Molmo and PixMo: Open Weights and Open Data for State-of-the-Art Vision-Language Models Supplementary Material

Appendix

The appendix includes the following sections:

- §A Model Details
- §B Training Details
- §C Evaluation Details
- §D Result Details
- §E Ablation Details
- §F Data Details
- §G Dataset Examples
- §H Related Work

A. Model Details

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

Our method of encoding images is shown in Figure 5. Cropping is done by first choosing a rectangular grid (*e.g.*, a 2×2 , 3×1 , *etc.*) where each square of the grid matches the ViT's input size. When using overlapping crops, these squares are moved closer together so that they overlap by a fixed margin (we use a margin of 4 patches or 56 pixels), which reduces the overall size of the grid.

Then the image is up-scaled to fit within that grid as well as possible while preserving its aspect ratio by making either its height or width the same size as the grid. The grid is chosen to require the least amount of up-scaling, and in the event of ties, to minimize its size. We also set a maximum number of crops, and if the image cannot be covered by that many crops, the image will instead be downscaled to fit the grid, and the grid is chosen to minimize the amount of down-scaling required while not exceeding the maximum number of crops. In either case, the re-scaled image is padded with black borders so that it exactly fits the grid, and then crops are extracted from this padded image. The low-resolution crop is built by resizing and padding the image so it matches image ViT's supported resolution.

Each crop is processed independently by the ViT and connector to get visual embeddings of each patch. A learned embedding is added to the patch features from each crop (before the connector is applied) depending on whether that patch includes no padding, some padding, or is all padding, so the model can distinguish padding from images that naturally have black borders. These embeddings are arranged with special tokens as described in Section 2, also shown in Figure 5 right. For image/text inputs we encode the input image first, followed by any text.



Figure 5. Converting an image into tokens. The image (top left) is turned into a single low-res and several overlapping high-res crops (bottom left). Padding (the black borders) is used so each crop is square and the aspect ratio of the image is preserved. The final token sequence for the image (right, arranged top-down left-to-right with line breaks for clarity) is built by extracting patch-level features from the crops, shown here using images of the patches, and special tokens. An image start and image end token are placed before/after the high-res and low-res patches, and column tokens are inserted after each row of patches. This example uses 4 high-res crops and extracts features from $36 (6 \times 6)$ patches per crop, in practice Molmo typically uses 12 high-res crops and extracts features for crop.

A.1. Image Encoding

A.2. Hyper-Parameters

Hyper-parameters for the Molmo models and the AdamW [50, 73] optimizers are shown in Table 6. The connector MLP uses the same intermediate dimension as the LLM, so its size depends on the LLM. The connector pooling layer and ViT architecture are the same between all models. All runs used a cosine learning rate schedule ending at 10% of the peak learning rate [72].

Learning rates are similar between the models, except we find it helpful to reduce the learning rate for Molmo-72B. We also find Molmo-72B learns faster than the other models and can therefore be trained for fewer steps. Molmo-7B-O was trained for slightly longer due to a minor configuration difference, but we do not think it affected performance.

A.3. Implementation

Our implementation uses PyTorch with Fully Sharded Data Parallel (FSDP) [135] based on the OLMo codebase [37]. We do not use FlashAttention [28, 29] since it does not sup-

		1 B -E	7B-D	7B-O	72B-D			
Image Encoder	Params Dim MLP Dim Act. Heads KV Heads Layers Image Size Patch Size Dropout		2900 102 409 GEL 16 23 336×: 14 0.0	n 4 6 U 336				
V/L Connector	Params Pool Size Pool Dim Pool Heads MLP Dim Act.	12m 1024	12m 110m 74m 310m 2×2 1024 16 1024 37888 22016 59136 SwiGUU					
	Dropout		0.0					
V	Params Embed Dim MLP Dim	1.2b (6.9b) 50304 2048 2048×64	7.6b 152064 3584 37888	7.3b 100352 4096 22016	72b 152064 8192 59136			
A H K L T D	Act.	SwiGLU						
	Heads KV Heads Layers Theta Dropout	16 16 16 10k	28 4 28 1m 0.1	32 32 32 0.5m	80 8 64 1m			
e-Train	Warmup ViT Warmup Con. Warmup LLM LR ViT LR Con.		200 200 200 6e-0 2e-4	0) 0 5 4				
Pr	LR LLM Cosine Decay Eps. Betas Batch Size Steps	2e-5 2e-5 2e-5 1e-5 10% 1e-6 0.9, 0.95 128 22.3k						
ne	Warmup ViT Warmup Con. Warmup LLM		200 200 200))				
Fine-Tu	LR ViT LR Con. LR LLM	5e-6 5e-6 2e-5	5e-6 5e-6 1e-5	5e-6 5e-6 1e-5	3e-6 3e-6 5e-6			
	Cosine Decay Eps. Betas Batch Size		10% 1e-0 0.9, 0 256	6 5 .95 5				
	Steps	30k	30k	32k	20k			

Table 6. Model and training hyper-parameters. Molmo-1B-E has 1.2b active parameters, but 6.9b total. Its LLM MLP layers have 64 experts with 8 active at once.

port the more complex masks that are required for multiannotated images, but we find using PyTorch's Scaled Dot Product Attention (SDPA) achieves close to the same speed.

To improve throughput, we utilize PyTorch's Automatic Mixed Precision (AMP) module³, which enables most oper-



Figure 6. **Training loss curves** for Molmo-7B-D with model weights and gradient reduction in bfloat16 (blue) and float32 (pink). Float32 is our default configuration.

ations to run in half-precision with bfloat16 numbers. However, as shown in Figure 6, keeping model weights and performing gradient reduction in half-precision degrades training loss, so these are retained in full precision. Additionally, computations for layer normalization [8] and Rotary Position Embedding (RoPE) [101] are explicitly carried out in full precision.

When computing gradients with FSDP, each GPU computes a gradient on a small mini-batch of examples, after which 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) since those examples tend to be paired with a smaller divisor if using the device-local number of loss tokens. Using the average number of loss tokens across all devices largely resolves this issue since our global batches are much larger than the device-local batches. This issue has been discussed in other places⁴ [40] and is known to have affected many codebases.⁵ We observe that captioning performance can drop by 0.5-1 points without this fix.

During fine-tuning, mixing is done within each batch so batches contain examples from a variety of tasks. We set a maximum sequence length of 2304 for both pre-training and fine-tuning, and truncate examples longer than that (in practice, truncation only happens for certain synthetic datasets like DVQA [46] which contains many annotations per image, or for the occasional outlier example in other datasets).

We find training to be stable, without loss spikes or NaNs, likely in part because we use pre-trained models.

³https://pytorch.org/docs/stable/amp.html

⁴https://unsloth.ai/blog/gradient

⁵https://github.com/huggingface/trl/issues/2175



Figure 7. **Captioning precision and recall with different length hints** for Molmo-7B-D after pre-training. A short hint reduces recall since the model describes fewer things, but can boost accuracy since the description tends to focus on the more salient, easier-tounderstand parts of the image.

B. Training Details

Here we discuss the training mixture and how tasks are formatted during pre-training and fine-tuning.

B.1. Pre-Training Task Details

During pre-training, we train on each image paired with its caption and one of its audio transcripts. For images with multiple transcripts, we select one randomly each epoch. We use multi-annotation training (see Section C) to train on both the caption and the transcript jointly.

We prompt the model with either "long_caption:" or "transcript:" for captions and transcript respectively (a natural language prompt is used instead during instruction fine-tuning). We also add a length hint: an integer providing a noisy hint as to the correct output length. This hint is computed as the length of the transcript/caption in characters, plus a noise factor drawn from a random normal with a standard deviation of 25. The hint is then divided by 15 and rounded down to keep the hint in roughly the range of 0 to 100. This noise is added so that the length functions more like a guideline than a hard constraint, leaving the model some flexibility to adjust the caption as appropriate for the image. For example, even with a long length hint, its preferable that a caption for a very plain image be short instead of becoming repetitive or inane due to lack of content to describe.

We add the hint to the prompt 90% of the time, for example: "long_caption_83:" for a length hint of 83, and 10% of the time no length hint is used to maintain the ability to output a default caption.

Adjusting the length hint allows a trade off between precision and recall in captioning, see Figure 7 (see Section C for captioning metric details). In all of these settings, the average caption length when using a length hint is within

name	rate	images	anno.	tokens	avg.crops
PixMo (Annotated)	38.1	1m	3.3m	350m	10.6
Points	28.8	220k	2.3m	160m	10.1
AskModelAnything	3.8	71k	160k	17m	10.0
Сар	3.2	712k	712k	160m	10.9
PointQA	2.4	14k	76k	11m	11.0
PixMo (Synthetic)	31.6	1.3m	3.3m	120m	11.0
Count	6.2	36k	37k	3m	11.9
CapQA	5.7	160k	210k	38m	10.8
Clocks	5.3	800k	800k	20m	10.5
Docs-Charts	5.2	120k	1.1m	34m	12.8
Docs-Other	4.0	71k	610k	15m	12.8
Docs-Tables	3.3	47k	420k	12m	12.1
Docs-Diagrams	1.9	16k	140k	3.6m	12.5
Academic	30.3	880k	25m	1b	8.0
TallyQA	3.9	130k	250k	4.6m	6.1
VQA v2.0	3.1	83k	440k	7.9m	6.7
AndroidControl	2.9	74k	300k	13m	11.0
A-OKVQA	2.8	17k	17k	380k	6.8
DocVQA	2.1	10k	39k	1m	12.9
TextVQA	2.0	22k	35k	700k	12.7
ChartQA	1.8	18k	28k	850k	9.3
ST-VQA	1.7	18k	25k	530k	4.6
InfographicVQA	1.6	4.4k	24k	670k	12.0
TabWMP	1.6	23k	23k	930k	2.3
PlotQA	1.5	160k	20m	930m	12.4
AI2D	1.3	6.2k	15k	630k	6.4
DVQA	1.1	200k	2.3m	51m	5.0
FigureQA	1.1	100k	1.3m	25m	6.1
OK-VQA	1.0	9k	9k	180k	6.8
ScienceQA	0.8	5k	6.2k	460k	4.3

Table 7. **Full list of instruction fine-tuning tasks**. Columns show the sampling rate, the total number of images and annotations (*i.e.*, the number of question/answer pairs), the number of text tokens using the Qwen2 tokenizer, and the average number of crops. The number of crops per an image can be at most 13 (one low-res, and 12 high-res), but can be lower for datasets with smaller images. Shaded rows show the total counts for all datasets in the category.

10 characters of the expected length, showing the models follow the length hints well. For our ablations, we report scores with a length hint of 65, which performs similarly or slightly better than using no length hint.

Preliminary experiments with mixing in other sources of captions (COCO Captions [17], Localized Narratives [93], or captions derived from Visual Genome annotations [52]) did not improve scores on our captioning metric, so we use PixMo-Cap alone for pre-training.

B.2. Fine-Tuning Task Details

Table 7 shows a list of our fine-tuning tasks. We only train on the train sets. Formatting and task-specific details are described below.

Multiple choice questions. For multiple choice questions in academic datasets (AI2D, A-OKVQA, ScienceQA), we append "Choices:", a newline, and then new-line separated options with capital letter answer labels. The model predicts the answer label only. Note some multiple-choice questions appear in other, more diverse formats in PixMoCapQA and PixMo-AskModelAnything.

Multiple answers. For datasets with multiple answers per question (*e.g.*, VQA v2.0), we only use the most common answer for training. If there are multiple answers that are equally common, we randomly select from among them each epoch.

Pointing. Pointing uses an HTML-like format. (x,y) coordinates are scaled to 0-100. For a single point, the format is:

```
<point x="10.0" y="10.0" alt="alt
text">Inline text/point>
```

For multiple points the format is:

<points x1="10.0" y1="10.0" x2="20.0"
y1="20.0" ... alt="alt text">Inline
text</points>

Numbering the points makes counting easier because the total count is always the number of the last point.

When interacting with users, we generally replace the point(s) text with the inline text, and show the image with the points using the alt text as hover text. For pointing and counting, the inline and alt text are both the name of what is being pointed at. For pointing-as-an-explanation these fields can be different.

PixMo-Points. Counting or pointing with a very large number of objects can lead to very long sequence lengths. To avoid memory errors we do not train on data with more than 40 counts; we expect to remove this limitation in future iterations of Molmo.

PixMo-AskModelAnything. "How many" questions are common in PixMo-AskModelAnything, but are not accompanied with pointing data. We observe that this can lead to the model failing to point when asked counting questions. To resolve this, we heuristically detect such questions and prefix them with an instruction to not point (*e.g.*, "Answer without points."), randomly selected from a pool of 20 such instructions.

AI2D. AI2 Diagrams requires labeling regions of the images with letters, and then training the model to answer questions by predicting the correct region by returning its letter. Evaluations in the literature have been mixed between labeling the regions with opaque boxes (*e.g.*, [5, 89]) and transparent boxes (*e.g.*, [10, 71, 106]). We train in both settings and present our main results with transparent boxes. Results with opaque boxes are in Section D.

For AI2D questions where the answers are just letters, we list the multiple-choice options without a letter prefix.

AI2D does not have a validation set, so we built our own custom validation set by separating out 384 images (with roughly 2000 questions-answer pairs). None of our models are trained on this set.

		pre-trai	in		fine-tune	e
	GPUs	time	GPU hr.	GPUs	time	GPU hr.
1 B- E	8	33.3	264	64	13.3	850
7B-D	64	8.6	550	128	11.2	1.4k
7B-O	64	8.9	570	128	13.5	1.7k
72B	128	33.3	4.2k	256	32.4	8.3k

Table 8. Training times for the Molmo models using H100 GPUs.

ChartQA. The ChartQA train set contains many synthetic questions (21k synthetic *vs.* 7k non-synthetic), which we observe can be noisy and lower quality. To reduce the weight of these examples we re-weight ChartQA so the total weight of the synthetic and non-synthetic examples are equal. This also means the training data better matches the validation and test data which are evenly split between synthetic and non-synthetic questions.

A-OKVQA. We train on the multiple choice questions and, for questions not marked as difficult direct answer, also use them as direct answer questions by not using the answer options. We use different style tags for direct answer and multiple choice versions of the questions.

TabWMP. For TabWMP we treat the task as short answer and do not show the multiple-choice options.

AndroidControl. We train on four input-output configurations: low-level instruction to action, high-level goal to action, low-level and high-level inputs to action, and highlevel goal to action with chain-of-thought reasoning. Only the instruction and screenshot are provided to the model as input; accessibility trees, action history, and a prompt with details like available actions are omitted. Target actions are represented as text output strings and (x,y) coordinates are scaled to 0-100 just as with regular pointing.

B.3. Training Time

Training time and number of GPUs used are shown in Table 8. All models were trained with H100 GPUs with infiniband connectivity.

C. Evaluation Details

Captioning metric (cap F_1). We measure captioning quality, relative to an evaluation set of 1500^6 images, using the harmonic mean of captioning *precision* and *recall*, *i.e.* the F_1 score. The evaluation set was gathered through a similar protocol as PixMo-Cap (selecting a small number of images matching a diverse set of categories), but the images were selected manually and are disjoint from images in PixMo-Cap. Each evaluation image has up to six audio transcripts associated with it. To define the precision and recall of a caption for an image, let g be the generated

⁶A few evaluations were done with a super-set of 2730 images. We do not expect this to have affected results significantly since the 1500 were a random subset of the 2730.

caption and T be the set of ground-truth transcripts for the image. We prompt GPT-40 to enumerate a list of all distinct atomic statements contained in g and, separately, the transcripts in T. We then prompt GPT-40 to match each item in the list of atomic statements from g to items in the list of atomic statements from T. To compute recall, we consider matches as true positives and unmatched items from T's list as false negatives. To compute precision, we prompt GPT-40 with the raw transcripts and the list of statements from qand ask it to say if each statement is consistent (a true positive) or inconsistent (a false positive) with the transcripts. (We avoid using the atomic statements from T when computing precision because it's a potentially noisy processing step that is not necessary.) We average precision and recall over all images in the evaluation set and compute the F_1 score of the averaged precision and recall values to produce our final summary metric: cap F_1 .

While this metric is imperfect (*e.g.*, GPT-40 makes mistakes, the transcripts do not contain all true statements about the image, *etc.*) we found that improvements to cap F_1 corresponded to improvements in our subjective impressions of caption quality, and thus it was a useful internal metric for guiding model and data design. Most of our model design and pre-training data decisions were based on improving captioning quality, see Section D and Figure 9 for more discussion.

Human evaluation. We defined 10 question categories and crowd sourced image-question pairs, using the same workers as for other annotation tasks. This resulted in the following categories and image-question pair counts:

- Output formatting: 1525
- Fine-grained detailed QA: 1510
- General: 1504
- Documents: 1499
- Captioning: 1493
- Counting: 1490
- Homework: 1489
- Charts: 1473
- Named entities: 1448
- Creativity: 1420

We performed two human evaluations, the first for a large set of models presented in our main results Table 1 and a second for a smaller set of ablation models in Table 5. To collect feedback, we presented an annotator with an image, a question, and the output of "model A" and "model B", without revealing the model identities. The annotator had five options: *tie (both bad), tie (both good), model A is better, model B is better,* or *I don't know*. The last option was for cases where the annotator did not know the correct answer (*e.g.*, a math problem they do not know how to solve). For the first study, we sampled image-question pairs randomly from the 10 categories, until we had collected ~450 feedback responses per model pair. For the second study,

model	score	95% CI	opennness
Gemini-Exp-114 [103]	1278	+28/-27	API only
ChatGPT-4o-latest (20240903) [90]	1256	+13/-13	API only
Gemini-1.5-Pro-002 [103]	1220	+15/-14	API only
Gemini-1.5-Flash-002 [103]	1219	+15/-17	API only
GPT-4o-2024-05-13 [90]	1213	+9/-9	API only
Claude 3.5 Sonnet (20240620) [7]	1187	+9/-7	API only
Claude 3.5 Sonnet (20241022) [7]	1184	+15/-15	API only
Gemini-1.5-Pro-001 [103]	1158	+9/-8	API only
GPT-4-Turbo-2024-04-09 [88]	1157	+7/-10	API only
Gemini-1.5-Flash-8B-Exp-0827 [103]	1137	+15/-13	API only
GPT-4o-2024-08-06 [90]	1131	+18/-20	API only
Gemini-1.5-Flash-8B-001 [103]	1133	+10/-15	API only
GPT-4o-mini-2024-07-18 [89]	1124	+7/-9	API only
Molmo-72B	1115	+18/-17	Fully Open
Qwen2-VL-72B [111]	1113	+15/-17	Open Weight
InternVL2-26B [104]	1096	+11/-10	Open Weight
Pixtral-12B-2409 [3]	1085	+13/-14	Open Weight
Llama-3.2V-90B-Instruct [5]	1085	+12/-14	Open Weight
Gemini-1.5-Flash-001 [103]	1087	+8/-8	API only
Molmo-7B-D	1076	+15/-18	Fully Open
Yi-Vision [4]	1070	+21/-26	Distilled
Claude 3 Opus [7]	1073	+6/-8	API only
Qwen2-VL-7B [111]	1068	+15/-14	Open Weight
Llama-3.2V-11B-Instruct [5]	1061	+14/-14	Open Weight

Table 9. **Chatbot Arena's vision leaderboard** for English queries. The table is up to date as of Nov. 13, 2024. We show up to 20 rows for clarity.

we used a refined methodology in which we first manually verified the quality of the question-image pairs, resulting in a fixed set of 500 questions (exactly 50 per category), all of which were used for each pair of models. After collecting the feedback, we removed the *I don't know* responses and computed an Elo ranking using the Bradley-Terry model, following the methodology of Chatbot Arena [21].

AndroidControl. To evaluate Molmo on AndroidControl, we provided only the task instruction (high-level or lowlevel) and the current screenshot as input. The results are reported on the in-domain data test set (IDD) and the metric is step-wise accuracy.

D. Result Details

Chatbot Arena. In Table 9 we report a summarized version of the vision leaderboard for queries in English from the Chatbot Arena [21], an independent third-party VLM evaluation. Molmo-72B outperforms all the fully open and open weight models but lags behind some of the propriety VLMs. As noted in the paper, we did our own Elo evaluation (see Section 5), in which Molmo-72B ranks higher (2nd place). The difference in rating likely stems from the types of questions evaluated. We do note our data includes many counting and image-description questions which are particular strengths of Molmo.

Clock reading. PixMo-Clocks is a novel source of clock reading data, a data type that is missing from most VLM

training data (for which data information is published; we cannot know for models without published data details, such as API-only models and many open-weight models such as Pixtral and Llama 3). PixMo-Clocks are entirely synthetic and show a variety of watch bodies and faces against plain backgrounds (see Figure 17 for examples).

We tested how well Molmo trained on this data performs on the in-the-wild clock reading benchmark introduced in [121]. The benchmark sources clock images from three different datasets, COCO [65], OpenImages [53] and Clock Movies, a newly collected dataset based on the film *The Clock* (2010).⁷ They are highly out-of-distribution relative to the PixMo-Clocks training data. We also benchmarked several API-only models and open-weight (+ opendata) models for comparison. We compare all of these VLMs against the model presented from [121] that is specialized at the single task of clock reading.

We used the same query for all the VLMs: "What time is being shown? Please respond only with the time as hours and minutes in HH:MM format.", and followed the official evaluation protocol.⁸ Table 10 highlights that all the VLMs including proprietary models struggle to read clocks, with the exception of Molmo; see the notable performance gaps between Molmo and other VLMs. Molmo-72B surprisingly underperforms Molmo-7B-D and MolmoE-1B. This might be partially due to that PixMo-Clocks accounts only for 5.3% of the fine-tuning data mixture and we trained Molmo-72B for fewer steps than the others. Augmenting PixMo-Clocks with real-world clock images could potentially increase performance, closing the gap between Molmo and the specialized clock reading model.

Despite training on synthetic data, we qualitatively observe the clocking-reading capabilities generalize effectively to more complex questions and to captioning. An example is in Figure 1 lower right of the main paper.

Pointing. To evaluate the model's pointing performance, we constructed an evaluation set of 493 image and pointing question pairs. Each example was manually verified to ensure that either there is no target object or each target object instance is annotated with a single point and an accurate segmentation mask. The segmentation masks are generated by SAM [51] using each ground-truth point as a prompt.

For cases with no target object, precision and recall are calculated as 1 if the model responds correctly (*e.g.* outputs "This isn't in the image.") and 0 otherwise. When a target object is present, we first compute the pairwise distances between the predicted points and the ground-truth points to serve as the cost in the Jonker-Volgenant algorithm [25, 45], which we then use to assign each predicted point to one of the ground-truth points. We then use the verified segmentation masks to determine if each predicted

model	acc.	hour acc.	min. acc.
GPT-40-0513 [90]	2.7	14.2	8.6
Gemini 1.5 Pro [103]	0.9	11.6	5.1
Claude-3.5 Sonnet [7]	6.6	22.3	17.5
PaliGemma-mix-3B [10]	6.1	21.0	15.8
Phi3.5-Vision-4B [1]	1.9	12.0	7.6
Qwen2-VL-72B [111]	<u>9.1</u>	<u>24.9</u>	18.4
InternVL2-Llama-3-76B [104]	3.3	16.3	9.9
Pixtral-12B [3]	1.7	11.9	6.7
Llama-3.2V-90B-Instruct [5]	3.4	17.9	10.1
LLaVA-1.5-13B [69]	0.8	11.6	5.7
xGen-MM-interleave-4B [119]	2.0	11.9	8.0
Cambrian-1-34B [106]	1.8	11.1	7.2
LLaVA OneVision-72B [59]	5.7	17.9	15.4
MolmoE-1B	65.8	77.9	74.1
Molmo-7B-O	64.2	76.3	73.8
Molmo-7B-D	68.2	78.6	76.0
Molmo-72B	65.6	77.1	73.7
Specialized single-task model [121]	78.9	84.2	82.9

Table 10. **Clock reading benchmark results**. We report the averages of overall, hour and minute accuracies, each evaluated on three different test sets based on COCO, OpenImages and Clock Movies, respectively. **Bold** numbers represent the highest VLM scores while the best numbers, excluding Molmo, are <u>underlined</u>. We categorize models into five groups: (first) API-only, (second) open-weight, (third) open-weight and open-data, (four) the Molmo family and (five) the specialized clock reading model.

model	precision	recall	F_1
MolmoE-1B	73.0	72.9	72.2
Molmo-7B-O	75.7	75.5	75.1
Molmo-7B-D	75.0	74.6	74.3
Molmo-7B-D (36 crops)	58.4	58.7	58.1
Molmo-72B	75.8	75.4	75.2

Table 11. **Pointing evaluation results**. Pointing can perform poorly when # of crops are unequal at training and test time.

point with an assignment is a true positive or false positive. Specifically, we calculate precision as the fraction of predicted points located within the segmentation mask of their assigned ground-truth point, and recall as the fraction of segmentation masks covered by predicted points.

Table 11 demonstrates Molmo's superior pointing capability. Similar to captioning and counting, pointing performance declines when the number of crops are unequal at training and test time.

High-resolution fine-tuning. In Table 2b of the main paper, we showed that training the model with higher resolution (*i.e.*, more image crops) yields slight improvements on the 11-avg metric (from 76.9 to 77.2 when increasing the number of crops used in training from 12 to 36). Rather than directly training at a higher resolution, we explore fine-tuning the model initially trained with 12 crops using a higher resolution. Specifically, we continue training the 12-crop model for 3000 additional steps (10% of the fine-

⁷https://www.imdb.com/title/tt2008009

⁸https://github.com/charigyang/itsabouttime

# crops train, test	CountBenchQA	Pixmo-Count val	11-avg
12, 36*	87.7	73.9	75.8
12, 36	88.5	85.2	76.9
36, 36	88.9	87.4	77.2
$12 \rightarrow 36, 36$	88.9	87.4	77.2

Table 12. **High-resolution fine-tuning results.** Result on counting datasets (CountBenchQA and PixMo-Count val set) and the overall average (11-avg) using different numbers of crops at train and test. Note: 12, 36* uses a higher number of crops (36 crops) at test time for counting datasets, which leads to a much worse accuracy. Our default setting (highlighted in gray) uses the same number of crops (12 crops) during training and inference for counting datasets. We experiment with fine-tuning the 12-crop model at higher resolution and evaluating with 36 crops (12 \rightarrow 36, 36).

tuning steps) with 36 crops, roughly halving the learning rates of the vision encoder (lr=2e-6), connector (lr=2e-6), and language model (lr=5e-6). We keep the global batch size at 256 and use a warmup of 200 steps for all modules.

Table 12 presents the results. Note that simply increasing the number of crops at inference time (first row) leads to degraded performance on counting tasks (88.5 \rightarrow 87.7 for CountBenchQA and 85.2 \rightarrow 73.9 for PixMo-Count). This suggests that a mismatch between training and testing resolutions adversely affects counting performance. As a result, our default model (second row) uses the same number of crops (12 crops) for counting datasets.

After fine-tuning the model with higher resolution (fourth row), we observe that its counting performance can recover when evaluated with 36 crops, matching that of the model trained directly with 36 crops (third row), without sacrificing the overall 11-avg performance. This demonstrates that a brief period of high-resolution fine-tuning can effectively restore counting capabilities without affecting the average performance.

Text-only benchmarks. PixMo consists exclusively of multimodal image-text data, without any text-only data. To investigate the potential impact of training solely on multimodal data on performance in text-only tasks, we report the results on common text benchmarks which assess a wide range of capabilities. We carefully follow the setup used by Llama 3 [5] for each task, ensuring that we can reproduce their numbers within the reported confidence intervals. As shown in Table 13, the Qwen2 language model employed in Molmo-7B-D appears to lose some knowledge across various tasks as a result of multimodal fine-tuning.

We run a small experiment at the 7B scale to test whether adding text-only data from Tulu 3^9 [113] to our fine-tuning data mixture can address this issue. We use two different ratios: the entire dataset and a version with 10% downsampling. Incorporating this text data enhances model per-



Figure 8. Human evaluation outcomes for matches between various models vs. Molmo-7B-D. We expand upon the win rate (excluding ties) shown in Table 5 to report the full breakdown of wins, losses, ties (both good), and ties (both bad). We removed the *I don't know* responses, which accounted for 2.9% of all human feedback, before calculating the outcome rates.

formance on text-only tasks, particularly those involving mathematical reasoning and programming. Interestingly, down-sampling to 10% of the data leads to better results on most text-only tasks and improves the average performance across the 11 multimodal academic benchmarks.

Human evaluation. Table 5 of the main paper reports the win rates of several ablation and API-only models *vs*. Molmo-7B-D *when ties are excluded* (a standard metric reported in the LMSYS Chatbot Arena). Ties make up a significant portion of the matches, so we report the full breakdown of match outcomes in Figure 8 to better characterize the human evaluation. For example, when paired against Claude-3.5 Sonnet 45.5% of matches resulted in a tie where both responses were good, 14.1% in a tie where both responses were bad, Claude won 26.1% of the time, and Molmo-7B-D won 14.3% of the time. As a second example, compared to Molmo fine-tuned only on academic data the breakdown is 28.4% (ties, both good), 22.1% (ties, both bad), 8.4% (it wins), 41.1% (Molmo-7B-D wins).

AI2D with opaque boxes. Table 14 shows result with and without opaque boxes on AI2D. The two options are described and discussed in Section **B**.2.

Cap F_1 **and 11-avg correlation.** For the majority of the project we did not look at downstream tasks¹⁰ and instead made most modeling decisions to maximize our captioning metric. At the conclusion of the project, we used our ablation experiments to analyze the relationship between cap F_1 and the 11 benchmark average (11-avg), shown in Figure 9. The scatter plot includes results from the 22 experiments that meet two conditions: (1) the experiment affects pretraining and (2) the experiment uses PixMo-Cap. We exclude a small number of experiments that use different pretraining data, *e.g.* ShareGPT4o/v, because cap F_1 becomes an out-of-domain evaluation that is not directly compara-

⁹https://allenai.org/papers/tulu-3-report.pdf

¹⁰We did a small number of sanity checks on VQA v2.0.

model	MMLU [38]	MMLU-Pro [114]	GSM-8k [24]	MATH [39]	ARC-C [23]	HumanEval [16]	11-avg
Qwen2-7B (language model)	70.2	42.1	71.8	40.3	87.5	47.6	-
Molmo-7B-D	64.6	32.2	58.8	11.5	81.5	36.6	76.9
Molmo-7B-D + Tulu 3	64.9	38.6	67.7	8.3	84.5	51.2	76.9
Molmo-7B-D + Tulu 3 \times 0.1 down-sample	65.4	37.3	71.2	27.5	84.9	55.5	77.1

Table 13. Text-only benchmark results. 11-avg denotes the average performance on 11 academic benchmarks.

model	opaque	transparent
MolmoE-1B	75.7	86.4
Molmo-7B-O	79.8	90.7
Molmo-7B-D	82.4	93.2
Molmo-72B	86.4	96.3

Table 14. AI2D test scores with transparent and opaque boxes.



Figure 9. **Relationship between cap** F_1 and 11-avg. Our model development was driven by increasing cap F_1 . Here, we show a scatter plot of cap F_1 vs. the 11 benchmark average (11-avg) from 22 ablation experiments, including all ablations that: (1) affect pre-training and (2) use PixMo-Cap. The Pearson correlation (ρ) is 0.82 and a least-squares regression line is shown in red.

ble to in-domain results. We observe a strong correlation (Pearson $\rho = 0.82$), suggesting that optimizing for dense captioning may be a reasonable proxy for a broad range of downstream tasks—though we have not established a causal relationship and this suggestion should be taken with that caveat in mind.

Leaderboards. We submitted Molmo-72B-D to several leaderboards. Molmo-72B-D achieves first on the VQA v2.0 leaderboard on the A-OKVQA leaderboard, achieves third on DocQA and InfoQA, behind QwenVL-72B, InternVL2-Pro. ¹¹

E. Ablations Details

E.1. Discussion of Main Paper Ablations

Vision encoder. We adopted OpenAI's CLIP early on and used it for our main results and as the default in our ablations. Later, we evaluated the three alternative choices in Table 2a. All encoders are ViT-L/14 with 336×336 pixel inputs, except for SigLIP which uses 384×384 pixels. For MetaCLIP, we started with the weights of the 224×224

model and resized the positional embeddings to 336×336 before using it in Molmo. To equalize computation, we slightly reduced the maximum number of crops for SigLIP so that the average vision token count is similar for all models. Overall, the three encoders that were trained on webscale noisy image-text data perform very similar to each other on both metrics. Of significant note, this includes MetaCLIP which is a *fully* open model (data and weights) meaning that *every model component and every bit of data in a Molmo model equipped with MetaCLIP and OLMo is open*. In retrospect, we should have used MetaCLIP as our default vision encoder, but we evaluated it too late in the process to retrain all Molmo models and ablations that were already based on OpenAI's CLIP vision encoder.

Also, surprisingly, when using the DINOv2 backbone which is trained on images only (no text, no label supervision)—Molmo performs only slightly worse than the vision-language supervised vision encoders. DINOv2 also performs well in our user study (Table 5), with a win-rate (excluding ties) of 45% compared to our standard Molmo-7B-D configuration (*i.e.*, Molmo-7B-D wins 55% of all non-tie matches against its DINOv2-based variant).

Image resolution. Table 2b shows that using more crops (and thus higher image resolution) for training and testing generally improves results. We found that some tasks, like document-heavy ones, even benefit from using more crops in inference than the number used for training. However, for captioning and pointing (and thus counting), results degrade when the number of test crops does not equal the number of training crops. Therefore, for captioning and counting tasks we always force these values to be the same. As shown in Table 12 this awkward detail can be remediated by a small amount of high-resolution fine-tuning and then always using that same number of crops during inference for all tasks.

Dropout. Dropout in the LLM generally improves both pre-training and fine-tuning (Table 2c). We also find that our novel text-token-only dropout, in which dropout is only applied to the text tokens of the caption, not to the vision or prompt tokens, improves the captioning metric. We hypothesize that this restricted dropout encourages the model to rely more on the vision tokens, rather than guess based on the previous text tokens, when generating tokens which may reduce hallucinations.

Length conditioning. Our captioning pre-training task includes a length hint. In Table 2e we ablate this design choice and find that it significantly impacts the captioning metric,

¹¹Results as of Nov. 21, 2024.

but also improves the downstream tasks. Note that length conditioning only changes the pre-training task; the fact that it improves the downstream metrics indicates that captioning with length conditioning is a better pre-training task than just captioning.

PixMo-Cap scaling. In Table 3a we show the scaling effects of PixMo-Cap data by training with smaller fractions of the data in both pre-training and as part of the fine-tuning data mixture. Both metrics clearly improve as the amount of captioning data varies from none at all to the full set of 712k images. We also tested the model with no PixMo-Cap data in our user study (Table 5), where it had a winrate (excluding ties) of only 35% compared to our standard Molmo-7B-D configuration. Removing PixMo-Cap data has a severe negative impact on its user preference score.

Pre-training data. We consider different choices of pretraining data in Table 3b. Pre-training VLMs, not just the vision encoder, with web-scale noisy image-text data is a popular data choice in contemporary methods (*e.g.*, [5, 10]). We test if this has any advantage using data from LAION [98]. To do this we add a preliminary training stage that tunes the model for 50k steps and a batch size of 1024 on image/text pairs from LAION 2B. In this stage only the V/L connector is tuned, the LLM and image encoder are frozen. This pre-trained model is then trained on the dense captions and then our instruction tuning mixture as normal. We find no improvement in metrics using this strategy, allowing us to keep the training pipeline simple.

Another popular choice is to use ShareGPT4V/o [15], which involves distilling from GPT-4 through captions. Using this data instead of PixMo-Cap performs worse on both metrics even when approximately controlling for the data scale (compare to 178k PixMo-Cap images in Table 3a. In contrast, if we caption all PixMo-Cap images with GPT-4o and train on those captions, both metrics perform strongly. We think this is likely because PixMo-Cap has a more diverse image distribution, and due to captioning improvements in GPT-4o. Finally, we compare our default setting to either using only the raw audio transcripts or only using the LLM cleaned transcripts, both of which perform slightly worse than our default strategy of using both.

Supervised fine-tuning data. We explore choices of finetuning data in Table 3c. Using only academic data sets (specifically the ones in Fig. 4, but excluding Android-Control) performs significantly worse than our full mixture (72.2% vs. 76.8%). The gap is primarily explained by PixMo-Docs, which improves results on document-heavy tasks, and the counting data from PixMo-Points and PixMo-Count. The other fine-tuning PixMo datasets have a small, and sometimes slightly negative, impact on the 11 benchmarks; they primarily add new skills to the model and improve user experience when chatting with it, as shown by user preference scores in Table 5.



Figure 10. **PixMo-Points distribution of counts.** We show the number of pointing questions (on a log scale) with answers in different ranges (*e.g.*, 1 to 10, 11 to 20, *etc.*).

Counting. In Table 4d we compare encoding points in plain-text as numbers between 0.0 and 100.0 with one significant digit of precision (our default) *vs.* adding 1000 special point tokens to the model's tokenizer, maintaining the same spatial precision. We find that using special point tokens performs substantially worse than the simple plain-text representation.

E.2. Additional Ablations

Additional model ablations are presented in Table 15 for vision encoder layers, learning rate warmup, and gradient normalization. See table captions for more details.

F. Data Details

PixMo-Points. The PixMo-Points dataset has a total of 229k unique images and a total of 1.98M referring expressions. It has an average of 8.7 distinct expressions per image with an average 5.5 points per expression, and an average of 47.7 total points per image. Additionally, there are 359k instances with no target object (no points). Figure 10 shows the distribution of number of points for expressions with non-zero points. PixMo-Points is a much larger and more diverse dataset than previous works such as gRef-COCO [66] (which contains a total of 20k images, 60k distinct instances, 278k expressions, of which 80k are multitarget and 32k are no-target expressions) and also much larger than RefCOCO, RefCOCOg and RefCOCO+ [126], each with about 86k, 142k and 141k unique referring expressions respectively and no multi-target references. Additionally, PixMo-Points focuses on referring to points and not segmentation masks, making it significantly more efficient to collect.

PixMo-Cap. We prompted our annotators with the following questions to answer in their spoken image descriptions.

layers	cap F_1	11-avg	steps (ViT / con. / LLM)	cap F_1	11-avg	grad norm	cap F_1	11-avg
3rd-to-last & 10th-to-last	54.1	76.9	2000 / 200 / 2000	54.1	76.9	component-wise	54.1	76.9
only 3rd-to-last	53.7	76.6	200 / 200 / 200	53.7	76.9	global	53.6	76.9
only 10th-to-last	52.5	76.3				global, fine-tune only	54.1	76.9
(a) Vision encoder layers. Concatenating		(b) Learning rate warmu	(b) Learning rate warmup . The number of (c			t normali	ization ap-	
features from multiple ViT	layers (near the	learning rate warmup step	s used in	caption	plied separately to the V	/iT, conn	ector, and

features from multiple ViT layers (near the end and towards the middle) *vs.* using features only from each of those layers.

(b) **Learning rate warmup**. The number of learning rate warmup steps used in caption pre-training for the three principal model components: ViT, connector, and LLM.

Table 15. Additional model ablations. Defaults are in gray.

- 1. What is the image at first glance?
- 2. What are the objects and their counts?
- 3. What does the text say?
- 4. What are the positions of the objects?
- 5. What subtle details are noticeable?
- 6. What is in the background?
- 7. What is the style and color?

PixMo-Docs. We developed a generation framework for synthesizing text- and figure-heavy images. The core idea is to harness the coding capabilities of a text-only LLM to generate programs that render image data. These programs are then used as context for another LLM to construct instruction-tuning datasets.

Our framework supports seven programming languages/rendering libraries, including Matplotlib, Plotly, La-TeX, HTML, Vega-Lite, Mermaid, and Graphviz. Using these tools, we designed specialized pipelines to generate charts, tables, diagrams, and various types of documents.

The framework accepts text input to control the generation process. For instance, given the input "restaurant menu", the system selects the appropriate tools to generate relevant data. To diversify the final datasets, we use a comprehensive set of input queries. Additionally, we enhance data diversity by incorporating personas [35], which control the content and style of the synthetic data. For example, when generating "restaurant menu" data with the persona "A barbecue enthusiast known for their amazing grilled food at every Tennessee Vols game", the framework produces a data point featuring a "Southern fusion menu combining traditional BBQ with international flavors, presented on a wooden board background". This approach allows us to enrich the variety within each category of synthetic data.

We use Claude-3.5 Sonnet [7] for code generation and GPT-40-mini [89] during the instruction-tuning data generation stage, prioritizing cost efficiency.

G. Dataset Examples

We include **randomly selected** examples from the **PixMo**-* datasets. Prompts are shown in bold, and points are shown with pink dots.

- PixMo-Cap Figure 12
- PixMo-AskModelAnything- Figure 13

- PixMo-Points Figure 14
- PixMo-Points with explanations Figure 15

LLM parameters, globally to all parameters

(row 2), or component-wise for pre-training and globally for fine-tuning (row 3).

- PixMo-CapQA Figure 16
- PixMo-Clocks Figure 17
- PixMo-Count Figure 18
- PixMo-Docs (charts) Figure 19
- PixMo-Docs (tables) Figure 20
- PixMo-Docs (diagrams) Figure 21
- PixMo-Docs (other) Figure 22

H. Related Work

Vision-language contrastive models. Vision-language models have become popular in the last few years. Models such as CLIP [96] and ALIGN [43] that are trained on noisy web data provide strong language-aligned image encoders and perform well on downstream classification and image-text retrieval tasks, without any task specific tuning. Previous works [33, 115] proposed similar ideas before transformers [109] became popular. Since CLIP was released, other works have focused on making the CLIP pipeline fully open [20, 118]. However, vision encoders trained with noisy web data have limitations in discerning details, as discussed in [107].

Multimodal LLMs. Multimodal LLMs often use CLIPstyle image encoders and align image embeddings with the LLM input space via a connector module [26, 41, 59, 69, 71, 85, 92, 105, 112, 131]. Some works have also explored using multiple image encoders in tandem with CLIP-style encoders [74, 106], such as using self-supervised learning (SSL) encoders [91]. Many works use a pre-training stage for just the connector weights [12, 34, 69, 106] while others do not have an explicit connector training stage [10, 18, 48, 67]. In contrast, two other common architecture strategies are (1) directly connecting the image embedding to different LLM layer embeddings via crossattention [5, 6, 58, 134] and (2) removing the image encoder and directly inputting the pixels [9, 56]. The cross-attention design naturally allows for the introduction of a large number of new parameters, which enables freezing the LLM while still training an effective VLM. This approach has the advantage of maintaining text-only task performance (cf. Table 13). Due to the compute constraints for training and inference of these models, there has also been a rise in effi-

			VLM	LLM Backbone		Vision Encoder	
Category	Model	Open Weights	Open Data + Code	Open Weights	Open Data + Code	Open Weights	Open Data + Code
	Molmo-72B	Open	Open	Open	Closed	Open	Closed
Molmo	Molmo-7B-D	Open	Open	Open	Closed	Open	Closed
	Molmo-7B-O	Open	Open	Open	Open	Open	Closed
	MolmoE-1B	Open	Open	Open	Open	Open	Closed
	GPT-40	Closed	Closed	Closed	Closed	Closed	Closed
	GPT-4V	Closed	Closed	Closed	Closed	Closed	Closed
	Gemini 1.5 Pro	Closed	Closed	Closed	Closed	Closed	Closed
API Models	Gemini 1.5 Flash	Closed	Closed	Closed	Closed	Closed	Closed
	Claude 3.5 Sonnet	Closed	Closed	Closed	Closed	Closed	Closed
	Claude 3 Opus	Closed	Closed	Closed	Closed	Closed	Closed
	Claude 3 Haiku	Closed	Closed	Closed	Closed	Closed	Closed
	Qwen VL2 72B	Open	Closed	Open	Closed	Open	Closed
	Qwen VL2 7B	Open	Closed	Open	Closed	Open	Closed
	Intern VL2 LLAMA 76B	Open	Closed	Open	Closed	Open	Closed
Open Weights	Intern VL2 8B	Open	Closed	Open	Closed	Open	Closed
	Pixtral 12B	Open	Closed	Open	Closed	Open	Closed
	Phi3.5-Vision 4B	Open	Closed	Open	Closed	Open	Closed
	PaliGemma 3B	Open	Closed	Open	Closed	Open	Closed
	LLAVA OneVision 72B	Open	Distilled	Open	Closed	Open	Closed
	LLAVA OneVision 7B	Open	Distilled	Open	Closed	Open	Closed
	Cambrian-134B	Open	Distilled	Open	Closed	Open	Closed
Open Weights & Data	Cambrian-18B	Open	Distilled	Open	Closed	Open	Closed
	xGen - MM - Interleave 4B	Open	Distilled	Open	Closed	Open	Closed
	LLAVA-1.5 13B	Open	Open	Open	Closed	Open	Closed
	LLAVA-1.5 7B	Open	Open	Open	Closed	Open	Closed

Figure 11. VLM Openness Comparison. We characterize the openness of VLMs based on two attributes (open weights, open data and code) across three model components (the VLM and its two pre-trained components, the LLM backbone and the vision encoder). In addition to open vs. closed, we use the "distilled" label to indicate that the data used to train the VLM includes images and text generated by a different, proprietary VLM, meaning that the model cannot be reproduced without a dependency on the proprietary VLM.

cient multimodal LLMs [14, 22, 64, 79, 123, 128, 137].

The best performing multimodal LLMs [7, 90, 103] are proprietary closed source models. While they are very capable, not much is known about how these models are trained and what data they use. In contrast, many works release their model weights [1, 3, 5, 10, 111] but don't release their training recipes or don't disclose all the data used. Other works provide all the training details and data [54, 59, 106, 119, 133], but use data generated by proprietary VLMs such as [15]. Hence, there is a need for a

fully open SoTA training pipeline that does not use previously trained multimodal LLMs to generate data.

Vision-language instruction tuning datasets. The rise in popularity of VLMs has also led to a rise of methods to build visual instruction-tuning data. A common approach is to annotate an image with vision models (or use ground-truth annotations), and then use a LLM to generate QA pairs [57, 69, 125, 137] from those annotations. However, these approaches are limited since the automatically generated annotations can be noisy, and even ground-truth an-

notations often do not comprehensively describe all the details in the image. PixMo-CapQA takes a similar approach but uses the detailed captions from PixMo-Cap which provide more comprehensive image descriptions. Many recent methods have used proprietary VLMs to annotate images directly [13, 15, 68, 71, 110], which is effective but makes the training pipeline dependent on a closed source VLM.

It is also very common to pair templated instructions with existing annotated datasets to build instruction tuning data (*e.g.*, [5, 44, 57, 75]). While Molmo also uses academic datasets, we prefer style tags over natural language instructions since we believe our data, and in particular PixMo-AskModelAnything, provides better training for conversational user interactions.

Our approach to having annotators work with a LLM when generating QA pairs is similar to the approach in [80], but we extend this idea to image/language data.

Synthetic vision-language datasets. Prior approaches to synthetic chart generation typically only support one or two types of charts [46, 47, 86], often with a heavy focus on bar charts or line plots. PixMo-Docs uses code as the text-only representation for the LLM which lets us support much more diverse formats, including heat-maps, violin plots, chord diagrams, geographic plots, tree maps among many others. Our use of HTML for document generation is also similar to [55], however we consider many additional approaches to representing documents besides HTML.

Synthetic clock data has been considered [121]. Our approach uses real watch faces instead of rendering clocks purely from a simulator, which gives our synthetic data more diversity (*e.g.*, watches with no second hand, stylized decorations or coloring, background images, a separate inner piece to show seconds). Combining these two datasets might yield additional improvements.

VLM grounding. Multimodal LLMs that support grounding language in an image are becoming more common [61, 92, 94, 97, 117, 127, 132]. These works commonly use automated object detectors and/or existing referring expression datasets [52, 66, 126] for training data. Of these datasets, GRES [66] is most similar to PixMo-Points in that it is human-annotated and includes arbitrary expressions (not just object categories), none-present annotations, and allows expressions to refer to multiple object in the image. However, it only grounds a limited category of objects (e.g., only COCO categories), and rarely grounds expressions in a large numbers of objects in a single image. For PixMo-Points we source a diverse set of images and collect points from human annotators. We annotate points instead of segmentation masks, which enables us to collect 1.98M unique referring expression instances with an average of 5.5 points per expression.

Bootstrapping from LLMs. Closed text-only LLMs are commonly used for data generation and curation [69]. We

also made use of closed text-only LLMs when building several of the PixMo datasets. Given our stance against using *VLMs* for building datasets, it's worth justifying the use of closed LLMs. It is true using closed LLMs means the current data pipeline is not entirely open. However, once *open* LLMs (*e.g.*, [37]) become sufficiently good they can be used in place of closed ones to build a dataset functionally equivalent to PixMo. Our philosophy is that we should not wait for open LLM research to achieve this goal and instead we should pursue research on building open VLMs *in parallel*. We note that using one VLM to build another VLM is entirely different than using an LLM, because the dependency is circular and therefore cannot result in a fully open system at a later point in time.

Generate a long caption about this image.



This black and white sketch depicts a boat, elegantly stationed at a pier. The boat, seemingly wooden, showcases a classic design with four prominent sails catching an imagined breeze. Below the boat text inscribed reads "TAPTANA" in clear, uppercase letters. In the lower right-hand corner, another text, written in cursive and appearing less legible, adds an air of authenticity to the artwork. The meticulous detail in the sails and the overall structure of the boat highlights the craftsmanship and artistry involved in this sketch.

This detailed color photograph captures an empty room within a house. The room's walls are painted a bright, dusty green, overlaid with a lighter green paint that is notably peeling and chipping away, revealing the layer beneath. Dominating one wall is a window divided into four rows of three panes each, making a total of 12 galas panes. The window frame and windowsill are painted a dark black, adding a stark contrast to the errore wall:

On the left side of the image, an open doorway reveals a white-painted door with its own peeling paint. Beyond the doorway, darkness obscur.

Screenshot Description: Mobile Application Interface



Create a detailed caption

This photograph, taken at night, captures the striking image of a hotel called Breeze Suites. The sky is pitch black, with no stars or a moon visible, enveloping the surrounding area in total darkness. In the middle ground, faint outlines of trees are visible in front of the building. The hotel itself appears to be five or six stories tall and is brightly illuminated with yellow and orange lights, highlighting its structure. At the very top of the building, a prominent neor ned sign isophasy the name "Breeze Suites," accompanied by a wave-shaped logo. The façade features crisscross light patterns, adding to the building's visual appeal. A set of staris is visible in the central breezeway, and although the hotel is well-lit, the details of the trees and surroundings remain indistinct due to the envelopine darkness. indistinct due to the enveloping darkness.

describe the image

A screenshot captured from an Android phone displays a Twitter.com interface, taken when the platform was still known as Twitter, prior to its rebranding as X. The user's device, likely a customized Android given the minimal status indicators at the top of the screen, shows the time as 3:04. The status bar reveals a strong Wi-Fi connection, indicated by the full Wi-Fi signal icon, alongside a charging and fully charged battery icon. The mobile data shows two bars of signal strength, with the GPS icon activated.

The screenshot focuses on the Twitter profile of Stu Peters (@realStuPeters). The user is not following this account, which has made a total of approximately 10,000 tweets. The visible tweet from Stu Peters, dated March 28th, is a reply to David Pakman...

Write a long caption

This image is a triptych featuring three detailed shots of pizzas from Catfishman's Pizzeria in Calgary, AB, established in 2018. The top, landscape-oriented photo showcases the pizzeria's whimscal box design, adorned with a cartoon catfish character donning a white chef's hat, a chef's outfit, and a red bandana.

The two images below provide closer looks at the pizzeria's offerings. On the bottom left, a close-up reveals an uncooked, cheese-loaded pizza that nearly overflows a plastic cutting board set on a brown surface. The pizza is generously coated in cheese and tomato sauce, unadorned by other toppings, ready to be sliced and served.



« O

Front Olanez, Melcarnel Pour room In rest manly at this time, Mr Sand an update in 60 minutes-or investidately when 8 becomes available. Text 1500 Micro

Frant Olivery: Your scient is util being prepared, We appreciate your patience! War will notify you as seen as your room is available. Next SPOP

Presi Oloney (1 of 2) Valuement Nu-room is ready, from norm number in 1536 tet Floor Congress Park. Please and the David to Room

**Status Bar Details **

Write a long caption

Create a detailed caption

walls ne gi

*Status Bar Detaus:** Time: 41:17 bit is displayed in the top-left corner. Cellular Signal: The phone shows 2 out of 4 cellular signal bars. Wi-Fi Signal: The Wi-Fi signal is weak, with 1 out of 3 bars. Battery Indicator: Although the battery percentage is not displayed, t...

The screenshot is of a mobile application on an iPhone, illustrating the messaging interface.





in this nonzontauly aligned landscape photograph, taken from a first-person point of view, a heavyset man in a dark blue, savdust-covered T-shirt is holding a bright orange and black weed eater. The image, taken with a GoPro camera, shows the man gripping the top handle of the weed eater with his right hand and a secondary handle with his left. His stomach protrudes slightly under the shirt. To the right of the image, there is a green bucket with a gray object inside it, standing on smoother concrete, indicative of a garage or workshop area which has a layer of dust or sawdust on the floor. In the.

describe this

This image captures a bustling downtown scene under a clear, sunlit sky, Central to the This image captures a bustling downtown scene under a clear, sunlit sky. Central to the composition is a large, triangular billboard structure, completely devoid of any advertisements, revealing blank white surfaces on all visible panels. The billboard is equipped with numerous lights positioned on poles above it, presumably to illuminate it at night. Surrounding the billboard are several prominent buildings, notable among them a towering red brick structure on the right, adorned with rows of windows and fire escape ladders. A water tower perches on one of the rooftops, adding to the urban atmosphere. To the left, a tall white building with additional ladders or metal structures is visible. The upper left corner of the image features a yellow crane, its pulleys and chains slightly extending into the frame. The entire billboard is encased in a metal fen...

image

Figure 12. Randomly selected examples from PixMo-Cap with our prompt templates.



SPATTER SCS

RESERVA YA STATES



The topping isn't cheese as ..

what kind of back brake is on this motorcycle?

The motorcycle in this image is equipped with a rear drum brake. You can see the drum brake assembly attached to the hub of the rear wheel. Drum brakes are an older brake technology that's less common on modern high-performance motorcycles, but they're still...

answer with text only, do not points

points how many times is the number 1 used in the photo

The number 1 appears six times in the photo. It's used in various parts of the contact information

displayed, such as phone numbers and ID codes. These

occurrences of the number 1 are distributed across different



AABBEEIRRSTT

Rabbit" in alphabetical order, with

what is in this image

trees, as well as trees with autumn colored leaves in shades of orange and yellow. The ground is covered in green grass and dotted with colorful flowers, particularly red and orange blooms. A small pond with lily pads adds to the...

Do not point How many blades this has?



This is a striking mural painted on the side of a building in Houston, Texas. It's a collaborative piece created by two renowned local grafifti artists, Daniel Anguilu and ACK!. The mural showcase a surreal and emotionally evocative scene

The artwork reflects the unique styles of both artists. Daniel Anguilu, known for ..

Figure 13. Randomly selected examples from PixMo-AskModelAnything.

Can you take the phrase East Rabbit and put all the letters alphabetically?

Certainly! Let's take the phrase "Easter Rabbit" and arrange its letters alphabetically:

This includes all the letters from "Easter

This image appears to be an AIgenerated fantasy forest scene. It depicts a lush, vibrant forest with various types of trees, including bamboo and palm



En l

The background of the image features a window with blurry white curtains visible on one side. Sunlight appears to be streaming in through the window, streaming in unrough the window, creating a bright and airy atmosphere. The background is intentionally out of focus, which helps draw attention to the main subject of the image in the foreground. This soft, blurred background provides a pleasant and ..

what is in the background?

what kind of car is shown in the

The car shown in the image is an Audi R8. This high-performance sports car is known for its sleek and distinctive design. The Audi R8 features a mid-

engine layout and is recognized for its

impressive performance capabilities. It's a popular choice among sports car enthusiasts and has been featured in

various racing events and competitions.

what is this



In this horizontally aligned landscape photograph, taken from a first-



Figure 14. Randomly selected examples from **PixMo-Points**. Even when text has been cut off, all points still appear in the image. Our templated prompts can be ungrammatical for some of these options, but we find they are still sufficient to let the model respond correctly to natural language instructions.



Figure 15. Randomly selected examples from the experimental **PixMo-Points** data that includes points with explanations.



Figure 16. Randomly selected examples from the synthetic PixMo-CapQA data generated from PixMo-Cap.



Figure 17. Randomly selected examples from the synthetic PixMo-Clocks data after our data augmentation.

Generate list of points showing where the people are and output the total count. Counting the epoints $x_1^{-1}5.5^{-y}y_1^{-1}52.1^{+x}2^{-2y}6.8^{-y}2^{-2y}54.3^{+y}2^{-2y}5.2^{+y}3^{-2y}2^{-2y}5.2^{+y}3^{-2y}$		Show me where a people is <pre>cpoint x="53.5" y="50.2" alt="people">people</pre>		How many people are there? Counting the cpoint x=*44.3* y==71.6° alt="people">peopletherefore the second sec	find any dogs in the picture epoints x1=#46.5° y1=#2.7° z2=707 9° z=*13? alt="dogs">dogs
Count the people. Conting the cpoints x_1^{-21} 9° y_1^{-6} 65.3° x_2^{-4} 7.5° y_2^{-6} 6.9° x_3^{-6} 8° y_1^{-6} 6.5° x_4^{-7} 7.3° y_4^{-6} 6.5° x_5^{-7} 7.1° y_5^{-6} 6.8° alt="people" speepleshows a total of 5.	1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	If you see any horses in the image, give me the count. Otherwise, says 'This isn' in the image.' Counting the -qooints x1=12.9' y1=17.0' x2=13.0' y2=43.0' x1=33.0' y2=43.0' x4=37.2' y4=15.6' x6=37.9' y6=54.2' x6=37.9' y6=68.6'	\$5 %	Can you point out all people in this image? How many are there? Counting the <points x1=14.7 y1=48.6* x2=22.5 y2=49.5* x3=410 y3=50.6* x4=48.6* y4=54.2* x5=57.3* y5=58.8* x6=74.3* y6=51.7* alt="people">people<td>Can you point out all people in this image? $z^{-37,77} = 25.7^{\circ} z^{-57,77} = 52.7^{\circ} z^{-37,77} = 52.0^{\circ} z^{-37,77} = 52.0^{\circ} z^{-57,07} = 4z^{-50,07} = 4z^{-50,07} = 4z^{-57,07} = 52.1^{\circ} z^{-57,07} = 52.1^{\circ}$</td></points 	Can you point out all people in this image? $z^{-37,77} = 25.7^{\circ} z^{-57,77} = 52.7^{\circ} z^{-37,77} = 52.0^{\circ} z^{-37,77} = 52.0^{\circ} z^{-57,07} = 4z^{-50,07} = 4z^{-50,07} = 4z^{-57,07} = 52.1^{\circ} z^{-57,07} = 52.1^{\circ} $

Figure 18. Randomly selected examples from the synthetic PixMo-Count data.



Figure 19. Randomly selected chart examples from the synthetic PixMo-Docs data.



Figure 20. Randomly selected table examples from the synthetic PixMo-Docs data.



Figure 21. Randomly selected diagram examples from the synthetic PixMo-Docs data.



Figure 22. Other randomly selected documents from the synthetic PixMo-Docs data.