
ISAAC-0.1: A SIMPLE PATH TO THE EFFICIENT FRONTIER OF PERCEPTION



ABSTRACT

We introduce Isaac-0.1, a 2B-parameter, open-weight vision-language model which excels at grounding. Isaac-0.1 couples a simple training recipe with a unified, text-native grounding interface that supports points, boxes, and polygons, enabling conversational pointing in which answers are aligned with cited visual regions. Despite its small size, Isaac-0.1 attains strong results on pointing, OCR, and visual question answering. Designed for the perception-efficient frontier, it delivers competitive capability at practical latency and cost suitable for deployment.



Q: Count the number of crosses on top of the domes.

A : There are **two crosses** **on the domes** **of the church**. One is at the top of the main dome, and the other is on the smaller dome to the left.



Q: Determine whether it is safe to cross the street. Look for signage and moving traffic.

A : No, it is not safe to cross the street at this time. The **traffic light shows a red signal**, and there is **moving traffic**; wait until the signal permits crossing.



Q: What price are the cucumbers and the lettuce? Output in JSON.

A : Here's the price information in JSON:

```
{  
    "cucumbers": "1.27",  
    "lettuce": "1.67"  
}
```

This JSON captures the price of **cucumbers** as \$1.27 and **lettuce** as \$1.67.



Q: Was a shot taken before the buzzer? Provide a one sentence rationale.

A : Yes. The **shot clock shows 1.5s left** and the **ball is in the air**, indicating the shot preceded the buzzer.

1 INTRODUCTION

The recipes to build large language models are converging to a set of known and scalable recipes, however, perceptive language models have required greater complexity to achieve frontier performance. Often this requires larger architectures [1], several training stages [2], or fine-tuning [3]. Additionally the current state of perceptive training centers around text generation working for traditional question answering tasks but struggling on perceptive tasks such as grounding, OCR, and counting.

In this technical report we present Isaac-0.1 a 2B-parameter dense model that matches or outperforms the state-of-the-art at comparable sizes in perception-centric tasks, released as open weights^{1,2} under an open license (CC BY-NC 4.0). Isaac-0.1 is a VLM consisting of a vision encoder initialized with SigLIP-2 [4], an LLM backbone initialized from Qwen3 [5], and was trained with a simple and efficient recipe. Isaac-0.1 optimizes for flexibility and efficiency by preserving native resolution [6] and using a lightweight connector between the vision encoder and the model backbone.

Training Isaac-0.1 has been designed around ensuring the right abstractions and primitives to simplify and optimize multimodal training. We introduce a novel data abstraction *TensorStream* [7] - a flexible data type enabling planning, packing, and collating multimodal input. Furthermore, heterogeneity in batch composition across modalities leads to uneven load across ranks. To account for this, we leverage a multimodal load balancing stage to optimize for uniform load distribution at each step. Our recipe consists of a single mid-training stage on 100 billion tokens and a light weight post-training stage to introduce a conversational grounding interface we term *conversational pointing*.

We address the short comings of existing VLMs by introducing a unified interface for pointing and grounding - this interface enables Isaac-0.1 to visually localize a mention encompassing tasks such as object detection, segmentation, referring expressions, and grounding into a single interface. We carefully design a serialization and tokenization scheme for autoregressive language modeling. We further prove this through a novel interface termed *conversational pointing* - allowing our model to point as a visual citation to any user request.

Isaac-0.1 outperforms strong baselines on grounding tasks and matches existing models across perception benchmarks at similar scale. In parallel, we do infrastructure "right": zero-padding training via document masking and sequence packing, zero-NFS streaming data access, and input load balancing to minimize variance and maximize MFU (Model Flops Utilization).

To summarize our contributions

- We release **Isaac-0.1** — an efficient 2B model which matches existing open and closed source comparable-scale models on multimodal benchmarks and excels at grounding.
- We showcase that simple training recipes can reach the frontier.
- We prove that our model is a good base for further training by post-training it to do conversational pointing.

2 INFRASTRUCTURE

Training language models efficiently across modalities represents a fundamental challenge in modern machine learning infrastructure due to heterogeneous batches, dynamic shapes, and padding overheads. Our framework tackles this end-to-end rather than forcing diverse data types into text-first abstractions. The core difficulty is variance: different modalities impose different computational profiles and token densities, which can destabilize distributed training. We reduce variance at every stage of the pipeline—from data streaming and packing to embedding and distributed execution—and pair it with high-utilization kernels and parallelism.

Design principles We structure the system around four principles: (1) **Zero-Pad** execution with document masking and sequence packing; (2) **Zero-NFS** data access via streaming object storage and local caching; (3) **Low-variance** execution across the pipeline (data streaming, packing, and

¹<https://huggingface.co/PerceptronAI/Isaac-0.1>

²<https://huggingface.co/PerceptronAI/Isaac-0.1-Base>

embedding) to stabilize distributed training; and (4) **High-utilization** kernels and parallelism (FSDP2, FlashAttention-3, `torch.compile`).

2.1 DATA

2.1.1 TENSORSTREAM

TensorStream [7] is a flexible and informative data format for multimodal interleaved data. The main innovation is packaging data into a stream of atomic packets tagged with rich descriptor metadata and allowing the data and descriptors to evolve independently. During each stage of data processing we know exactly how a given operation impacts the final interleaved tensor without forcing the data into complex formats which are difficult to expand on and maintain.

2.1.2 PACKING

We collate on the TensorStream level where Events are appended until a target sequence length is reached; when the next event would overflow, we slice an event to fill the remainder exactly and emit a packed sample. This produces dense sequences with minimal padding and preserves per-event masks for document masked training. The buffer can hold multiple packed samples and supports deterministic replay (for checkpointing) or randomized selection (for throughput).

2.1.3 TOKEN LEVEL DATA MIXTURES

Mixture targets are specified over *tokens*, not samples, because preprocessing choices (native resolution, packing) change tokens-per-sample and otherwise distort realized shares. We therefore estimate densities under the *actual* streaming pipeline and solve for dataset weights that achieve the requested mix.

Spec grammar. We support a compact string grammar with two target types and grouping:

- Epoch targets: `dataset:2.5e` means 2.5 epochs (passes over the dataset).
- Batch-fraction targets: `dataset:0.3b` means 30% of batches are drawn from this dataset.
- Groups: `(d1, d2):0.5b` allocates the *group* 50% of batches; for batch groups we preserve the group’s *natural token ratios* observed under streaming, i.e., $\text{fraction}(d) = \text{group_value} \cdot \frac{T_d}{\sum_{g \in \text{group}} T_g}$.

Simulation. A short simulation runs the real dataloader and collects, for each dataset i : unique-sample counts S_i , token counts T_i , and total available samples N_i . This yields token densities $r_i = T_i/S_i$ that already reflect packing and resolution settings. We first honor all batch-fraction specifications (including grouped fractions), then allocate the *leftover* batch mass to datasets with epoch spec in proportion to the *tokens required* to meet their epoch targets:

$$\text{tokens_needed}(i) = (\text{epoch}_{i_0} N_i) \cdot \underbrace{\frac{T_i}{S_i}}_{\text{tokens per unique sample}}$$

If no epoch-spec dataset needs tokens, we split the leftover equally over epoch datasets.

Weights. Fractions are converted to sampling weights with a simple correction for token density:

$$w_i \propto \frac{\text{fraction}_i}{\max\{1, r_i\}}, \quad \sum_i w_i = 1. \quad (1)$$

Intuitively, higher token-density datasets get proportionally smaller weights to keep realized *token* shares aligned with targets. We optionally run a second verification pass using the computed weights and, when epoch targets are present, derive a recommended maximum step count that minimizes a weighted squared error between achieved and target epochs.

Reproducibility and usage. The simulator runs with the same streaming config used for training (tokenizer, patch budgets, collator, shuffle settings) and writes (i) per-dataset sample/token statistics and (ii) a normalized weight string ready to drop into the training config.

2.1.4 ZERO-NFS

At petabyte scale, shared filesystems are expensive to provision and operate, and they become hot spots under many concurrent readers. Object storage (for example, S3 or GCS) is comparatively cheap, durable, and elastic. Existing solutions make progress [8] however operating at the python layer as limitations around GIL and shared memory. We therefore avoid shared filesystems entirely by streaming from object storage into an on-node cache coordinated by a Rust shard engine. The engine maintains a per-shard state machine (Remote → Preparing → Local, with Blacklisted and Evicting as needed) in shared memory so all workers and ranks on a node coordinate. Downloads are multi-process and concurrent across shards; for any given shard, coordination prevents duplicate fetches and enforces cache-space reservations against a node-level limit. Eviction uses an LRU policy with hysteresis (high and low watermarks) to prevent thrashing.

The engine is designed for Python multithreading and async I/O: blocking I/O releases the GIL, and background workers maintain progress independent of the dataloader. Process locks ensure exclusive download rights per shard, and timeouts plus blacklisting recover from slow or corrupt shards. On top of this, a greedy preloader maintains a dynamic lookahead window driven by the sampling plan so the next shards are staged before the model needs them.

On the data side we keep sampling cache friendly: we partition plans across data-parallel ranks, support balanced or probability-weighted mixtures, and avoid reshuffling within a cycle so that repeated samples hit the local cache. In multi-node training the cache is purely ephemeral and local to each node; there is no hard dependency on an NFS mount. Empirically we observe no MFU regressions relative to fully local datasets.

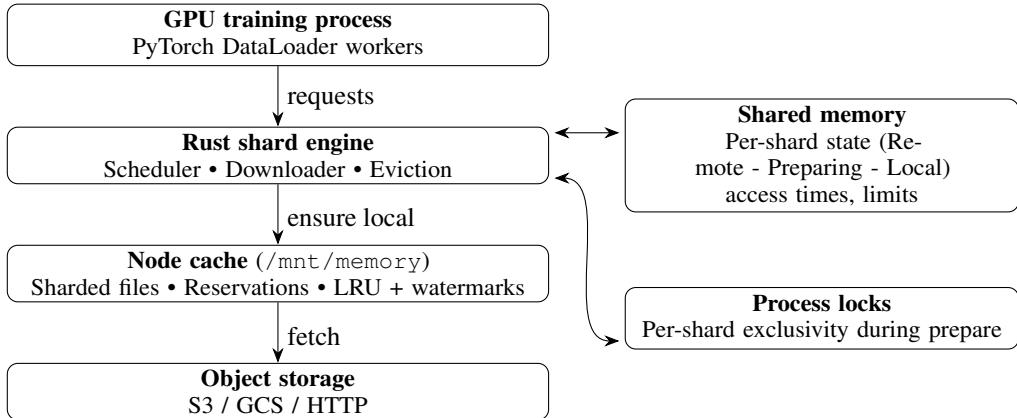


Figure 1: On-node streaming and caching with a Rust shard engine. DataLoader workers request shards; the engine coordinates multi-process downloads, cache reservations, and eviction via shared memory and process locks.

2.2 EMBEDDING

Embedding can become the performance bottleneck in multimodal training: vision/audio encoders are heavy, and the per-rank mix of modalities varies. Our vision embedder operates on *packed variable-length sequences*. For each image we form a block of NaViT patches (row-major order) and track its grid shape (h, w). A linear patch projection produces token embeddings; a learnable 2-D positional grid is resized to (h, w) and added. The encoder applies multi-head attention with variable-length kernels by supplying cumulative sequence lengths cu_seqlens computed from per-image block sizes. After the encoder we optionally apply pixel unshuffle [9] on the concatenated sequence: for scale r , we merge each $(r \times r)$ group of adjacent patches and increase channels by r^2 using a precomputed

gather-index map that respects each image’s grid. The resulting vision tokens are then projected to the backbone width by the connector MLP.

To keep utilization high when the per-rank mix of images varies, we *load balance inputs* before embedding. Each rank first all-gathers the list of block sizes (e.g., tokens per image/video chunk); a coordinator computes a movement plan to equalize load across ranks; and an `all_to_all_single` redistributes blocks accordingly. After embedding locally, we invert the plan to restore original ownership. Forward and backward use custom autograd wrappers so gradients follow the same communication pattern.

We implement multiple movement plans: a greedy bin-packing policy and an entropically-regularized optimal transport (Sinkhorn) policy with intra-node preference. Production builds use C++/CUDA inline extensions for both policies and dispatch them lazily on first use. Plans operate on per-block sizes and generate three tensors (`src_ranks`, `src_sizes`, `dst_ranks`); from these we derive per-rank send/recv sizes for variable-size `all_to_all_single`.

LOAD BALANCING PSEUDOCODE

Algorithm 1 Load-balanced embedding over variable-size blocks

```

1: function LOADBALANCE(tokens, sizes, plan_fn)
2:    $S_{\text{all}} \leftarrow \text{AllGatherObject}(\text{sizes})$ 
3:    $P \leftarrow \text{Broadcast}(\text{plan_fn}(S_{\text{all}}))$             $\triangleright$  movement plan tensors  $P = (\text{SRC}, \text{SIZE}, \text{DST})$ 
4:   (send_sizes, recv_sizes)  $\leftarrow \text{ComputeSendRecv}(P, \text{rank}, \text{world\_size})$ 
5:    $\text{blocks} \leftarrow \text{Split}(\text{tokens}, \text{sizes})$ 
6:    $\text{sendbuf} \leftarrow \text{Pack}(\text{blocks}, \text{DestOfMyBlocks}(P))$ 
7:    $\text{recvbuf} \leftarrow \text{AllToAllSingle}(\text{sendbuf}, \text{send_sizes}, \text{recv_sizes})$ 
8:   return (recvbuf, P)
9: function RESTORELOAD(balanced, P)
10:   $P^{-1} \leftarrow \text{Inverse}(P)$ 
11:  (send_sizes, recv_sizes)  $\leftarrow \text{ComputeSendRecv}(P^{-1}, \text{rank}, \text{world\_size})$ 
12:   $\text{blocks} \leftarrow \text{Split}(\text{balanced}, \text{SizesIHold}(P))$ 
13:   $\text{sendbuf} \leftarrow \text{Pack}(\text{blocks}, \text{SrcOfThose}(P))$ 
14:   $\text{restored} \leftarrow \text{AllToAllSingle}(\text{sendbuf}, \text{send_sizes}, \text{recv_sizes})$ 
15:  return ReorderToOriginal(restored, P)

```

For the planner, during image-text training we typically use a greedy redistribution algorithm; however when load becomes much less balanced we rebalance variable-sized blocks across R ranks by solving an entropically regularized optimal transport (OT) problem and then rounding to discrete moves, following Sinkhorn scaling [10]. Per-rank loads define source marginals p ; the uniform target load defines q . Movement cost is $\lambda |i - j|$ with a $\frac{1}{2}$ discount for intra-node transfers ($C_{ii} = 0$). We form $K = \exp(-C/\varepsilon)$ and run Sinkhorn scaling to obtain $T = \text{Diag}(u) K \text{Diag}(v)$ with row/column sums p, q , which favors short, intra-node moves (sharper as ε decreases). For integer assignments, we iterate over blocks, initialize capacities to L/R , mask destinations whose remaining capacity is below the block size, renormalize the i -th row of T as probabilities, and sample a destination (fallback: `argmax` capacity), updating capacities after each placement. This yields near-balanced loads with minimal movement; λ tunes distance aversion, ε controls plan sharpness, and `nodes_size` sets the intra-node discount.

2.3 ZERO-PAD

Native-resolution vision (*NaViT*) and interleaved multimodal streams make padding particularly costly. An image’s token count depends on its aspect ratio and target resolution; batching multiple images therefore produces ragged sequences with large intra-batch variance. Naïvely padding each sequence up to the batch maximum wastes memory and FLOPs, and on GPUs it degrades kernel efficiency by forcing small, heterogeneous matmuls. The problem is amplified when sequences contain alternating blocks of vision and text tokens with different densities.

Our Zero-Pad approach combines (i) **sequence packing** to form dense sequences from multiple samples and (ii) **document masks** to enforce block-diagonal attention . Packing eliminates most padding by exactly filling the target length with full or partial events (for vision, contiguous runs of NaViT patches; for text, token chunks). Document masks then prevent attention from crossing sample boundaries and restrict loss to the intended spans. This preserves NaViT’s benefits—native aspect ratios and dynamic shapes—without paying the padding tax.

2.4 EFFICIENT TRAINING & PARALLELISM

We build on the open-source `torchtitan` [11] stack, extending it to support native-resolution vision and interleaved sequences. Our design prioritizes low-variance execution and high hardware utilization, while remaining straightforward to operate.

Our training setup combines FSDP2 [12; 13], selective activation checkpointing [14], and `torch.compile` [15]. Transformer blocks are compiled statically, while embedders are compiled dynamically. To further accelerate performance, we employ FlashAttention-3 [16] and linear cross entropy [17], and train in BF16 precision.

Together, these choices allow our main runs to consistently achieve around **50% MFU**.

2.5 EXTENDING TO INFERENCE

We align the inference stack with the same design principles: (1) **Zero-Pad** and (2) **High utilization**. Inference consists of two phases: (i) *prefill*, which consumes a batched TensorStream of interleaved modalities; and (ii) *decode*, as we do text only generation this is the standard autoregressive token generation.

Our serving layer extends `sglang` [18]. The core difference in our fork is the scheduler’s unit of work is a TensorStream rather than token IDs, which lets us plan and batch over variable-length, interleaved vision/text blocks.

Zero-Pad. We use continuous batching and a batchless, document masked attention path allowing us to efficiently batch prefill and decode operations without padding.

High utilization. To keep the critical path GPU-bound, we minimize Python/IPC overhead: (a) custom binary serialization for TensorStreams (avoid `pickle`); and (b) multiprocessing tokenization/asset loading, since multimodal tokenization (fetching and converting assets to tensors) is a common bottleneck. We also avoid unnecessary movement of large tensors between processes and keep metadata-light control messages wherever possible.

3 RECIPE

3.1 ARCHITECTURE

We use a transformer backbone initialized from Qwen3 [5]. The vision path is a SigLIP-2 encoder [4] with native-resolution NaViT tokenization [6]. To control sequence length at native-resolution, we apply pixel unshuffle [9] to reduce spatial tokens and increase channel dimensionality, then bridge modalities with a 2-layer MLP that projects vision tokens to the backbone hidden size. We use multi-dimensional rotary embeddings (MRoPE) from Qwen2.5-VL [19].

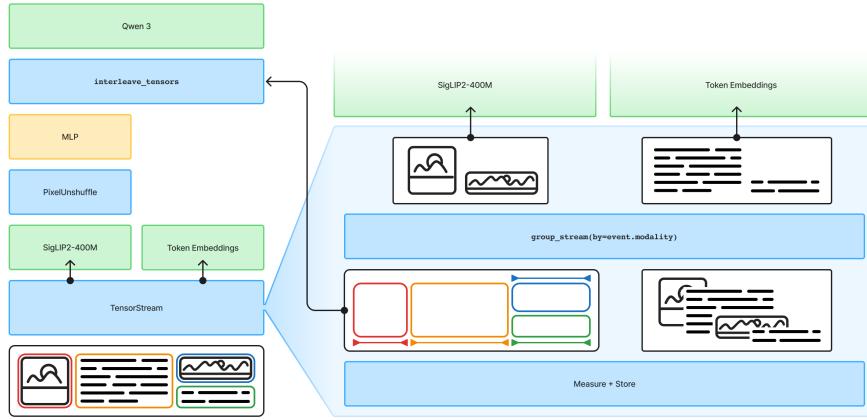


Figure 2: Overview of the architecture of Isaac-0.1

3.2 DATA

One of the main aspects we iterated on during this work was tokenization and formatting of our data. We paid special attention to how we format grounding data to minimize token waste and maximize generalization between samples.

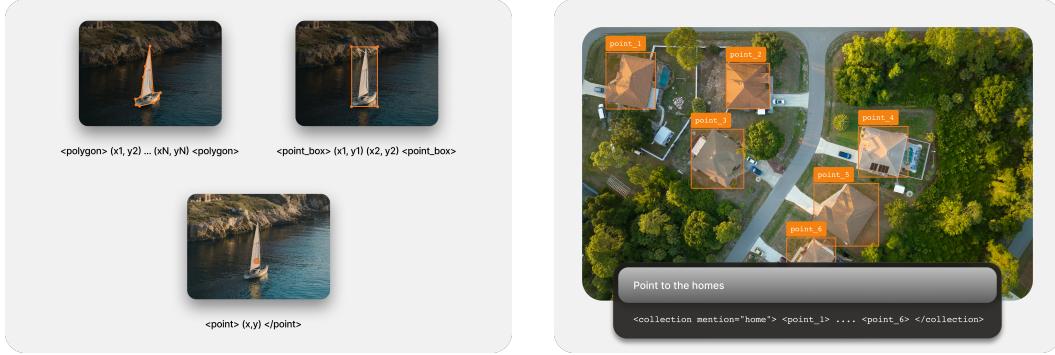
3.2.1 DATA SOURCES AND CURATION

Our training corpus aggregates approximately **300 unique datasets** spanning pointing - including individual points, bounding boxes, and polygons, detection, referring expressions, and pointing inside of VQA. Further we include, OCR and document understanding, chart and diagram understanding, image captioning with spatial tags, and counting. Each source is *instruction-wrapped* with a custom template to normalize task phrasing, serialize grounding targets into our text-native interface, and harmonize answer styles (conversational by default, structured when prompted). We de-duplicate at the image and token levels, canonicalize label taxonomies, and apply quality filters for malformed annotations and unsafe content. **Language: we filter for English only**, restricting training samples to English prompts/answers and, where applicable, English OCR content. Mixture targets are expressed in tokens (not samples) and enforced with the simulator in § 2.1.3 to remain stable under resolution and packing changes.

Text-only mixture To preserve instruction following, robustness, and general linguistic competence without drifting away from perception, we include a small *text-only* component amounting to **5% of total tokens**. This slice combines reasoning and non-reasoning instruction data and is interleaved with the multimodal stream during training. Capping pure-text exposure at roughly five percent keeps the model anchored on grounded perception tasks while retaining fluent, helpful responses in mixed-modality prompts.

3.2.2 ROBUST GROUNDING IN LANGUAGE MODELS

To enable integration with downstream systems perceptive scenarios often require machine-readable structured outputs. In these cases, the model must reference visual objects through explicit grounding representations. Such grounding covers tasks like localization, object detection and segmentation, where spatial references (pointing) - expressed as single points, bounding boxes, or polygons - provide necessary link between language and visual content. While several specialized models exist such as the YOLO [20] variants, promptable foundation models struggle reasoning over precise visual locations in text often supporting a single point type [21; 22]. We design a text based tokenization for pointing enabling Isaac-0.1 to learn grounding capabilities (Figure 3).



(a) Our 3 pointing primitives (point, box, polygon) are serialized as the above

(b) Collections and labels serve as meta attributes for pointed regions

Figure 3: Overview of our grounding representation (points and boxes)

This representation is applied consistently across different tasks, ensuring a unified format for spatial grounding. For example, in pointing tasks it serves as the model’s output to identify locations. In counting tasks it provides a way to enumerate and track multiple objects. It can also be embedded directly into user prompts as in-context examples, allowing users to reference specific regions or objects within an image using normalized coordinates.

Pointing Primitives (shown in figure 3a)

- **Single Points** (`<point> (x, y) </point>`): represent precise locations in image space.
- **Bounding Boxes** (`<point_box> (x0, y0) (x1, y1) </point_box>`): define rectangular regions.
- **Polygons** (`<polygon> (x0, y0) (x1, y1) ... </polygon>`): capture complex shapes through contour points.

In addition, to represent groups of objects, we introduced a collection primitive (figure 3b):

- **Collection** (`<collection> ... </collection>`), where ... is two or more other primitive pointing tags.

For visual citations and detection - we serialize an attribute for each point termed `mention` a natural language descriptor of the object connecting the spatial annotation to the conversational context.

Controlling Consistency A complexity of modeling structured output in text is the left to right nature of language modeling requires consistency in tokenization and order when representing an object. We employ the following to ensure our points are consistent.

- **Range Normalization:** We leverage a 0–1000 normalized coordinate system [22–24] that provides resolution independence across different image sizes.
- **Collection Sorting:** Collections are sorted by each of the points by min Y coordinate then X coordinate to have a canonical order.
- **Class-Level Sorting:** During multi-class training we ensure that the order of the mentioned points is consistent with the order the classes are referenced in the input.
- **Polygon Simplification:** Tokenization of complex polygons can result in several hundred coordinate points. We keep tokenization manageable by simplifying our polygons to a hull of 20 points with the Douglas–Peucker algorithm [25; 26]. We empirically select 20 points as it was able to achieve 95% IoU (Intersection over Union) after simplification on RefCoco [27].
- **Structure Hints:** Several questions can be ambiguous on the output format whether the structure should be a bounding box, polygon, or point, especially with interleaved text and points. We control for this by introducing a structure hint as a prefix to the system prompt

(<hint>POINT</hint>, <hint>BOX</hint>, <hint>POLYGON</hint>). The absence of the hint disables pointing.

3.2.3 CONVERSATIONAL POINTING (VISUAL CITATIONS)

Our grounding language is intended to be used in conversation, not as a separate head. Instead of emitting raw coordinates that must be post-processed, the model writes structured tags inline with its explanation. Mentions in the text (for example, "the large yellow excavator") are linked to spatial spans through attributes [21], so the answer and evidence travel together. Collections make multi-object prompts natural ("these three workers"), and structure hints select the intended primitive when the task is ambiguous. See figure 4 for an example.

This pattern has three practical advantages. First, it keeps the output self-describing and easy to parse in downstream systems. Second, it encourages the model to cite what it is reasoning about, which reduces ungrounded claims. Third, the same language is usable for input: a user or tool can insert a short example with tags to steer the model toward the desired behavior without fine-tuning.

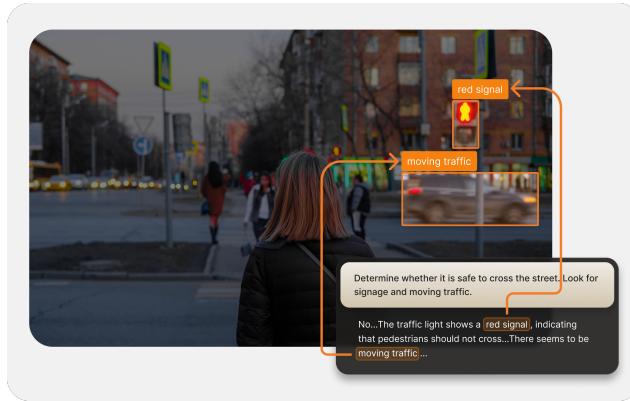


Figure 4: We interleave points across conversations to create grounded conversations

Hypothesis: Conversational Pointing Scales To Visual Reasoning We hypothesize that this interface is a natural way to scale to visual reasoning. Citations couple claims to specific spatial evidence, which shifts learning away from exploiting language priors and toward using the image. In practice this has several effects:

- **Better credit assignment:** the loss is applied to grounded spans, aligning language tokens with the visual tokens they depend on. This reduces shortcut solutions that ignore the image.
- **Evidence-driven reasoning:** multi-step answers can cite intermediate regions, which encourages the model to look before it reasons. At evaluation time we can require a correct citation for a response to receive credit.

3.3 TRAINING PROCEDURE

3.3.1 MIDTRAINING

Our recipe is uniquely simple which makes it a great base to build on top of. We perform one mid-training stage of 100B tokens. We train the model to perform next-token prediction only with response only supervision. More details of the mixture are in §2.1.3.

Hyperparameters

Component	Value
Optimizer	AdamW [28] LR = 2×10^{-5} ; $\beta = (0.9, 0.95)$; weight decay 0.1
Learning rate schedule	WSD-S [29] with cosine decay warmup 500 steps; cycle length 100 000 steps; decay ratio 0.1; final LR ratio 0.1
Gradient clip	$\ g\ _2 \leq 1.0$
Sequence length	8192
Batch size	64
Training steps	200 000

3.3.2 POST-TRAINING A CONVERSATIONAL GROUNDING INTERFACE

Midtraining datamix often contains a diverse set of datasets, which may have varying style, different (and often templated) prompt format, which negatively affect end-model perceived quality. We find that our mid-trained models excel across perceptive benchmarks; however they can require a rigid prompt structure to elicit capabilities. To align the model with the desired default style and behavior patterns we introduced a lightweight post-training stage aimed at smoothing out the experience and enabling a more interactive interface with Isaac, enabling conversational grounding via visual citations using our rich grounding language described above.

Post-training is done in two stages:

1. **Supervised Fine-Tuning (SFT).** We prepared a mix of datasets covering different perceptive tasks in a few variations, such as: conversational text response is our default, with visual citations added if any of the `<hint>` control sequences are included, as well as short structured output (such as detected object tags only), conditioned on the corresponding user prompt instruction to avoid conversational response.
2. **Direct Preference Optimization (DPO) [30]** In addition to SFT, we collected a small set of a few hundred prompts with responses graded as chosen or rejected in a pairwise way, and performed DPO to improve response quality. Following observations reported for instruction tuning (e.g., Llama 3 [31]), we make two tweaks: (i) add an auxiliary NLL term (on the chosen response) to stabilize optimization, and (ii) *mask out control and chat-template tokens* when computing the DPO loss so the algorithm does not lower the probability of low-entropy formatting tokens (system/user markers, BOS/EOS, etc.), which can otherwise cause premature stops during decoding [31].

4 RESULTS

We focus evaluation on perception-centric “Perception Core” benchmarks (grounding/pointing, OCR with layout, and VQA with localization); policy details and anti-benchmark criteria are in the Appendix.

Using our training techniques we created a model that competes with much larger systems on cognitive image QA (e.g., AI2D, RealWorldQA), while excelling on practical grounding applications such as segmenting or localizing large numbers of objects in diverse, real-world imagery.

We report results on our model as well as Gemini 2.5 (Pro & Flash) [32] and GLM-4.5V [1].

4.1 PROMPT SELECTION FOR EVALUATION

Our evaluation prompts were typically sourced from open source repositories like LMMS Labs Eval [66]. We simplified implementation where possible by using the same style prompt across tasks of similar type (e.g. all multiple choice exams use the same prompt, as do all counting tasks).

In circumstances where “standard” prompts resulted in low-style compliance from our external reference models (Gemini 2.5 Flash/Pro and GLM4.5V), we modified the prompts in semantics-preserving ways to elicit the desired behavior out of *all* models.

Task	Benchmark	Isaac-0.1	Gemini 2.5 Pro	Gemini 2.5 Flash	GLM 4.5V
Size	-	2B	-	-	106B-A12B
Pointing	Aerial Grounding	67.6	64.4	53.3	54.0
	Perceptron Grounding	50.5	28.1	24.8	0.3
	Perceptron Grounding (Reasoning-Ref)	65.9	71.1	74.3	70.2
	RefL4 (val) [33]	76.3	75.9	72.8	81.3 (89.5 [†])
	RefCOCO+ [34]	70.6	73.3	69.7	81.7
	RefCOCOg [35]	74.0	78.8	75.8	81.2
	RefCOCO [34]	77.1	78.4	76.7	84.0
Overall		68.9	67.1	63.9	64.7
OCR	ChartQA [36]	76.8	83.8	79.4	85.1
	DocVQA [37]	91.2	94.3	94.0	95.5
	A-OKVQA (val) [38]	89.0	93.2	91.2	89.8
	ClockBench [39]	21.2	54.6	35.4	15.0
	TextVQA [40]	79.3	76.4	80.3	82.6
	Overall	71.5	80.5	76.1	73.6
Counting	Aerial Counting	24.2	32.8	20.5	25.5
	CVBench [41]	74.4	70.2	70.8	76.8 (86.5 [†])
	PixMoCount [42]	62.8	73.7	57.6	65.0
	PGCount	39.6	41.8	39.6	11.9
	CountBench [43]	86.9	90.6	82.4	82.2
	iWildCam 2022 (Counting) [44]	36.8	56.2	54.4	14.0
Overall		54.1	60.9	54.2	45.9
General	VQA v2 [45]	82.2	69.0	76.9	80.6
	VLMs Are Blind [46]	64.6	83.4	60.1	60.1
	HallusionBench [47]	52.6	64.6	62.2	50.2 (59.1 [†])
	SEED-Bench [48]	76.0	79.2	76.2	43.2
	RealWorldQA [49]	75.0	80.5	80.1	54.3
	M3Exam (English) [50]	57.6	94.0	84.6	79.7
	MMMU [51]	39.7	73.9	66.7	53.9 (68.4 [†])
	ViBE-Eval [52]	46.2	76.8	72.9	67.2
	VLMs Are Biased [53]	23.8	42.6	29.9	12.8
	MMMU-Pro [54]	37.6	74.3	66.1	49.2 (59.8 [†])
	NLVR2 [55]	84.4	91.4	86.2	91.2
	BLINK [56]	42.6	79.4	72.6	61.3 (63.7 [†])
	VSR (Zero-Shot) [57]	78.4	87.4	79.6	86.2
	MathVista [58]	63.1	80.6	70.5	71.3 (78.2 [†])
	MathVision [59]	22.0	60.6	39.9	44.0 (52.5 [†])
	MME [60]	76.8	87.6	82.8	87.2
	AI2D [61]	76.4	90.8	87.6	85.4 (86.6 [†])
	ChartMuseum [62]	21.1	67.7	58.1	48.5 (47.1 [†])
	TreeBench [63]	39.3	52.3	51.1	48.9 (47.9 [†])
	ERQA [64]	34.8	54.9	48.2	47.1 (46.5 [†])
	MuirBench [65]	35.8	78.1	72.7	47.2 (71.1 [†])
Overall		53.8	74.7	67.9	60.5

Table 1: Overall results across 4 categories consisting of both open and proprietary benchmarks we compare Isaac-0.1 against the current closed and open source frontier. Values marked [†] are paper-reported.

We evaluated all models under the exact same prompting set up for fairness. Additionally, we disabled thinking when possible and explicitly instructed models to answer the questions directly though we designed our answer parsing functions to be robust to chain-of-thought reasoning if the models still exhibited thinking.

4.2 ARCHITECTURE ABLATIONS

We conduct a series of ablations behind the architecture for Isaac-0.1. For ablation experiments we leverage the 600M Qwen3 backbone, train on 16 GPUs for 50k steps. We compare the 2 architectural changes by (1) We remove pixel unshuffle and (2) we freeze the vision encoder, keeping the SigLIP-2 initialization. Our results are presented as category averages across our evaluation datasets in Table 2. We find that training through the vision encoder gives us a consistent improvement across all categories - with primary improvements in OCR and Document understanding. When comparing pixel unshuffle - the primary benefits come from the ability to process larger images at the same sequence length allowing for finegrained visual understanding.

Model	Pointing	Counting	OCR	Image QA
Isaac-0.1 (base)	59.9	47.4	61.8	46.8
- Pixel unshuffle	59.6	49.2	60.1	47.4
+ Frozen Vision Encoder	58.9	46.0	53.5	45.8

Table 2: Architecture ablations for Isaac-0.1 comparing removing pixel unshuffle and freezing the vision encoder. Bold indicates best per column.

4.3 OVERALL RESULTS

Overall, Isaac-0.1 is very strong at pointing of all kinds and at basic visual understanding and OCR. As expected for a 2B model, it underperforms on advanced mathematical and scientific reasoning. Counting is slightly below our pointing strength; we expect targeted reasoning supervision to improve it. See results in Table 1.

5 CONCLUSION

We presented Isaac-0.1, a 2B-parameter, open-weight vision–language model designed for grounded perception at the efficiency frontier. The model couples a unified, text-native grounding interface with a simple, reproducible recipe and an infrastructure stack that reduces variance and waste: Zero-Pad training via sequence packing with document masked attention, Zero-NFS streaming through an on-node cache, and input load balancing. With a single checkpoint, Isaac-0.1 delivers strong localization and competitive OCR and VQA at practical latency and cost (Table 1).

REFERENCES

- [1] V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, Aohan Zeng, Baoxu Wang, Bin Chen, Boyan Shi, Changyu Pang, Chenhui Zhang, Da Yin, Fan Yang, Guoqing Chen, Jiazheng Xu, Jiale Zhu, Jiali Chen, Jing Chen, Jinhao Chen, Jinghao Lin, Jinjiang Wang, Junjie Chen, Leqi Lei, Letian Gong, Leyi Pan, Mingdao Liu, Mingde Xu, Mingzhi Zhang, Qinkai Zheng, Sheng Yang, Shi Zhong, Shiyu Huang, Shuyuan Zhao, Siyan Xue, Shangqin Tu, Shengbiao Meng, Tianshu Zhang, Tianwei Luo, Tianxiang Hao, Tianyu Tong, Wenkai Li, Wei Jia, Xiao Liu, Xiaohan Zhang, Xin Lyu, Xinyue Fan, Xuancheng Huang, Yanling Wang, Yadong Xue, Yanfeng Wang, Yanzi Wang, Yifan An, Yifan Du, Yiming Shi, Yiheng Huang, Yilin Niu, Yuan Wang, Yuanchang Yue, Yuchen Li, Yutao Zhang, Yuting Wang, Yu Wang, Yuxuan Zhang, Zhao Xue, Zhenyu Hou, Zhengxiao Du, Zihan Wang, Peng Zhang, Debing Liu, Bin Xu, Juanzi Li, Minlie Huang, Yuxiao Dong, and Jie Tang. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025. URL <https://arxiv.org/abs/2507.01006>.
- [2] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhui Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025. URL <https://arxiv.org/abs/2508.18265>.
- [3] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang

-
- Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. Paligemma 2: A family of versatile vlms for transfer, 2024. URL <https://arxiv.org/abs/2412.03555>.
- [4] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features, 2025. URL <https://arxiv.org/abs/2502.14786>.
 - [5] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
 - [6] Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim Alabdulmohsin, Avital Oliver, Piotr Padlewski, Alexey Gritsenko, Mario Lučić, and Neil Houlsby. Patch n' pack: Navit, a vision transformer for any aspect ratio and resolution, 2023. URL <https://arxiv.org/abs/2307.06304>.
 - [7] Maciej Kilian and PerceptronAI. Tensorstream: Structuring multimodal data for causal transformers. *PerceptronAI Research Blog*, 2025. <https://perceptron.inc/blog/tensorstream>.
 - [8] The Mosaic ML Team. streaming. <<https://github.com/mosaicml/streaming/>>, 2022.
 - [9] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016. URL <https://arxiv.org/abs/1609.05158>.
 - [10] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, 2013.
 - [11] Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. Torchttitan: One-stop pytorch native solution for production ready llm pre-training, 2025. URL <https://arxiv.org/abs/2410.06511>.
 - [12] PyTorch Community. Pytorch fsdp2 rfc. <https://github.com/pytorch/pytorch/issues/114299>, 2023. URL <https://github.com/pytorch/pytorch/issues/114299>. GitHub issue #114299.
 - [13] PyTorch Community. Pytorch dtensor rfc. <https://github.com/pytorch/pytorch/issues/88838>, 2023. URL <https://github.com/pytorch/pytorch/issues/88838>. GitHub issue #88838.
 - [14] PyTorch Community. Pytorch checkpoint. <https://pytorch.org/docs/stable/checkpoint.html>, 2023. URL <https://pytorch.org/docs/stable/checkpoint.html>. PyTorch documentation (stable).
 - [15] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael

-
- Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Pührsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS ’24, page 929–947, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703850. doi: 10.1145/3620665.3640366. URL <https://doi.org/10.1145/3620665.3640366>.
- [16] Tri Dao et al. Flashattention-3: Fast and memory-efficient exact attention, 2024. URL <https://arxiv.org/abs/>. arXiv preprint.
 - [17] Erik Wijmans, Brody Huval, Alexander Hertzberg, Vladlen Koltun, and Philipp Krähenbühl. Cut your losses in large-vocabulary language models, 2025. URL <https://arxiv.org/abs/2411.09009>.
 - [18] SGLang Contributors. Sglang. GitHub repository, 2024. <https://github.com/sgl-project/sqlang> (accessed 2025-09-22).
 - [19] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL <https://arxiv.org/abs/2502.13923>.
 - [20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.
 - [21] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models, 2024. URL <https://arxiv.org/abs/2409.17146>.
 - [22] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL <https://arxiv.org/abs/2502.13923>.
 - [23] Google DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL <https://arxiv.org/abs/2507.06261>.
 - [24] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. Paligemma 2: A family of versatile vlms for transfer, 2024. URL <https://arxiv.org/abs/2412.03555>.
 - [25] Urs Ramer, David Douglas, and Thomas Peucker. Ramer–douglas–peucker algorithm. https://en.wikipedia.org/wiki/Ramer%20-%20Douglas%20-%20Peucker_algorithm, 1973. Accessed 2025-09-21.

-
- [26] Shapely Contributors. Shapely: manipulation and analysis of planar geometric objects. <https://shapely.readthedocs.io/>, 2025. Version referenced for polygon simplification.
- [27] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. Refcoco, refcoco+, and refcocog: Datasets for referring expressions, 2016. URL <https://arxiv.org/abs/1608.00272>.
- [28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL <https://arxiv.org/abs/1711.05101>.
- [29] Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective, 2024. URL <https://arxiv.org/abs/2410.05192>.
- [30] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL <https://arxiv.org/abs/2305.18290>.
- [31] Meta AI Research. Llama 3: Open and efficient foundation language models. Technical report / blog, 2024. <https://ai.meta.com/research/publications/llama-3-open-and-efficient-foundation-language-models/> (accessed 2025-09-22).
- [32] Google Deepmind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL <https://arxiv.org/abs/2507.06261>.
- [33] Jierun Chen, Fangyun Wei, Jinjing Zhao, Sizhe Song, Bohuai Wu, Zhuoxuan Peng, S.-H. Gary Chan, and Hongyang Zhang. Revisiting referring expression comprehension evaluation in the era of large multimodal models, 2024. URL <https://arxiv.org/abs/2406.16866>. Introduces the Ref-L4 benchmark and cleaned RefCOCO/RefCOCO+/RefCOCOg splits.
- [34] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. Modeling context in referring expressions. In *European Conference on Computer Vision (ECCV)*, pages 69–85. Springer, 2016. doi: 10.1007/978-3-319-46475-6_5. URL https://link.springer.com/chapter/10.1007/978-3-319-46475-6_5. Introduces RefCOCO and RefCOCO+.
- [35] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Mao_Generation_and_Comprehension_CVPR_2016_paper.html. Introduces the Google Refexp (RefCOCOg) dataset.
- [36] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning, 2022. URL <https://arxiv.org/abs/2203.10244>.
- [37] Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021. URL https://openaccess.thecvf.com/content_WACV2021/papers/Mathew_DocVQA_A_Dataset_for_VQA_on_Document_Images_WACV_2021_paper.pdf.
- [38] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In *European Conference on Computer Vision (ECCV)*, 2022. URL https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136680141.pdf.

-
- [39] Alek Safar and Oleg Chichigin. Clockbench: Visual time benchmark where humans beat the clock, llms don't. <https://clockbench.ai/ClockBench.pdf>, 2025. Visual analog clock reading benchmark.
- [40] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. URL https://openaccess.thecvf.com/content_cvpr_2019/papers/Singh_Towards_VQA_Models_That_Can_Read_CVPR_2019_paper.pdf. Introduces the TextVQA dataset.
- [41] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Austin Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal llms, 2024. URL <https://arxiv.org/abs/2406.16860>. Introduces the Cambrian Vision-Centric Benchmark (CV-Bench).
- [42] Allen Institute for AI. Pixmo-count. <https://huggingface.co/datasets/allenai/pixmo-count>, 2025. Dataset card; part of the PixMo collection used in Molmo & PixMo (arXiv:2409.17146).
- [43] Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching clip to count to ten. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://openaccess.thecvf.com/content_ICCV2023/papers/Paiss_Teaching_CLIP_to_Count_to_Ten_ICCV_2023_paper.pdf. Introduces Count-Bench.
- [44] iwildcam 2022 (fgvc9): Counting the number of animals across image sequences. <https://www.kaggle.com/competitions/iwildcam2022-fgvc9>, 2022. Competition task and data description.
- [45] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/papers/Goyal_Making_the_v_CVPR_2017_paper.pdf.
- [46] Pooyan Rahmazadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind: Failing to translate detailed visual features into words. In *Asian Conference on Computer Vision (ACCV)*, 2024. URL https://openaccess.thecvf.com/content/ACCV2024/papers/Rahmazadehgervi_Vision_language_models_are_blind_ACCV_2024_paper.pdf.
- [47] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusion-bench: An advanced diagnostic suite for entangled language hallucination & visual illusion in large vision-language models, 2023. URL <https://arxiv.org/abs/2310.14566>.
- [48] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension, 2023. URL <https://arxiv.org/abs/2307.16125>.
- [49] xAI. Realworldqa: A benchmark for real-world visual question answering. <https://huggingface.co/datasets/xai-org/RealworldQA>, 2024.
- [50] Wenxuan Zhang, Sharifah Mahani Aljunied, Chang Gao, Yew Ken Chia, and Lidong Bing. M3exam: A multilingual, multimodal, multilevel benchmark for examining large language models, 2023. URL <https://arxiv.org/abs/2306.05179>. English subset commonly referenced as M3Exam_English.

-
- [51] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/papers/Yue_MMMU_A_Massive_Multi-discipline_Multimodal_Understanding_and_Reasoning_Benchmark_for_CVPR_2024_paper.pdf.
- [52] Piotr Padlewski, Max Bain, Matthew Henderson, Zhongkai Zhu, Nishant Relan, Hai Pham, Donovan Ong, Kaloyan Aleksiev, Aitor Ormazabal, Samuel Phua, Ethan Yeo, Eugenie Lampecht, Qi Liu, Yuqi Wang, Eric Chen, Deyu Fu, Lei Li, Che Zheng, Cyprien de Masson d'Autume, Dani Yogatama, Mikel Artetxe, and Yi Tay. Vibe-eval: A hard evaluation suite for measuring progress of multimodal language models, 2024. URL <https://arxiv.org/abs/2405.02287>.
- [53] An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. Vision language models are biased, 2025. URL <https://arxiv.org/abs/2505.23941>.
- [54] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhui Chen, and Graham Neubig. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL <https://aclanthology.org/2025.acl-long.736.pdf>.
- [55] Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6418–6428, Florence, Italy, 2019. doi: 10.18653/v1/P19-1644. URL <https://aclanthology.org/P19-1644/>.
- [56] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision (ECCV)*, 2024. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/papers/03356.pdf.
- [57] Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023. doi: 10.1162/tacl_a_00566. URL <https://aclanthology.org/2023.tacl-1.37/>.
- [58] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=KUNzEQMWU7>.
- [59] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with MATH-vision dataset. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad0edc7d5fa1a783f063646968b7315b-Abstract-Datasets_and_Benchmarks_Track.html.
- [60] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2023. URL <https://arxiv.org/abs/2306.13394>.
- [61] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European Conference on Computer Vision*

-
- (ECCV), 2016. URL <https://arxiv.org/abs/1603.07396>. Introduces the AI2D diagram dataset.
- [62] Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Manya Wadhwa, Zeyu Leo Liu, Zayne Sprague, Ramya Namuduri, Bodun Hu, Juan Diego Rodriguez, Puyuan Peng, and Greg Durrett. Chartmuseum: Testing visual reasoning capabilities of large vision-language models, 2025. URL <https://arxiv.org/abs/2505.13444>.
- [63] Haochen Wang et al. Traceable evidence enhanced visual grounded reasoning: Evaluation and methodology, 2025. URL <https://arxiv.org/abs/2507.07999>. Introduces TreeBench.
- [64] Google DeepMind Gemini Robotics Team. Erqa: Embodied reasoning question answer benchmark. <https://github.com/embodiedreasoning/ERQA>, 2025. Released with the Gemini Robotics tech report: arXiv:2503.20020.
- [65] Fei Wang, Xingyu Fu, James Y. Huang, Zekun Li, Qin Liu, Xiaogeng Liu, Mingyu Derek Ma, Nan Xu, Wenxuan Zhou, Kai Zhang, Tianyi Lorena Yan, Wenjie Jacky Mo, Hsiang-Hui Liu, Pan Lu, Chunyuan Li, Chaowei Xiao, Kai-Wei Chang, Dan Roth, Sheng Zhang, Hoifung Poon, and Muham Chen. Muirbench: A comprehensive benchmark for robust multi-image understanding, 2024. URL <https://arxiv.org/abs/2406.09411>.
- [66] Bo Li, Peiyuan Zhang, Kaichen Zhang, Fanyi Pu, Xinrun Du, Yuhao Dong, Haotian Liu, Yuanhan Zhang, Ge Zhang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Accelerating the development of large multimodal models, March 2024. URL <https://github.com/EvolvingLMMS-Lab/lmms-eval>.

6 APPENDIX

6.1 EXAMPLES



Q: Determine the focal point of this scene.

A: The focal point is the **white sailboat** at the center. Its single sail, reflection, and wake draw attention against the calm blue water and shoreline.



Q: Run a quality assessment on the solar panel. Localize any damage. Be concise.

A: Significant damage: **17 small holes** scattered across the surface. Efficiency likely reduced; recommend inspection and possible replacement.



Q: Determine when someone is present at either workstation. Send me a one sentence alert.

A: Alert: A worker is present at the **left workstation**!



Q: point to the homes

A: **home**



Q: (two images) Which objects, if any, moved between frames? Think carefully.



Q: Segment out the leak

A: **leak**

A: The **marker** moved: from the **bottom-right** to the **top-left** of the black surface. No other objects appear to have moved.

6.2 PERCEPTION CORE AND ANTI-BENCHMARKS POLICY

Prioritization of pointing/grounding and other perceptual benchmarks. We focus attention on OCR, pointing, and counting as high-signal benchmarks for a 2B parameter vision model. Many commonly used multi-modal benchmarks like MMMU or AI2D show a strong bias towards large, cognitive models with poor visual perception. Questions from these datasets can often be answered directly from world-knowledge, or rely only very minimally on the model's visual or spatial reasoning, instead relying on the model's ability to do symbolic problem solving tasks like complex math. As a perception company training a 2B vision model we view benchmarks like these as being 'anti-benchmarks' when compared with benchmarks more directly relevant to visual grounding, such as those in our counting and pointing categories.

6.3 DETAILED LIST OF EVALS

6.3.1 PROPRIETARY EVALS

- **Perceptron Grounding (PG)**

Metric: All point union IoU

Description: A multi-object, multi-class “tagged” pointing benchmark over proprietary imagery. Given an image and a list of target categories, the model must localize each instance by outputting bounding boxes tagged with the correct category.

Equal weight subsets:

- PG Clean: Base version of the dataset.
- PG Many: Subset with 10 or more objects per image.
- PG Fine: Subset where target boxes are smaller than 32×32 pixels.
- PG Multi-Class: Subset with 5 or more distinct object categories in the answer.

- **Aerial Grounding**

Metric: All point union IoU

Description: Combined bounding box pointing tasks adapted from CarPK and AID.

Equal weight subsets:

- AID 0-10: Adaptation of AID, subset with 10 or fewer objects
- AID 10-30: Adaptation of AID, subset with 10 to 30 objects.
- AID 30+: Adaptation of AID, subset with 30+ objects.
- CarPK 10-30: Adaptation of CarPK, subset with 10 to 30 cars.
- CarPK 30-100: Adaptation of CarPK, subset with 30 to 100 cars.
- CarPK 100+: Adaptation of CarPK, subset with 100+ cars.

- **Aerial Counting**

Metric: Exact match with relative tolerance

Description: Combined counting tasks adapted from CarPK and AID.

Equal weight subsets:

- AID 0-10: Adaptation of AID, subset with 10 or fewer objects. Relative tolerance of 10%.
- AID 10-30: Adaptation of AID, subset with 10 to 30 objects. Relative tolerance of 10%.
- AID 30+: Adaptation of AID, subset with 30+ objects. Relative tolerance of 10%.
- CarPK: Adaptation of CarPK, not stratified by count, taken in the original distribution of the dataset. Relative tolerance of 2.5%.

- **PG Count**

Metric: Multiple-choice exact match; numeric counting exact match (with the same relative tolerance as above for counts)

Description: Synthetic counting tasks derived from PG Clean. We select common object categories and construct three formats: (i) single-image counting (e.g., “How many helmets are in this image?”), (ii) multi-image counting over a small set, and (iii) multiple-choice comparisons (e.g., “Which image contains the most X?”). For numeric items, ground-truth counts are validated against the underlying annotations.

- **PG Reasoning Referring Expression**

Metric: All point union IoU

Description: Synthetic referring-expression tasks derived from PG Clean. We overlay the labeled boxes on an image and generate natural-language prompts that uniquely identify the target entities using semantic properties (size, color, relations). A prediction is considered correct when the localized regions align with the ground-truth annotations under the metric.

- **iWildCam Counting 2022**

Metric: Exact match within $\pm 2.5\%$ relative tolerance

Description: Curated from the 2022 iWildCam wildlife counting challenge. Each item is a short burst from trail cameras; the task is to count the number of distinct animals across the sequence (not per frame).

N.B. All point union IoU is a multi-point generalization of IoU that takes all the pixels enclosed by the predicted bounding boxes and all the pixels enclosed by the ground truth bounding boxes and computes the traditional intersection-over-union (IoU) of the resulting binary masks.

$$\text{All point union IoU} = \frac{\left| \left(\bigcup_i B_i^{\text{pred}} \right) \cap \left(\bigcup_j B_j^{\text{gt}} \right) \right|}{\left| \left(\bigcup_i B_i^{\text{pred}} \right) \cup \left(\bigcup_j B_j^{\text{gt}} \right) \right|}$$

N.B. The exact match with the relative tolerance X% is defined as:

$$\text{EM}_{X\%}(\hat{y}, y) = \mathbf{1} \left\{ \frac{|\hat{y} - y|}{|y|} \leq \frac{X}{100} \right\}$$

N.B. All evals run with temperature 0.0, and with single-attempt pass@1.