Open Source VLM Comparison: BBS Puppet Show (Under Progress)

Kazi Islam

Version 1.0

Prepared for Prof David Smith

Date [10-21-2025]

1. Abstract

This document presents a clear, step-by-step framework for comparing five open-source vision-language models—Qwen2-VL-7B-Instruct, LLaVA-OneVision-1.5 (4B/8B), Idefics-2-8B, Phi-3.5-Vision, and InternVL2-8B—on the task of image description. We upload the same set of images for the BBS project to each model and collect their captions. The captions are graded using simple, consistent criteria: correctness, detail, faithfulness to the image (low hallucinations), and speed. We also note how easy each model is to run and customize. Results are summarized with straightforward scores and brief observations of strengths and weaknesses. The goal is to provide a practical, repeatable way to pick the best model for BBS captioning in rehearsals and live use.

2. Objective

The objective of this document is to establish a standardized and transparent framework for evaluating image descriptions generated by five open-source vision-language models—Qwen2-VL-7B-Instruct, LLaVA-OneVision-1.5 (4B/8B), Idefics-2-8B, Phi-3.5-Vision, and InternVL2-8B. Using the same BBS puppet/prop images and consistent prompts, we apply a simple scoring rubric across key criteria: correctness, level of detail, faithfulness/low hallucination, on-image text (OCR) accuracy, speed/latency, and ease of use. The goal is to provide an unbiased, replicable process that yields clear scores and concise observations, enabling informed comparisons and practical model selection for BBS captioning in rehearsals and live workflows. This framework is also used in our VLM Benchmarking.

3. Why These Open Source LLMs?

We chose these five vision-language models (Qwen2-VL-7B-Instruct, LLaVA-OneVision-1.5 4B/8B, Idefics-2-8B, Phi-3.5-Vision, InternVL2-8B) because they're open-weight with permissive licences, have strong docs and communities, span useful sizes for real hardware (laptop → single GPU → small server), support LoRA/fine-tuning, and align directly with our captioning/OCR task. In contrast, we excluded closed or API-only systems (harder to reproduce/cost-control), projects with restrictive or unclear licensing, immature repos lacking stable training/eval code, narrow research prototypes that don't fit captioning, and heavyweight models that require multi-GPU H100-class setups. If we need a quality ceiling or specialized OCR later, we can add a closed-API baseline (e.g., GPT-4o/Claude Vision) or OCR-focused models. Overall, this set gives a reproducible, well-documented, size-diverse benchmark that's practical for BBS today with clear upgrade paths for tomorrow.

4. Image Selection

We will curate internet-sourced images of diverse everyday objects such as tools, toys, containers, signage with intentionally varied resolutions and aspect ratios from low (≈480p) to medium (≈720–1080p) and high (≥2K). We will mainly focus on these image resolution and aspect ratios:

1. **1080p (Full HD)** — 1920×1080
   The everyday standard for streaming, games, laptops, webcams, and TVs.
2. **4K UHD (2160p)** — 3840×2160
   Common on new TVs/cameras and YouTube/Netflix "4K" content; preferred for high-detail and future-proofing.
3. **720p (HD)** — 1280×720
   Still widely used for live streams, low-bandwidth viewing, and budget devices.
4. **1440p (QHD)** — 2560×1440
   Popular with PC gamers/creators for sharper monitors without the heft of 4K.

5. Non Quantified Grading Criteria

This non quantified grading criteria provides a clear, repeatable way to judge vision-language models for BBS captioning in both rehearsal and live use. Reviewers follow the same setup, run the same tests, and record observations about what the model describes, how grounded and detailed those descriptions are, how quickly and consistently results arrive on the intended hardware, and how smoothly the software installs, runs, and adapts. The aim is a practical, side-by-side comparison that surfaces strengths, weaknesses, and deployment readiness without requiring deep expertise—so a team can quickly choose the best option for their workflow.

Criteria	Excellent	Good	Poor
Correctness (objects, attributes)	Identifies all primary objects and key attributes correctly; no factual errors.	Core objects correct; only minor slips or omissions.	Misidentifies the main subject or repeatedly labels elements incorrectly.
Level of Detail (salient coverage, specificity)	Captures all salient elements with specific, concise details (color, material, spatial relations).	Covers most salient points; a few specifics missing; still informative.	Vague/telegraphic; overlooks important elements
Faithfulness / Low Hallucination	No invented objects/text; avoids unsupported inferences; strictly image-grounded.	Rare minor speculation that doesn't change meaning.	Frequent fabrications or contradictions with what's visible.
Speed / Latency (practicality on target hardware)	Responsive for interactive use; stable times across low/med/high resolutions.	Generally fast with occasional slowouts; acceptable for rehearsal.	Too slow/variable for practical workflows.
Ease of Use (setup, docs, tooling)	Clear docs; quick start; working examples; smooth LoRA/PEFT path and inference scripts.	Good docs; a few manual steps; examples mostly work.	Fragile setup; poor docs; repeated blockers without deep expertise.

6. Quantified Grading Criteria

This quantified grading criteria below provides a comprehensive way to score each model consistently with a text-first, repeatable process. For **Correctness**, list the expected primary objects and their key attributes, give credit only for accurate, image-grounded mentions, optionally reward correct secondary objects, and reduce credit when the caption invents objects or attributes. For **Level of Detail**, preselect a small set of salient slots (main subject, key action, on-image text, a distinctive attribute, and scene or a crucial relation) and assess how completely and specifically the caption covers them; reward clear specifics such as color, count, size, material, pose, simple spatial relations, brand/logo text, or numbers, and prefer concise, readable phrasing while reducing credit for irrelevant or not-visible claims. For **Faithfulness**, begin from a fully faithful baseline and reduce credit for invented primary or secondary elements, fabricated or misread text, wrong attributes or relations, unsupported storytelling, or internal contradictions; keep deductions confined to faithfulness so you don't double-penalize elsewhere. For **Speed/Latency** on your target hardware, warm up once, then run repeated single-image requests at a baseline and a higher resolution; compare typical timing to tail timing to judge stability, check how latency scales with resolution, and note any slow first response or long load/initialization. For **Ease of Use**, evaluate whether official docs clearly guide install and first run, how quickly and with how many steps you reach a first caption in a clean environment, whether a "hello world" script works as written, how reliable installs are across different machines, and whether examples, runtime controls, a usable CLI or Python API, fine-tuning recipes (e.g., LoRA/QLoRA) with configs and logging, and ecosystem packaging (model cards, pip/transformers, or Docker) are available; reduce credit for undocumented blockers or mismatches between documentation and code. Use consistent language for "Excellent," "Good," and "Poor" bands to summarize outcomes and, if desired, average the dimensions for a single overall rating.

6.1. Correctness (Objects, Attributes) — Rubric (0–100)

Correctness measures how accurately a caption names what's actually in the image and describes those things with the right properties. For each image, we define the primary objects (must-have subjects) and their attributes (e.g., color, material, simple relations), plus optional secondary objects (nice-to-have context). Scores prioritize getting primary objects right, then their attributes, with a small bonus for correctly noting secondary items. We also deduct for hallucinations—objects the caption claims that aren't present—so guessing doesn't help. The final result is a 0–100 score mapped to clear bands (Excellent, Good, Poor), giving a reliable, image-grounded snapshot of how truthful and specific a model's recognition really is.

6.1.1. What to grade (define once per image)

- **Primary objects (must-have):** the main subjects you expect.
- **Secondary objects (nice-to-have):** supporting items that appear.
- **Attributes:** facts tied to objects (color, material, simple relations like "on top of").

6.1.2. How to score (step by step, in words)

6.1.2.1. Primary objects — 60 points

- Split 60 points evenly across all primary objects in the ground truth.
- For each primary object correctly named, give its share of points.
- If it's missed or misnamed, give 0 for that object.

6.1.2.2. Attributes — 40 points

- Split 40 points evenly across all attribute slots in the visible sight.
- For each attribute value correctly stated (and attached to the correct object), give its share of points.
- If the attribute is missing or wrong, give **0** for that slot.

6.1.2.3. Secondary objects — bonus up to +10 (optional)

- Split 10 bonus points evenly across all secondary objects.
- For each secondary object correctly named, add its share to the score.
- If you don't annotate secondary objects, bonus = 0.

6.1.2.4. Hallucinations — subtract up to −15

- For each extra object the model claims that is not in the ground truth, subtract 3 points.
- Stop subtracting once you reach −15 (maximum penalty).

6.1.3. Final score

- Start from the Primary points + Attributes points + Secondary bonus.
- Subtract the Hallucination penalty.
- Between 0 and 100.

`Grade bands

- Excellent: 90–100
- Good: 75–89
- Poor: 0–74

Quick example

- GT primaries: 3 objects → each worth 20 pts (total 60). The Model gets 2 right → 40 pts.
- GT attributes: 4 slots → each worth 10 pts (total 40). The model gets 3 right → 30 pts.
- GT secondaries: 2 objects → bonus 5 pts each (total +10). The model gets 1 → +5 pts.
- Hallucinations: model adds 2 nonexistent objects → −6 pts.
- Final: 40 + 30 + 5 − 6 = 69 → Poor.

6.2. Level of Detail — Text Rubric (0–100)

Level of Detail captures how completely and specifically a caption describes what matters in the image—without fluff. For each image, we preselect a small set of salient elements a reasonable viewer would mention first (main subject, key action, any readable on-image text, a distinctive attribute, and scene or a crucial relation). The score rewards covering these elements clearly, then adds credit for useful specifics (e.g., color, count, size, material, pose, simple spatial relations, logo/text snippets) and gives a small boost for concise, readable phrasing. We also reduce credit for details that aren't actually visible or are off-topic, so verbosity doesn't inflate results. The outcome is a 0–100, image-grounded snapshot of how rich, precise, and economical the model's description really is.

What is a "salient element"?

A salient element is something a reasonable viewer would mention in the first 1–2 sentences because it's visually important or distinctive. To keep grading consistent, pick up to 6 slots per image from this short list (in order):

1. Main subjects (people/puppets/props/objects) — up to 2 slots
2. Key action or event (what the subject is doing) — up to 1 slot
3. On-image text (legible signs/labels/numbers) — up to 1 slot
4. Distinctive attribute (striking color/pattern/material/size) — up to 1 slot
5. Scene/context (where it is: stage, workshop, street) or a crucial spatial relation (e.g., puppet holding ball) — up to 1 slot

6.2.1. Salient Coverage (60 pts total)

Score each of the **K** salient slots (max 6), **10 pts each**:

- **10** = correctly and explicitly covered (e.g., "blue puppet holding a red ball" covers subject + action)
- **5** = mentioned but too generic (e.g., "a puppet" without the notable color/action)
- **0** = missing or wrong

Coverage = sum of slot points (max 60).

6.2.2. Specifics Count (30 pts total)

Award +3 pts per correct specific detail, up to 10 details (cap at 30 pts).
Valid specifics (must be visible and accurate): color, count, size, material, pattern, pose, part names, simple spatial relations (left/right/on/behind), brand/logo text, numbers/dates.

Specifics = 3 × (# correct specifics, capped at 10).

6.2.3. Conciseness & Clarity (10 pts)

- **10** = clear, non-repetitive, about **15–60 words**
- **5** = slightly short/long (5–14 or 61–90 words) but still clear
- **0** = telegraphic or rambling/unclear

6.2.4. Irrelevant/Not-Visible Detail Penalty (up to −10)

- Subtract −5 for each detail that is not visible or off-topic, cap at −10.
  (If you penalize hallucinations in a separate “Faithfulness” criterion, don’t double-penalize here.)

6.2.5. Final Score

Level of Detail = Coverage (0–60) + Specifics (0–30) + Conciseness (0–10) − Penalties (0 to −10)

Grade bands

- **Excellent:** 90–100
- **Good:** 75–89
- **Poor:** 0–74

Example

Salient slots chosen (K=5): puppet-A (subject), holding a red ball (action/relation), on-image text “BBS” (text), distinctive blue costume (attribute), stage backdrop (context).

- Coverage: subject (10), action (10), text (10), blue costume (10), backdrop (5 generic) → 45/60
- Specifics (7 correct): “blue,” “red,” “one ball,” “left hand,” “felt material,” “front of stage,” “‘BBS’ text” → 7 × 3 = 21/30
- Conciseness: 34 words, clear → 10/10
- Penalties: none → 0
  Total = 45 + 21 + 10 − 0 = 76 → Good

6.3. Faithfulness / Low Hallucination — Rubric (0–100)

Faithfulness / Low Hallucination judges whether every claim in the caption is truly supported by the image. We begin from a fully faithful baseline and deduct for anything invented or overstated: made-up primary or secondary objects, fabricated or misread on-image text, incorrect attributes or spatial relations, and speculative inferences or internal contradictions. This keeps the focus on image-grounded evidence rather than guesswork—rewarding captions that stick to what’s visible and penalizing those that add, distort, or over-claim details. The result is a clear 0–100 snapshot of how reliably a model describes what’s actually there.

6.3.1. Invented Objects or On-Image Text (max −60)

- **Primary object invented** (claims a main subject that isn’t visible): **−20** each
- **Secondary object invented** (minor/background item not visible): **−10** each

- **On-image text fabricated/misread** (claims text that isn't there or grossly wrong): **−10** each

6.3.2. Fabricated Attributes or Relations (max −25)

- **Wrong key attribute** (e.g., says "red" when it's blue; says "wood" when it's metal): **−5** each
- **Wrong spatial/ownership relation** (e.g., "on top of," "holding," "left/right" contradicted by image): **−6** each

6.3.3. Unsupported Inferences (max −15)

- **Speculative claim** beyond what's visible (e.g., "preparing for a concert," "brand is X" with no evidence): **−5** each

6.3.4. Internal Contradictions / Overclaiming (max −10)

- **Self-contradiction** within the caption (e.g., "no text" then "logo says BBS"): **−5** each
- **Over-certain wording despite obvious ambiguity** (asserts details that are visibly unclear): **−5** each

6.3.5. Final Score

```
Faithfulness = 100 - (A + B + C + D)
```

Grade bands

- **Excellent:** 90–100 (no hallucinations; all claims image-grounded)
- **Good:** 75–89 (rare, minor slips that don't change meaning)
- **Poor:** 0–74 (frequent fabrications or contradictions)

6.4. Speed / Latency — Practical Rubric (0–100)

Speed / Latency captures how fast and how steady a model feels on the hardware you plan to use, so you can tell if it's practical for rehearsal or live, interactive work. We warm up once, then send repeated single-image caption requests at a baseline and a higher resolution, timing each end-to-end to get a typical response and a sense of worst-case spikes. We look for quick, predictable responses at the baseline, limited slowdowns as images get larger, and no outsized "first call" delays. The outcome is a clear readiness signal: does this setup feel responsive and consistent enough on your target machine to trust in real workflows?

“Is this model fast and stable enough, on our hardware, for live rehearsal or interactive use?”

1. Run a small timing test in a repeatable way,
2. Compute a few simple stats (median and p95), and
3. Convert those stats into a 0–100 score.

Test setup (do this the same way for every model/config)

- Hardware: Note exactly what you’re using (e.g., “RTX 4070 12GB” or “8-core laptop CPU”).
- Batch size: Always 1 (one image per request).
- Prompt: Use the same short caption prompt every time (e.g., “Describe this image briefly.”).
- Images and sizes: Pick a small set of representative BBS images. For each model/config, test at 448 px (baseline) and 896 px (scaled).
- Warm-up: Make one request first and ignore that time (caches compile, weights get hot, etc.).

Keep everything else constant: precision, context length, quantization settings, etc., for the duration of the test.

What “30 runs” means (per condition)

- One run = send one image with the fixed prompt and measure the time from send to full caption received.
- 30 runs = repeat that exact request 30 times in a row (same image size, same prompt).
- Do this for 448 px and again for 896 px.
- Record all 30 times (in seconds) for each condition.

Why 30? It’s enough samples to smooth out randomness and lets you compute two useful stats:

- Median (p50): line up your 30 times from fastest to slowest; p50 is the middle one. This is your “typical” latency.
- p95: the value that 95% of runs are faster than. This captures “bad spikes” in latency.

6.4.1. Converting the timings into a score

You’ll compute three sub-scores (A, B, C) and possibly one penalty. Then you add them up.

6.4.2. Baseline Speed at 448 px (p50) — 0–70 points

Pick the table that matches your target hardware. If you intend to run on a consumer GPU, use the GPU table; if you’re CPU-only, use the CPU table.

GPU (consumer 16–24 GB, e.g., RTX 3060–4070)

- p50 ≤ 0.8 s → 70
- ≤ 1.2 s → 60
- ≤ 1.8 s → 45
- ≤ 2.5 s → 30
- > 2.5 s → 15

CPU (8-core laptop/desktop)

- p50 ≤ 1.8 s → 70
- ≤ 2.5 s → 60
- ≤ 3.5 s → 45
- ≤ 5.0 s → 30
- > 5.0 s → 15

How to pick: Score once using the hardware you actually plan to use in rehearsal/live. If you're comparing both CPU and GPU deployments, keep two separate Speed/Latency scores.

6.4.3. Stability (how spiky is it?) at 448 px — 0–20 points

Compute TF = p95 / p50 using the 30 runs at 448 px. This tells you how much the tail slows down versus the median.

- TF ≤ 1.30 → 20 (very steady)
- ≤ 1.60 → 12 (small spikes)
- ≤ 2.00 → 6 (noticeable spikes)
- > 2.00 → 0 (too spiky for comfort)

Example: If p50 = 1.10 s and p95 = 1.43 s, TF = 1.43 / 1.10 = 1.30 → 20 pts.

6.4.4. Resolution Scaling (896 px vs 448 px) — 0–10 points

Now see how much slower it gets when you increase image size. Compute:
Scale = p50(896) / p50(448)

- Scale ≤ 2.0 → 10 (scales well)
- ≤ 2.5 → 5 (acceptable)

- > 2.5 → 0 (degrades too much)

Example: p50 at 448 px = 1.1 s; p50 at 896 px = 2.0 s → Scale = 2.0 / 1.1 ≈ 1.82 → 10 pts.

Optional one-time penalty

Cold start penalty (−5) if either:

- Model load/init takes > 30 s, or
- The first response (after load) is > 3× slower than the steady-state p50.

(This captures “we can’t quickly get going” issues.)

6.4.5. Final score

Speed/Latency = A (0–70) + B (0–20) + C (0–10) − penalty (0 or 5)

Grade bands

- Excellent: 90–100 → Feels interactive and consistent across sizes
- Good: 75–89 → Generally fast; occasional slowouts are okay for rehearsal
- Poor: < 75 → Too slow or too spiky for practical workflows

Worked example (step by step)

Setup: RTX 4070, batch=1, prompt “Describe this image briefly.” Warm-up once, then measure.

448 px (30 runs): times sorted → p50 = 1.10 s, p95 = 1.45 s

- A (GPU table): p50 ≤ 1.2 s → 60 pts
- B: TF = 1.45 / 1.10 = 1.32 → 12 pts
- 896 px (30 runs): p50 = 2.00 s
    - C: Scale = 2.00 / 1.10 = 1.82 → 10 pts
- Penalty: model loads in 18 s; first response is 2.5× p50 → no penalty

Total: 60 + 12 + 10 − 0 = 82 → Good

6.5. Ease of Use — Quantified Rubric (0–100)

Ease of Use captures how quickly and smoothly a model goes from “clone” to a working caption on a clean machine—and how reliably it stays usable after that. Reviewers start from a fresh environment and follow only the official docs to reach a first caption, noting how clear the steps are, how many actions it takes, and whether the “hello world” example runs without edits. They then check installation reliability across machines, whether CPU/GPU paths are documented and accurate, and whether examples beyond the basics actually work. Practical controls (image size, device, batch, context/precision, paths) and a usable CLI or Python API are expected, along with a straightforward fine-tuning path (e.g., LoRA/QLoRA) that includes configs and logging. Packaging and ecosystem support (model card, pip/transformers or Docker) should make setup routine; undocumented blockers or doc/code mismatches count against ease of use. The result is a clear sense of how approachable, reproducible, and integration-friendly the model is for a small team.

6.5.1. Quickstart to First Caption — 0–40 pts

Score using the first successful run from a clean env (Conda/venv or Docker).

6.5.1.1. Docs completeness (install → run) — 0–10

- 10 = clear install + run steps with commands shown
- 5 = partly clear; you must infer 1–2 steps
- 0 = missing/unclear

6.5.1.2. Time to first caption — 0–10

- 10 = ≤ 15 min
- 7 = 16–30 min
- 4 = 31–60 min
- 0 = > 60 min / not reached

6.5.1.3. Number of steps (typed commands/edits) — 0–10

- 10 = ≤ 5 steps
- 7 = 6–9 steps
- 4 = 10–14 steps
- 0 = ≥ 15 steps

6.5.1.4. “Hello world” script/notebook runs as-is — 0–10

- 10 = runs without edits
- 5 = minor edit (path/typo)
- 0 = fails / multiple edits

Subtotal A: 0–40

6.5.2. Install Reliability — 0–25 pts

6.5.2.1. One-command env or Docker works — 0–8

- 8 = works as written
- 4 = works with 1–2 version pins
- 0 = fails / major fixes

6.5.2.2. Dependency friction — 0–7

- 7 = none (no extra pins/patches)
- 4 = light (≤ 2 pins)
- 0 = heavy (> 2 pins or code patch)

6.5.2.3. CPU & GPU paths documented and accurate — 0–5

- 5 = both present and correct
- 2 = only one path reliable
- 0 = unclear/missing

6.5.2.4. First-try success rate across two setups — 0–5

- 5 = 2/2 succeed
- 3 = 1/2
- 0 = 0/2

Subtotal B: 0–25

6.5.3. Examples & Inference Controls — 0–20 pts

6.5.3.1. Working task example beyond “hello” (e.g., VQA or caption batch) — 0–8

- 8 = provided and runs as-is
- 4 = minor edits needed
- 0 = absent/broken

6.5.3.2. Basic runtime controls documented & working — 0–8

(Any 4 of these = full credit): image size, device (CPU/GPU), batch, max tokens/context, dtype/precision, path flags

- 8 = ≥ 4 options shown + work
- 4 = 2–3 options
- 0 = 0–1 option

6.5.3.3. Usable interface — 0–4

- 4 = clear CLI and/or Python API with code snippet
- 2 = minimal/partial
- 0 = none

Subtotal C: 0–20

6.5.4. Fine-Tuning Readiness — 0–10 pts

6.5.4.1. LoRA/QLoRA example with command + expected VRAM — 0–6

- 6 = step-by-step + hyperparams + VRAM notes
- 3 = partial (missing key bits)

- 0 = absent

6.5.4.2. Configs & logging (yaml/json + seed + loss/metrics) — 0–4

- 4 = present and usable
- 2 = partial
- 0 = absent

Subtotal D: 0–10

6.5.4.3. Tooling & Ecosystem — 0–5 pts

- 5 = Hugging Face card (releases/tagged versions) and `pip/transformers` or Docker image provided
- 3 = only one of the above
- 0 = neither

Subtotal E: 0–5

6.5.4.4 Penalties (apply once, cap −10)

Undocumented blocker (README says it works but it doesn't; needed code change or GitHub issue): −5 each, cap −10

README/code mismatch (paths/flags wrong): −3 each, cap −9
(but still cap total penalties at −10)

Final Score

Ease of Use = A (0–40) + B (0–25) + C (0–20) + D (0–10) + E (0–5) − penalties (0 to −10) → clamp 0–100

Grade bands

- **Excellent:** 90–100 — "Clone, install, run, fine-tune" is smooth
- **Good:** 75–89 — Minor friction; fine for most users
- **Poor:** < 75 — Fragile or time-consuming

7. Sources

Qwen2-VL / Qwen2.5-VL

- Qwen2-VL-7B-Instruct — official Hugging Face model card (license & usage). Hugging Face
- Qwen2-VL-7B (base) — official card (family overview). Hugging Face

- Qwen2-VL-7B-Instruct — license file (Apache-2.0). Hugging Face
- Qwen2.5-VL-7B-Instruct — official model card (current variant; for awareness). Hugging Face+1

LLaVA-OneVision-1.5 (4B/8B)

- Official GitHub (framework, data, training/eval). GitHub
- arXiv paper for OneVision-1.5 (framework/results). arXiv+1
- HF collection for OneVision-1.5 assets/datasets. Hugging Face+1
- Earlier OneVision paper (context/history). arXiv

Idefics-2-8B

- Official Hugging Face model card (Apache-2.0, checkpoints). Hugging Face+1
- HF blog: "Introducing Idefics2" (overview + usage). Hugging Face
- Transformers docs page for Idefics2. Hugging Face
- Quantized AWQ checkpoint (deployment option). Hugging Face

Phi-3.5-Vision-Instruct

- Official Hugging Face model card (MIT license). Hugging Face+1
- License file (MIT). Hugging Face
- ONNX variant (for export/deployment reference). Hugging Face

InternVL2-8B

- Official Hugging Face model card (family details, usage). Hugging Face
- AWQ quantized release (INT4; LMDeploy notes). Hugging Face+1
- Project GitHub (updates; AWQ support context). GitHub
- (Optional) InternVL 2.5 AWQ (newer variant, out-of-scope but related). Hugging Face

Evaluation & Leaderboards (for triangulation / replication)

- VLMEvalKit (OpenCompass) — official repo (one-command evals). GitHub
- VLMEvalKit paper/entries (overview). Hugging Face+1
- Open VLM Leaderboard (OpenCompass space). Hugging Face
- OpenCompass multi-modal leaderboards collection. Hugging Face

8. Version History

Version	Created on	Created by
1.0	10/28/2025	Kazi Islam
