Do Vision-Language Models Measure Up? Benchmarking Visual Measurement Reading with MeasureBench

Supplementary Material

1. Model Training

1.1. Training details

We employ reinforcement finetuning (RFT) on the synthesis datasets with 3900 images by reinforcement learning framework verl. Following Deepseek-R1, we employ GRPO with a format reward function to optimize the model to output the thinking process within "`<think>...</think>`". Training is performed on 8×H100 GPUs for 15 epochs with a global batch size of 128 and a learning rate of 1×10^{-6}, and a rollout number of 8.

1.2. Soft-margin Reward

We also conduct experiments with a soft-margin reward that grants partial credit to predictions near the target interval. For the numeric component of the answer, define the distance to the interval as

$$d(\hat{y}, I) = \begin{cases} 0, & \hat{y} \in [l, r], \\ \min(|\hat{y} - l|,\ |\hat{y} - r|), & \text{otherwise}, \end{cases} \tag{1}$$

and the margin

$$m(I) = \begin{cases} r - l, & r > l, \\ 0.05\, l, & \text{otherwise}, \end{cases} \tag{2}$$

with a small constant $\varepsilon > 0$. We define a linearly decaying partial-credit score

$$s_{\text{sm}}(\hat{y}; I) \ = \ \tfrac{1}{2}\ \max\Big\{0,\ 1 - \tfrac{d(\hat{y}, I)}{m(I) + \varepsilon}\Big\}. \tag{3}$$

The soft-margin reward replaces the value term by taking the better of exact correctness and soft credit, while keeping the formatting term unchanged:

$$R_{\text{soft}} \ = \ \alpha\ \max\big\{\, c_{\text{all}},\ s_{\text{sm}}(\hat{y}; I)\,\big\} \ + \ (1 - \alpha)\, c_{\text{fmt}}. \tag{4}$$

This keeps rewards consistent with evaluation when the answer is exact, while giving informative feedback to near-miss predictions.

As shown in Table 1, the soft-margin reward yields comparable performance to the original hard-margin reward after RFT on both real-world and synthetic subsets. This suggests that the original reward design is already effective, while the soft-margin variant provides an alternative that may be more suitable in scenarios where near-miss predictions are common.

2. Examples of synthesized images

Here we provide additional examples of synthesized images of measuring instruments generated by our framework. Each generator in our framework is expected to render an image of an instrument with similar appearance along with random readout. In Figure 1, 2D images are rendered by offline-only libraries like Pillow, NumPy and Matplotlib, 3D images are rendered by Blender which is more realistic.

3. 3D modeling with Blender

To construct a large collection of measurement-related 3D assets, we use **Blender (v4.2)** in combination with publicly available online repositories. The procedure was as follows.

3.1. Asset retrieval

We integrate the **BlenderKit** plugin into Blender to access free 3D assets, including models, HDRs, and materials. For categories underrepresented in BlenderKit (e.g., cylinder, hygrometer), we also retrieved models from **Sketchfab**. The queries included *watches, clocks, scales* and *rulers, thermometers*, covering both **pointer-based** and **linear-scale instruments**.

3.2. Model normalization

1. **Pointer-based models** (e.g., clocks, scales): In many assets, the pointer was not initially aligned with the zero position. We manually rotated the pointer to zero and reset its transformations (rotation along the x, y, z axes set to 0).
2. **Linear-scale models** (e.g., thermometers): For these, we determined the minimum-maximum mapping on the scale and adjusted the geometry proportionally so that the linear transformations of pointer correctly represented measurement values.

3.3. Contextual scene augmentation

Some models only represented the measurement instrument itself, which led to unrealistic renderings when the pointer indicated a nonzero value. To improve semantic consistency, we augmented scenes with additional objects:

- **Scales**: To avoid showing a dial reading *1 kg* with an empty plate, we placed an additional object (e.g., a fruit model, such as dragon fruit) on the weighing surface.

Model/Dataset	Overall	Value	Unit
Qwen2.5-VL-7B (Real-world)	14.6	15.0	93.4
Qwen2.5-VL-7B+GRPO (Real-world)	19.7 (+34.9%)	20.4 (+36.0%)	92.3 (-1.2%)
Qwen2.5-VL-7B+GRPO-soft (Real-world)	19.6 (+34.2%)	20.4 (+36.0%)	91.9 (-1.6%)
Qwen2.5-VL-7B (Synthetic)	10.9	11.5	88.5
Qwen2.5-VL-7B+GRPO (Synthetic)	35.2 (+222.9%)	35.6 (+209.6%)	96.7 (+9.3%)
Qwen2.5-VL-7B+GRPO-soft (Synthetic)	35.3 (+223.9%)	35.6 (+209.6%)	98.0 (+10.7%)

Table 1. Results of Qwen2.5-VL-7B variants with GRPO on real-world and synthetic subsets.

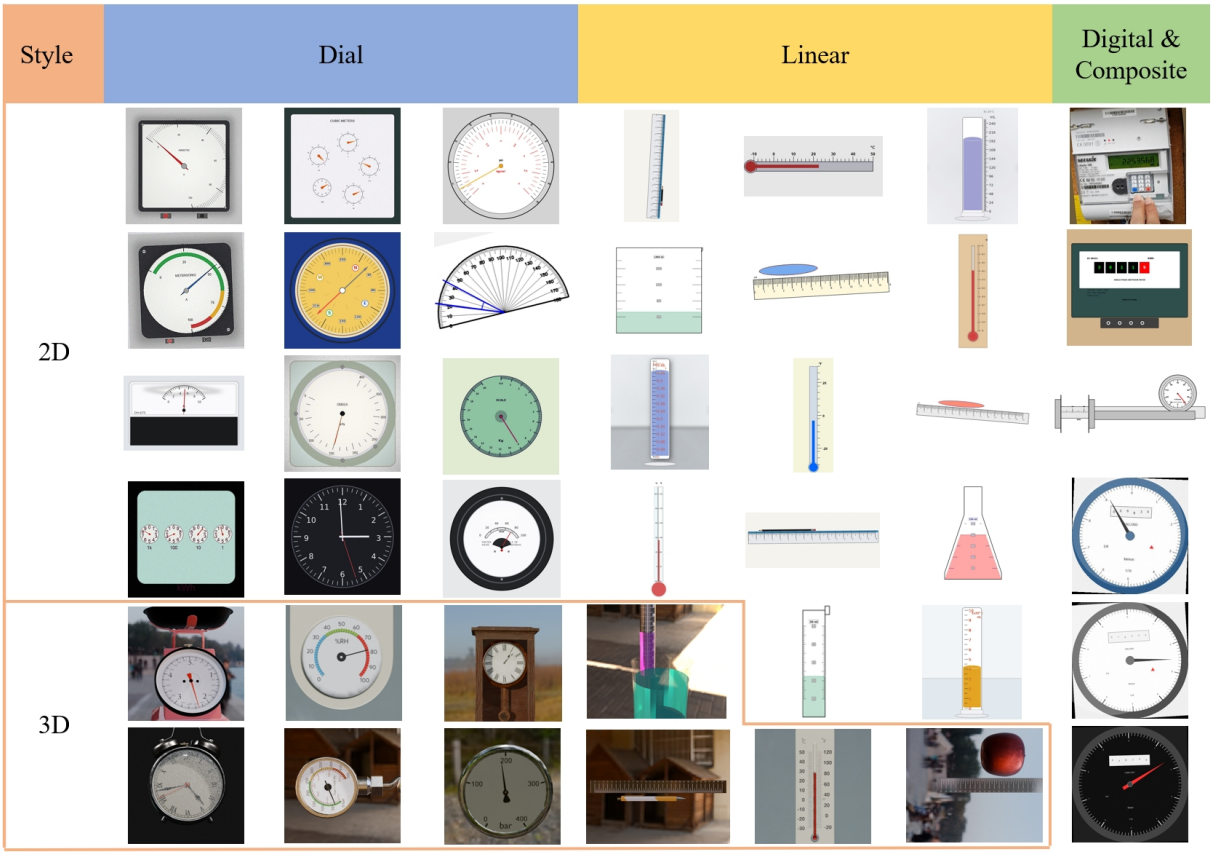

Figure 1. Additional examples of synthetic measuring instruments generated by our pipeline.

- **Rulers**: Since rulers measure relative length, we included a reference object (a pen). The pen was rescaled and positioned alongside the ruler, allowing queries such as "How long is the pen?" to be grounded in the rendered image.

These contextual additions ensured that pointer readings were visually consistent with the surrounding scene, enhancing dataset realism, and reducing ambiguity for vision-language evaluation.

3.4. Pointer rotation control

Pointer manipulation was automated with Blender's Python API.

- **Clocks**: For clocks, rotation angles were computed directly from the target hour, minute, and (optionally) second values:

```
second_angle = math.radians(target_second * 6)
minute_angle = math.radians(target_minute * 6 + target_second * 0.1)
hour_angle   = math.radians((target_hour % 12) * 30 + target_minute * 0.5)
```

The axis of rotation varied across different models (i.e. whether Oxy, Oxz, or Oyz). For

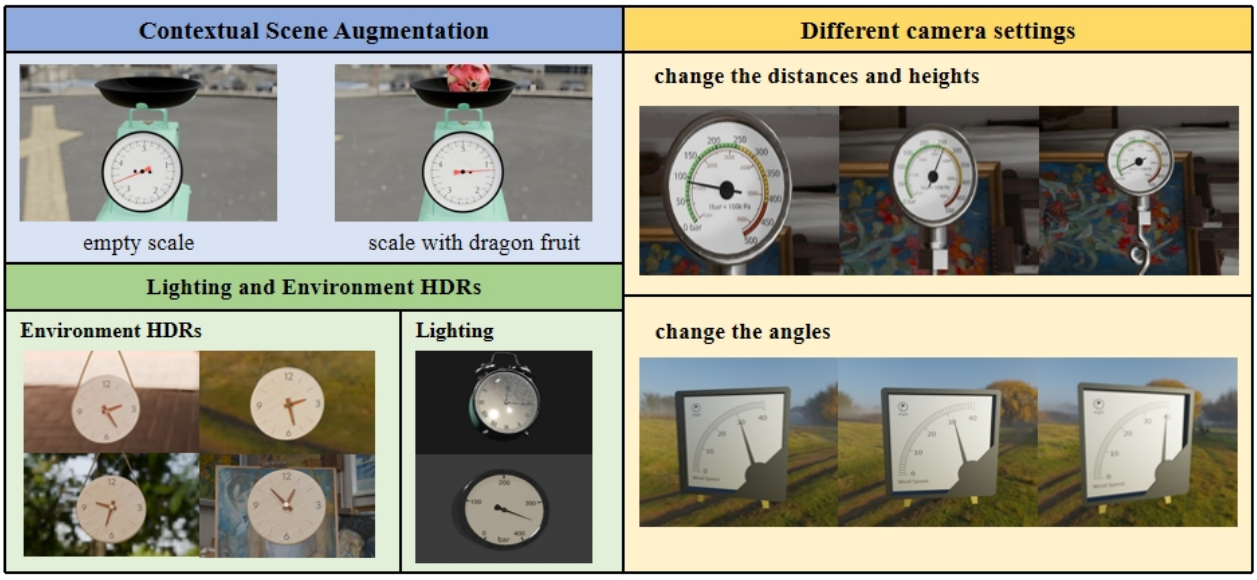

Figure 2. Examples of augmentation strategies applied during 3D model acquisition and preparation with Blender (v4.2).

example, a clock's hour hand can be controlled with: `hour_hand.rotation_euler = (0, 0, -hour_angle)`. However, depending on the model, the rotation angle might be applied to the first or second component of the Euler tuple rather than to the third.

- **Other dials** (e.g., hygrometers): For these, the degree of pointer rotation depends on the specific model geometry. We first check for the maximum rotation angle ($max.rot.deg$) that corresponded to the maximum scale value, and set pointer positions linearly:

```
max_rot = math.radians(max_rot_deg)
rot_z = min_rot + (humidity-min_humidity)
    / (max_humidity-min_humidity)
            * (max_rot-min_rot)
```

This approach is generalized to other instruments with linear or semi-linear dial mappings.

If the geometry of the model used a nonstandard orientation, we rotated the entire object to align it with the desired axis.

3.5. Camera alignment

Since the dial panels of many models were not centered at the origin, we applied offsets to position the camera such that it directly faces the dial. Camera distance and angle were tuned empirically to maximize legibility of the dial face and pointer. For small-scale instruments, shorter distances and narrower angle ranges provided clearer renderings, whereas larger instruments benefited from wider perspectives.

3.6. Lighting and environment HDRs

To ensure consistent illumination across renderings, we used two strategies depending on the dataset requirements:

- **HDR environment maps**: For most models, we initialized scenes with background environment maps (.exr files), either using Blender's built-in HDRIs or downloading additional ones via BlenderKit. These provided realistic lighting and surface reflections. HDRs were first manually configured and later automated using Python.
- **Direct light sources**: For cases where a clean background was preferred, we disabled HDRs and instead added light objects from different positions (e.g., point lights or area lights). This clearly illuminated the dial while leaving the background neutral.

3.7. Rendering execution

Scripts were executed either directly within Blender's Scripting panel or externally via Python (importing the bpy module) in an IDE such as Visual Studio Code. This flexibility enabled large-scale automated rendering of models across different instrument categories.

Figure 2 provides illustrative examples of the augmentation strategies described above.

4. Additional Experiments

4.1. Impact of Detailed Instructional Prompts

We investigate whether providing step-by-step reading instructions tailored to different instrument types can improve model performance. The instruction includes design-level

common rules shared by instruments of the same readout design and instrument-specific guidance. As shown in Table 2, even with explicit guidance on how to read the instruments, the performance gain on real-world MeasureBench is very limited. Adding in-context examples does not seem to help either, suggesting that the bottleneck lies in fine-grained visual perception rather than a lack of procedural knowledge.

Model	Overall	Value	Unit
Gemini-2.5-Pro	31.8(+1.6)	32.2(+1.5)	96.8(+0.6)
GPT-5	19.9(+0.1)	20.1(+0.2)	97.5(+1.5)
Qwen2.5-VL-7B	14.6(+0.0)	15.5(+0.5)	93.2(-0.2)

Table 2. Impact of more detailed instructional prompts on MeasureBench (real-world images). Numbers in parentheses indicate the change compared to the default prompt.

4.2. Generalization on General Benchmarks

To assess potential negative transfer from reinforcement finetuning on synthetic measurement data, we evaluate Qwen2.5-VL-7B before and after GRPO training on popular general-purpose benchmarks. As shown in Table 3, GRPO training on synthetic data yields comparable performance on these benchmarks with no degradation, indicating that the model retains its general capabilities.

Model	MMMU	MMMU-Pro	MathVista	TextVQA
Qwen2.5-VL-7B	52.44	37.75	**69.40**	**84.74**
+ GRPO	**54.33**	**37.86**	69.00	84.24

Table 3. Qwen2.5-VL-7B results on general benchmarks before and after GRPO training on synthetic measurement data.

4.3. Comparison with Supervised Fine-Tuning

We compare GRPO with supervised fine-tuning (SFT) on the same synthetic dataset. We experiment with two SFT response formats: (i) direct answer only (e.g., "4.3 cm", "6.5 L"), and (ii) rationale + final answer, where GPT-5.2 is used to generate step-by-step rationales given the image and the ground-truth readout. As shown in Table 4, while SFT substantially improves in-domain synthetic performance, it degrades accuracy on real-world images, indicating severe overfitting to synthetic patterns. In contrast, GRPO improves on both synthetic and real-world subsets, demonstrating better generalization.

5. Extended Analysis

We provide further analysis of model behavior on specific challenging cases, as well as statistical distributions of numerical outputs.

Model	Real-world			Synthetic		
	Ovr	Val	Unit	Ovr	Val	Unit
Qwen2.5-VL-7B	14.6	15.0	**93.4**	10.9	11.5	88.5
+ GRPO	**19.7**	**20.4**	92.3	**35.2**	**35.6**	96.7
+ answer SFT	12.5	13.4	89.5	29.7	30.9	**96.8**
+ rationale SFT	8.8	9.1	91.3	21.7	22.0	95.8

Table 4. Comparison of GRPO and SFT training on Qwen2.5-VL-7B. SFT overfits to synthetic patterns, degrading real-world performance, while GRPO improves both.

5.1. Reading of complex measuring instruments

Complex measuring instruments with composite readout designs or multiple dials and pointers remain highly challenging for current VLMs: an error at any step of fine-grained visual perception or reading interpretation will propagate to an incorrect final result. Figure 3 shows a multi-dial electricity meter with five dials, each with a pointer. To read the meter, one should note the position of each pointer and record the numbers from **left to right**, remembering that adjacent dials rotate in opposite directions. If a pointer is between two numbers, the **lower** number is recorded, unless it is between 9 and 0, in which case 9 is recorded. We present the results of three models, although all of them correctly describe the reading procedure, they still struggle to localize the pointer positions, misreading almost all pointers and thus producing incorrect final readings.

5.2. Example of correct "guessing"

We observed that some correct answers may arise from guessing rather than accurate visual understanding. Figure 4 shows an ammeter whose pointer indicates a reading of 20 A. Both GPT-5 and GPT-5-Mini provide detailed reasoning steps before giving their final answers. GPT-5 incorrectly estimates the reading as 26 A, whereas GPT-5-Mini is clearly uncertain and essentially guesses 20 A, which happens to be correct.

6. The "10:10" phenomenon

We found an interesting phenomenon that many models tend to answer "10:10" when reading clock times, regardless of the actual time shown in the image. During the real-world data collection, we deliberately avoid including images with "10:10", and in the synthesis process,clock times are uniformly sampled. As a result, there are few ground-truth of "10:10" in our benchmark.

However, as shown in Table 5, we calculate the proportion of string "10:10" in the models's answers on clock images. It is surprising that the powerful open-source model Qwen2.5-VL-72B-Instruct outputs "10:10" for 72.88% of real-world clock images and 50.74% of synthetic clock im-

Question: What is the reading of the electricity meter?
Type: Electricity Meter
Reading: 8504 ± 1 kilowatt hours or kWh
Reading Instructions: Read the dials from left to right and the adjacent dials rotate in opposite directions. If a pointer is between two numbers, record the lower number

Claude-opus-4.1-thinking
Answer: 50190 kWh

Reason: I need to read the electricity meter shown in the image. This is an ABB kilowatt hour meter with 5 dial displays. Reading the dials from left to right: First dial (0-9): The pointer is between 5 and 6, closer to 5, so it reads 5. Second dial (0-9): The pointer is between 0 and 1, closer to 0, so it reads 0. Third dial (0-9): The pointer is between 1 and 2, closer to 1, so it reads 1. Fourth dial (0-9): The pointer is between 9 and 0, closer to 9, so it reads 9. Fifth dial (0-9): The pointer is between 0 and 1, closer to 0, so it reads 0. When reading analog meters with multiple dials, we read them from left to right, and if a pointer is between two numbers, we use the lower number unless it has passed it. Final Answer: 50190 kWh.

Qwen3-vl-235b-a22b-thinking
Answer: 17306 kilowatt hours

Reason: To determine the reading of the electricity meter, we analyze each dial from left to right. The key rule for analog meters is to take the last number the pointer has passed(i.e., the lower number if between two values). First dial (leftmost): The pointer is between 1 and 2 → 1 . Second dial: The pointer is between 7 and 8 → 7 . Third dial: The pointer is between 3 and 4 → 3. Fourth dial: The pointer is between 0 and 1 → 0 . Fifth dial (rightmost): The pointer is between 6 and 7 → 6. Combining these readings gives 17306. The unit specified on the meter is kilowatt hours. Final Answer: 17306 kilowatt hours.

GPT-5
Answer: 12198 kWh

Reason: Analyzing electromechanical meter dials. (… omitted), reading from left to right. I notice there are five dials in the image. The leftmost dial seems just past the number 1, with numbers arranged in a clockwise manner. (… omitted)If the pointer is between two numbers, we take the lower one; if it's exactly on a number, I need to check the next dial to see if it hasn't passed zero. (… omitted)The leftmost pointer looks to be around 1.1, so I'd record a 1. The second dial appears to be between 2 and 3, leading me to choose 2. The third dial seems just before 2, suggesting I take 1. The fourth dial is around 9, and the fifth is likely between 8 and 9, so I'd record 8. I'll finalize with 12198 kWh.

Figure 3. Results comparison on an electricity meter

Question: What is the reading of the ammeter?
Type: Ammeter
Reading: 20 A

GPT-5
Answer: 26 A

Reason (… omitted) a numeric value with units like \"26 A.\" There are 10 small divisions between 20 and 40, which suggests each division equals 2 A. The pointer seems to be around the third division above 20, which leads me to think it's 26 A. However, it could also be closer to 28 A. The exact position is tricky to pinpoint, possibly around 27 A, (… omitted) I think I should approximate it to 26 A, (… omitted) It appears slightly left of the tick for 30, maybe around 27 A (… omitted) I'm ready to finalize the ammeter reading as 26 A. (… omitted)

GPT-5-Mini
Answer: 20 A

Reason: **Reading the ammeter**
I need to determine the reading on the analog ammeter, which has a scale from 0 to 100, labeled as A. The needle seems to be pointing slightly above 20, maybe at 22, but it's hard to tell. It looks like it might actually be at 20, just left of the mark. I think I should go with 20 A for the answer, even if it could be 18 A. So, I'll finalize with: \"Final Answer: 20 A

Figure 4. An example where GPT-5 answers incorrectly while GPT-5-Mini guesses the correct reading.

ages. The frontier commercial model GPT-5 also predicts "10:10" in more than 20% of its answers on real-world clock images. This bias likely stems from training data, where clocks are frequently depicted at "10:10" for aes-

Model	Real-world	Synthetic
Qwen2.5-VL-72B-Instruct	72.88%	50.74%
GPT-5-Mini	29.66%	7.78%
Claude-Sonnet-4	26.27%	16.30%
Qwen2.5-VL-7B-Instruct	23.73%	15.56%
Qwen2.5-VL-32B-Instruct	21.19%	8.89%
GPT-5	20.34%	6.30%
Mistral-medium-3.1	16.95%	4.07%
InternVL3.5-38B-thinking	16.10%	12.96%
Claude-Opus-4.1	13.56%	9.63%
InternVL3.5-8B-thinking	12.71%	7.04%
Claude-Opus-4.1-thinking	12.71%	10.37%
Qwen3-VL-235b-instruct	12.71%	12.96%
InternVL3.5-38B	11.86%	10.00%
Qwen2.5-VL-3B-Instruct	11.86%	3.70%
Gemini-2.5-Pro	11.86%	3.33%
Qwen3-VL-8B	7.63%	9.63%
Gemini-2.5-Flash	4.24%	1.11%
InternVL3.5-8B	3.39%	6.30%
Qwen2.5-VL-7B-GRPO	3.39%	1.11%
Grok-4	3.39%	4.81%
Gemini-2.5-Flash-thinking	2.54%	1.48%
LLaMA-4-maverick	0.85%	1.11%
LLaMA-4-scout	0.00%	1.85%

Table 5. Proportion of "10:10" responses on clock images in MeasureBench.

thetic reasons in advertisements and product listings. To further verify this, we examine the answers of Qwen2.5-VL-7B with RFT training on our synthetic dataset, where clock times follow a uniform distribution. The RFT-trained model predicts "10:10" on only 3.39% of real-world clock images, whereas the original Qwen2.5-VL-7B does so on 23.73% of images, indicating that training with a more uniform distribution can effectively mitigate this bias.

7. Spikes distributions at integers

We further analyze the distribution of numeric outputs from different models. Figure 5 and Figure 6 show the spikes at integer values for InternVL3.5 series and Qwen2.5-VL series respectively. We can observe that both models exhibit significant spikes at multiples of ten, the same model series have similar distribution patterns, and thinking mode can not mitigate these spikes. This may be attributed to the models' training data, where round numbers are more frequently represented.

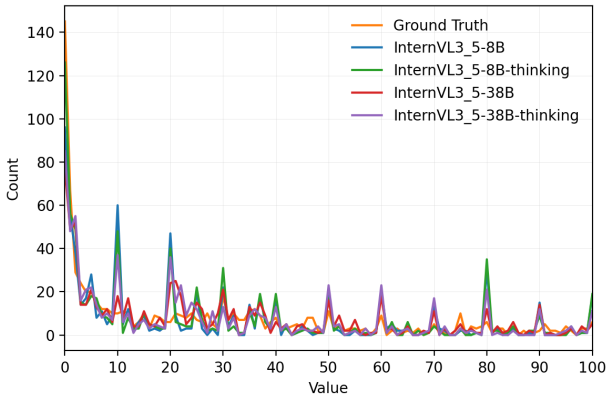

Figure 5. The numeric spikes distribution of InternVL3.5 series on real-world subset.

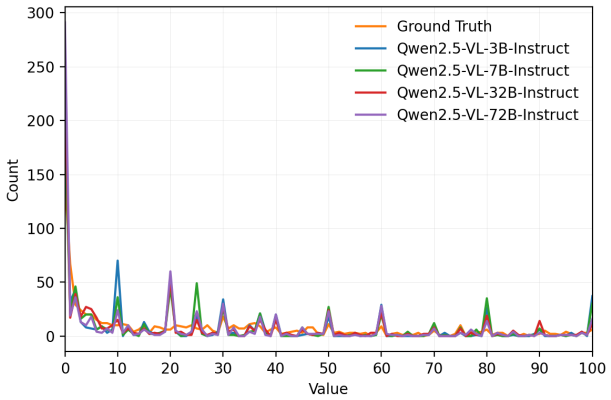

Figure 6. The numeric spikes distribution of Qwen2.5-VL series on real-world subset.