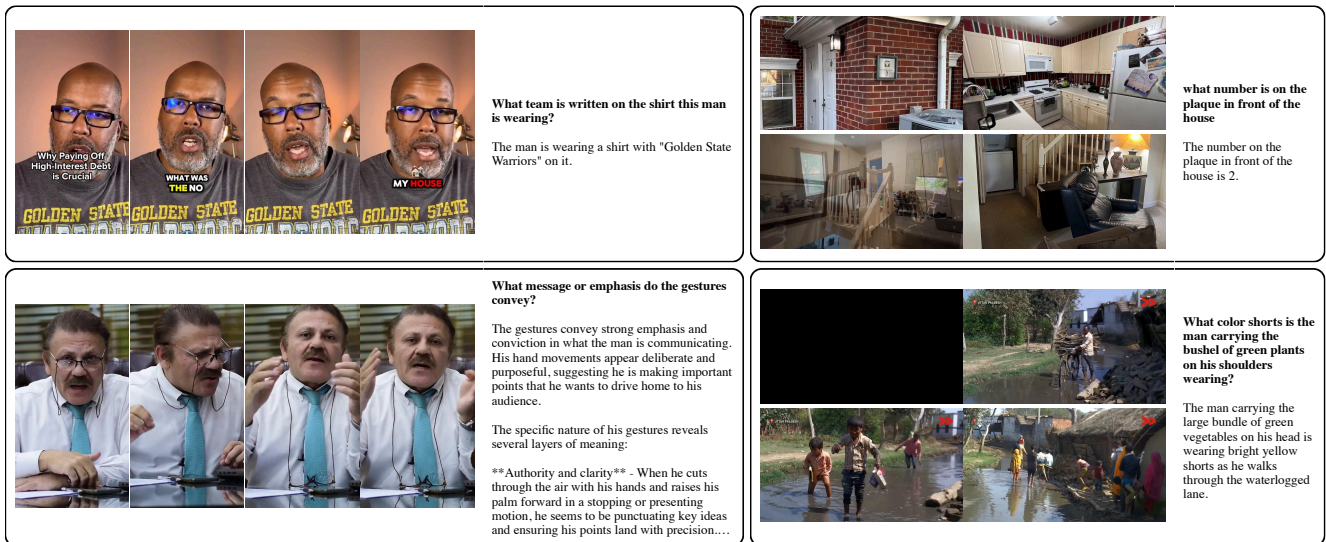


Figure 26. Random examples from Molmo2-AskModelAnything.

<p>BUCKLE UP Incident Report Case Number 20151023-119</p> <p>...ly 2200 hours, I was on routine patrol on car 527 traveling... ...I observed a tan Cadillac in front of me in the left lane... ...I observed the driver throw a lit cigarette out of the window... ...I indicated my intent to pass the vehicle... ...I also jerk like car around other cars in an attempt to pass... ...The driving of the Cadillac was very erratic... ...I also jerk like car around other cars in an attempt to pass... ...The driving of the Cadillac was very erratic... ...I also jerk like car around other cars in an attempt to pass...</p>	<p>How many digits are in the case number shown on the report page? Short answer:</p> <p>11</p>		<p>What human relationship is depicted by the small sculpture above the building's door? Only return the correct answer option.</p> <p>A. Two men shaking hands. B. A woman holding a child. C. A couple embracing. D. A soldier carrying a wounded comrade.</p> <p>B. A woman holding a child.</p>
	<p>What specific action does the deer closest to the camera perform? Only return the correct answer option.</p> <p>A. It stamps a hoof. B. It licks its lips. C. It lies down. D. It shakes its head vigorously.</p> <p>B. It licks its lips.</p>		<p>In the opening advertisement, which way does the scrolling "listen to Dr. Tony Evans" text move? Only return the correct answer option.</p> <p>A. downward B. left C. upward D. right</p> <p>C. upward</p>

Figure 27. Random examples from Molmo2-CapQA.

	<p>When the video shows the outdoor yard with boats under tarps, what does the narrator say drives much of the economy near the shore? Only return the correct answer option.</p> <p>A. Lobster fishing fleets B. Tourist beach resorts C. Sailboats and their maintenance D. Offshore wind turbines</p> <p>C. Sailboats and their maintenance</p>		<p>The word SUSTAINABILITY is highlighted over nature scenes; why is this concept emphasized according to the speaker? Only return the correct answer option.</p> <p>A. Because tourism requires more development B. Because oceans are always calm and safe C. Because wildlife prefers urban environments D. Because nature education teaches how to live in a balanced and sustainable way</p> <p>D. Because nature education teaches how to live in a balanced and sustainable way</p>
	<p>Right after she says to let the glue sit for a few seconds, what element does she press onto the center of the base? Only return the correct answer option.</p> <p>A. a green succulent B. a spray of baby's breath C. a yellow billy button D. a yellow ranunculus flower</p> <p>D. a yellow ranunculus flower</p>		<p>Right after he says to be firm but not too hard while lining up the tube over the housing, what does he do on screen? Only return the correct answer option.</p> <p>A. Pushes the new tube down onto the connector housing to seat it B. Twists the tube counterclockwise to remove it C. Wraps the joint with electrical tape D. Pours oil into the opening</p> <p>A. Pushes the new tube down onto the connector housing to seat it</p>

Figure 28. Random examples from Molmo2-SubtitleQA.

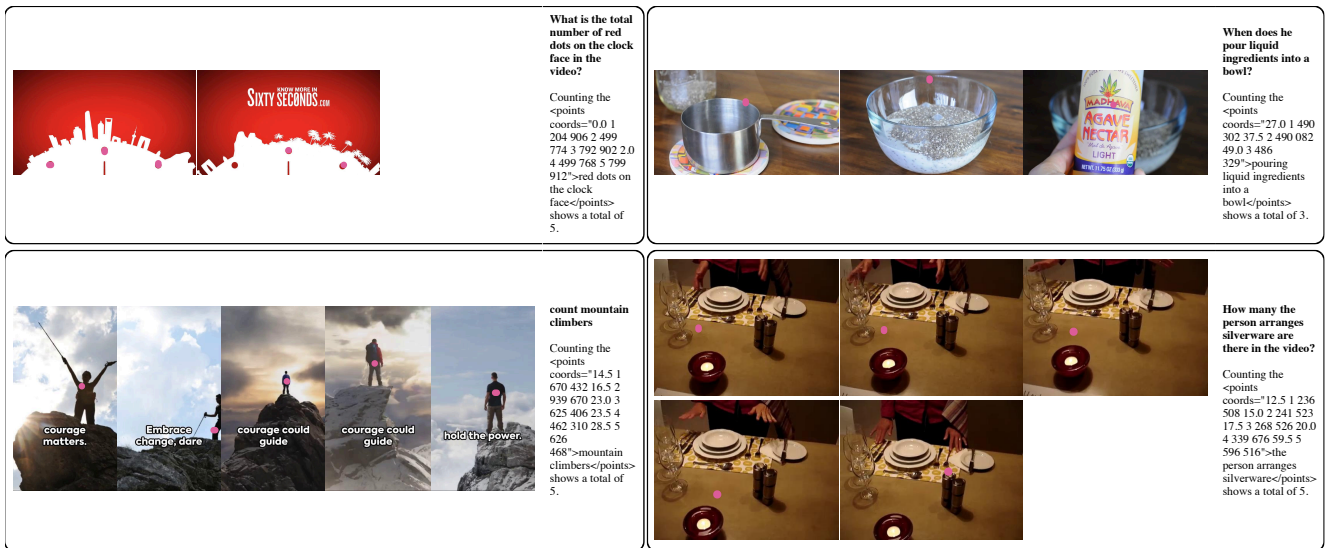


Figure 29. Random examples from Molmo2-VideoPoint. Points are shown in pink, output text follows Molmo2's point formatting.



Figure 30. Random examples from Molmo2-VideoTrack. Points are shown in different colors that are shared between the same objects, output text follows Molmo2's point formatting.

Are these canyons?

The first image definitely shows a canyon environment, with narrow, steep rock walls characteristic of slot canyons often found in places like Utah. We can see a rock climber navigating the tight space between smooth, reddish-brown walls.

For the second image, while we can see a mountainous landscape in the foreground with a truck parked on a hillside, it's possible there could be canyons beyond what's visible in the image. The jagged mountains in the background could potentially hide canyon formations that aren't immediately apparent from this vantage point.

...

How many people can we see in image 1 and Image 2 combined?

The total number of people visible in both images combined is approximately 57.

In Image 1, we see a large formation of soldiers on a sandy surface. The group consists of roughly 55 individuals arranged in rows, with the front row kneeling and the back rows standing.

Image 2 shows two soldiers in a snowy environment. One soldier is holding a rifle, while the other is...

Which image shows the single bookcase with the most individual shelves?

Let's consider each image:

Image 1 shows a bookshelf with 12 cubby holes arranged in a 3x4 grid.

Image 2 depicts three bookshelves, but we don't have specific information about the number of shelves on each.

Image 3 only shows two visible shelves of a bookcase.

Image 4 presents a large bookshelf with five shelves, divided into two sections.

...

Which images show a tiger covered in snow?

Image 2 shows a tiger covered in snow. The tiger is lying down in a snowy forest with its body dusted lightly with snow, clearly indicating snow on its fur.

Image 1 depicts a tiger interacting with a snowman in a snowy environment, but there's no snow directly on the tiger's body.

In image 3, while the tiger is walking through a thick blanket of snow with snow falling gently around it, only a small amount o...

Figure 31. Random examples from Molmo2-MultiImageQA.

cosyn_doc_exp: Would a refund for 100 tabloid color copies on glossy photo paper require manager approval under the complaints policy? Provide reasoning steps and then give the short answer.

Price for color tabloid at 100-499 is \$0.99 each. $100 \times 0.99 = \$99$. Glossy photo paper adds \$0.50 per sheet: $100 \times 0.50 = \$50$. Total = \$149. The complaints policy requires manager approval for refunds over \$50, so approval is needed (<IMAGE-2> for pricing, <IMAGE-1> for policy) Answer: Yes

cosyn_diagram_exp: Which scheduled task most directly corresponds to building the DFA engine with state transitions shown in the lexer architecture? Provide reasoning steps and then give the short answer.

<IMAGE-2> centers on a lexer DFA engine that reads characters, classifies them, transitions states, and emits tokens. In <IMAGE-1>, the task that matches implementing such a lexer mechanism is "Implement Lexer," making it the most direct correspondence Answer: Implement Lexer

cosyn_chart_exp: Where is the nest marker located in terms of the coordinate axes? Provide reasoning steps and then give the short answer.

In <IMAGE-1>, a red circular marker labeled "Nest" is plotted; it sits exactly at the intersection of the x and y axes Answer: At the origin (0, 0)

cosyn_table_exp: Starting from the top-ranked researcher in the performance table, find their group's number of NSF successes in the grant table Provide reasoning steps and then give the short answer.

The top-ranked researcher is Peter Fratzl in <IMAGE-2>. In the grant table, the Fratzl Group row shows NSF successes as 15 in <IMAGE-3> Answer: 15

Figure 32. Random examples from Molmo2-SynMultiImageQA.






	<p>Show me where a area to right of plant is in image_1.</p> <p><points coords="1 1 592 799">area to right of plant</points></p>		<p>count cursive writing in image_1, image_2?</p> <p>Counting the <points coords="1 1 754 817 2 789 813 2 3 473 752">cursive writing</points> shows a total of 3.</p>
	<p>Where are the Football cleats?</p> <p><points coords="1 1 193 391 2 349 376 3 384 846 4 432 376 5 445 843 6 578 786 7 616 892 8 647 360 9 647 391 10 862 880 11 922 387 12 938 871 13 956 291 2 14 009 726 15 251 302 16 256 984 17 278 296 18 324 512 19 335 488 20 351 661 21 362 415 22 364 629 23 368 387 24 441 518 25 450 506 26 496 315 27 503 381 28 525 341 29 548 734 30 557 532 31 558 159 32 570 153 33 580 546 34 583 450 35 601 748 36 603 421 37 626 399 38 634 157 39 651 407 40 673 383 41 683 163 42 724 500 43 871 504 44 889 482 3 45 021 672 46 066 716 47 211 585 48 314 535 49 329 431 50 433 790 51 435 409 52 448 735 53 519 644 54 533 689 55 543 617 56 575 243 57 585 578 58 671 550 59 681 285 60 743 522 61 766 483 62 825 347 63 905 463 64 986 377 4 65 014 719 66 067 737 67 208 768 68 227 820 69 258 630 70 324 642 71 452 724 ..."></p>	 <p>Point to any cockpit in all images.</p> <p><points coords="2 1 764 549">cockpit</points></p>	

Figure 33. Random examples from Molmo2-MultiImagePoint. Points are shown in pink, output text follows Molmo2's point formatting.




Q: Describe this video.

A: Under bright stadium lights at night, a packed crowd fills the stands as a college baseball game unfolds on a lush green field with crisp white baselines and a brown infield. The home team wears white pinstripes with blue helmets and blue socks, while the visiting team is in maroon jerseys with gray pants and black helmets...



The sequence shifts to a close-up of a player in a white pinstriped jersey with blue accents and a blue cap, the number "21" visible on his back, his right arm extended forward and his left arm bent at the elbow, a black glove on his left hand. Behind him, a green wall with a yellow stripe and a white railing frames the scene, with a few spectators visible in the stands...

The celebration erupts on the field. Players in white pinstripes with blue helmets and blue socks converge in a jubilant pile, jumping and hugging amid the cheers of the packed stadium....



Q: How many national flags are there in the video?

A: Counting the `<points coords="7.0 1743 468 9.5 2 776 500 13.0 3 743 470 16.0 4 642 471 19.5 5 695 469 23.0 6 766 471 25.5 7 776 473 28.5 8 743 471 31.5 9 677 469 35.0 10 776 471">national flags</points>` shows a total of 10.

Q: Track the horse.

A: `<tracks coords="0.0 1 381 528 0.5 1 340 562 1.0 1 300 507 1.5 1 431 569 2.0 1 465 535 2.5 1 498 514 3.0 1 431 410 3.5 1 292 472 4.0 1 354 493 4.5 1 531 347 5.0 1 656 514 5.5 1 617 340 6.0 1 398 264">horse</tracks>`




Figure 34. Qualitative examples of captioning, counting, and tracking from Molmo2-8B



Q: Find cars on the highway.

A: < points coords="2 1 222 460 2 365 140 3 545 420 4 615 950 5 805 340 6 895 600 3 7 298 060 8 300 170 9 450 270 10 500 420 11 570 010 12 620 770 4 13 082 344 14 178 128 15 198 190 16 290 300 17 368 534 18 450 514 19 500 764 20 658 724 21 940 744" > cars on the highway < /points >





Q: How long should I marinate chickens?

A: You should marinate the chicken for 45 minutes in the refrigerator.



Q: Which team won the game?

A: The winning team was UCLA. You can see their blue-and-white pinstriped uniforms throughout the celebration, and the championship trophy they hoisted displays "UCLA" along with "Baseball National Champions 2013."



Q: Point to visual artifacts.

A: < points coords="1.5 1 081 491" >

Figure 35. Qualitative examples of pointing and QA from Molmo2-8B

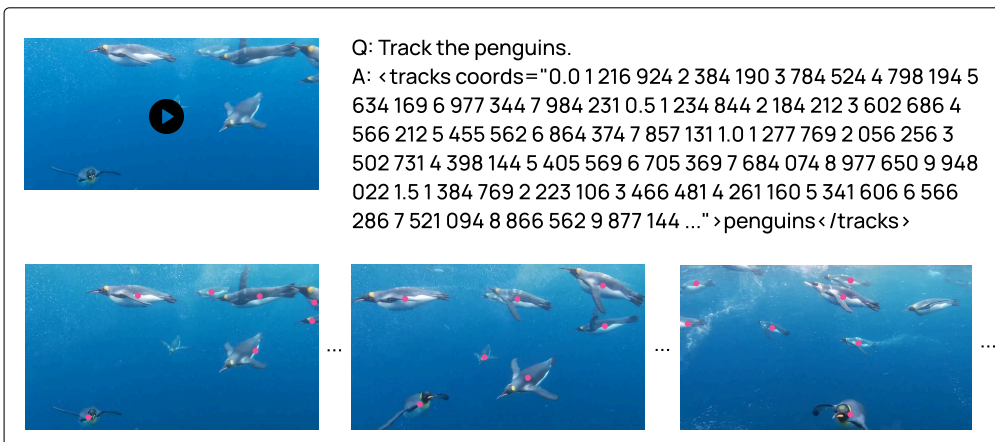
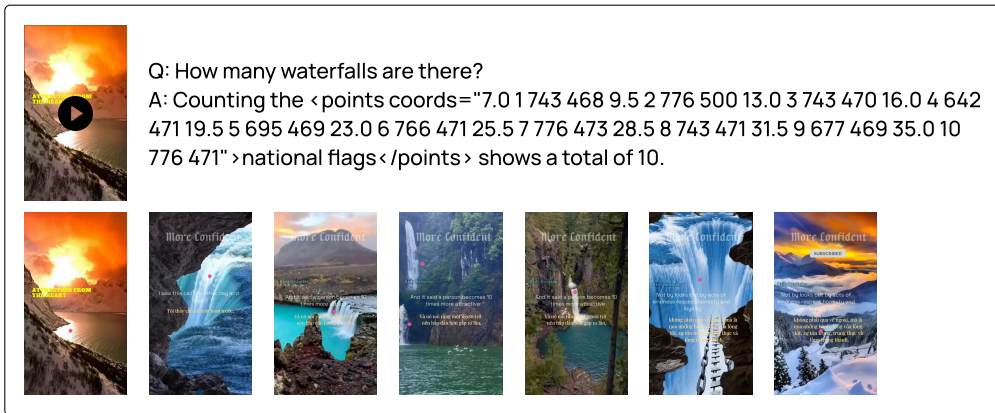
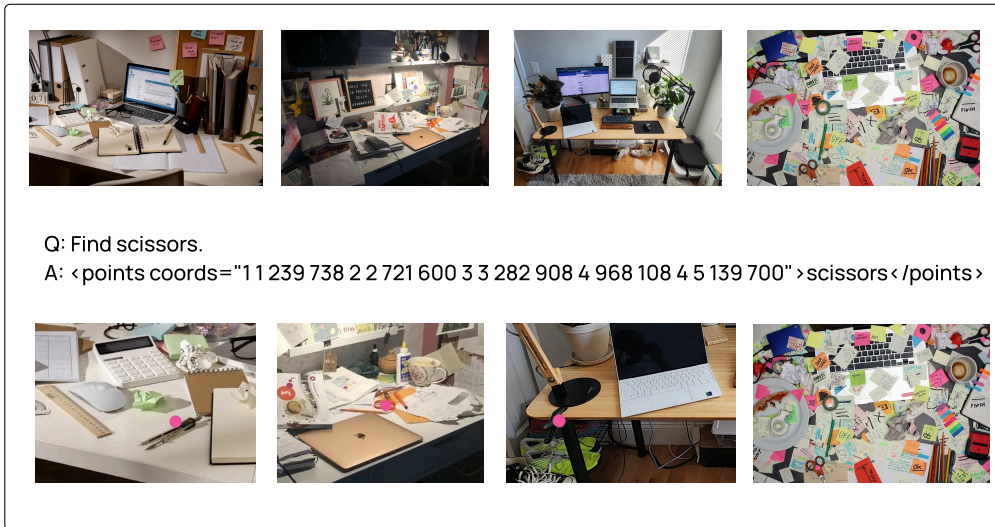


Figure 36. Qualitative failure cases from Molmo2-8B. The model identifies false positives in the first two examples and misses several of the penguins in the bottom example.