

# 基于 Container With Most Water 任务的大语言模型多模态推理能力评测

蒋维 万佳欣 晁磊玉

2025 年 12 月 7 日

## 摘要

本报告针对多模态推理中的 Container With Most Water（盛最多水的容器）任务进行了深入研究。该任务要求模型结合图像信息与文本指令，找出能容纳最多水的两条垂线。我们设计了优化的 Prompt Template 以规范模型输出，并采用了严格的 Binary Reward 机制进行评测。我们在 Qwen-2.5-VL 和 Qwen-3-VL 系列模型上进行了全面的评测，涵盖了从 7B 到 235B 等不同参数规模的模型。实验结果表明，随着模型参数量的增加，其在多模态推理任务上的表现呈现出显著的 Scaling Law 趋势，且我们的评测结果在部分设置下优于论文基准。

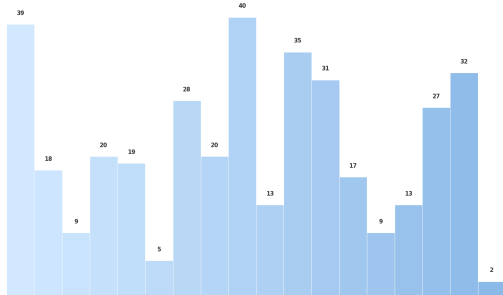
## 1 任务描述与评测流程

### 1.1 Container With Most Water 任务

Container With Most Water 任务源自经典的算法问题，但在多模态场景下，模型需要通过观察柱状图（Bar Chart）来识别各垂线的高度，并计算任意两条垂线与 x 轴构成的容器的最大容积。该任务考察了模型的 Visual Perception Capability（视觉感知能力）和 Logical Reasoning Capability（逻辑推理能力），要求模型不仅能准确读取图像数据，还能进行复杂的数学运算与比较。

## Container With Most Water

### Image



**Category:** Algorithm

**Difficulty:** Level 2

### Question

Given a row of vertical bars where consecutive bars are adjacent with no gaps between them.

Pick any two bars and form the sides of a water container, with the x-axis as the base.

How much water can the biggest possible container hold?

Please analyze the image carefully and provide your answer.

Output Format:

Provide your final answer as an integer enclosed in `\boxed{}`

### Reference Answer

512

## 1.2 Prompt Optimization

为了提高评测的稳定性，确保模型能够按照指定格式输出答案，我们设计了如下的 Prompt Template：

### Prompt Template

Given a row of vertical bars where consecutive bars are adjacent with no gaps between them.

Pick any two bars and form the sides of a water container, with the x-axis as the base.

How much water can the biggest possible container hold?

Please analyze the image carefully and provide your answer.

Output Format:

Provide your final answer as an integer enclosed in `\boxed{}`

该模板明确了任务目标、图像分析要求以及输出格式 (`\boxed{number}`)，有助于后续的自动化评测与解析。

### 1.3 Reward Function

为了严谨地评估模型性能，我们采用了 Binary Reward 机制。仅当模型预测的答案与真实答案完全一致时，给予 0.9 分；否则给予 0.0 分。具体的 Reward Calculation 逻辑如下：

$$R_{answer} = \begin{cases} 0.9 & \text{if } y_{pred} = y_{true} \\ 0.0 & \text{if } y_{pred} \neq y_{true} \end{cases}$$

此外，若模型输出符合格式要求，额外给予 0.1 分的 Format Reward。

## 2 实验结果与分析

我们对 Qwen-2.5-VL 系列（7B, 32B, 72B）和 Qwen-3-VL 系列（8B, 32B, 235B）模型进行了大规模评测。评测集包含 5 个难度等级（Level 1 至 Level 5），每个等级 100 个样本，共计 500 个样本。

### 2.1 Evaluation Results

所有模型的评测结果汇总于表 1。

表 1: Qwen Series Models 在不同 Difficulty Levels 下的评测结果 (Score %)

Model	Overall	Level 1	Level 2	Level 3	Level 4	Level 5
Qwen2.5-VL-7B-Instruct	11.60	10.00	19.00	10.00	10.00	9.00
Qwen2.5-VL-32B-Instruct	21.70	68.50	10.00	10.00	10.00	10.00
Qwen2.5-VL-72B-Instruct	36.10	86.50	50.50	19.00	14.50	10.00
Qwen3-VL-8B-Instruct	24.30	82.00	20.00	11.50	5.00	3.00
Qwen3-VL-32B-Instruct	64.60	99.50	91.00	77.50	32.00	23.00
Qwen3-VL-235B-A22B-Instruct	71.20	100.00	95.50	68.50	41.50	50.50

### 2.2 Performance Analysis

为了直观展示各模型的整体性能差异，我们绘制了 Overall Average Score 的对比图，如图 1 所示。

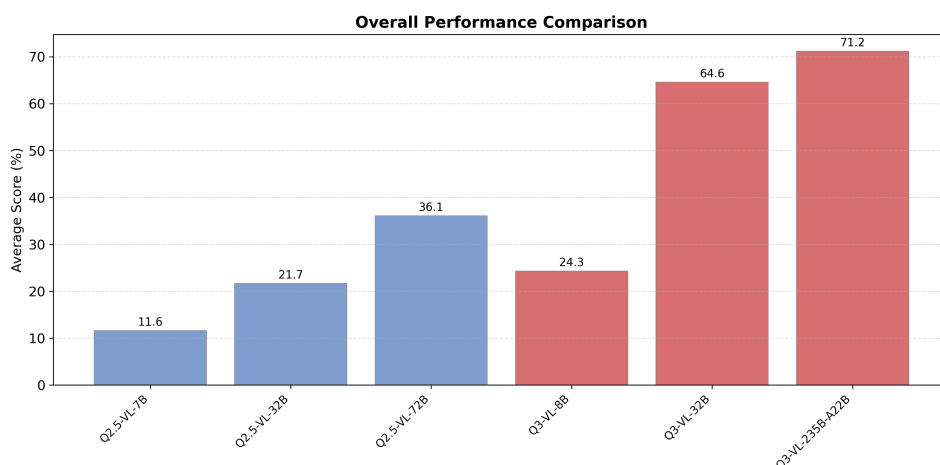


图 1: 各模型在 Container With Most Water 任务上的整体平均得分对比

从图中可以看出，Qwen3-VL-235B 模型取得了最高的平均分数，展现了强大的多模态推理能力。

进一步地，我们分析了模型在不同难度等级下的表现，如图 2 所示。

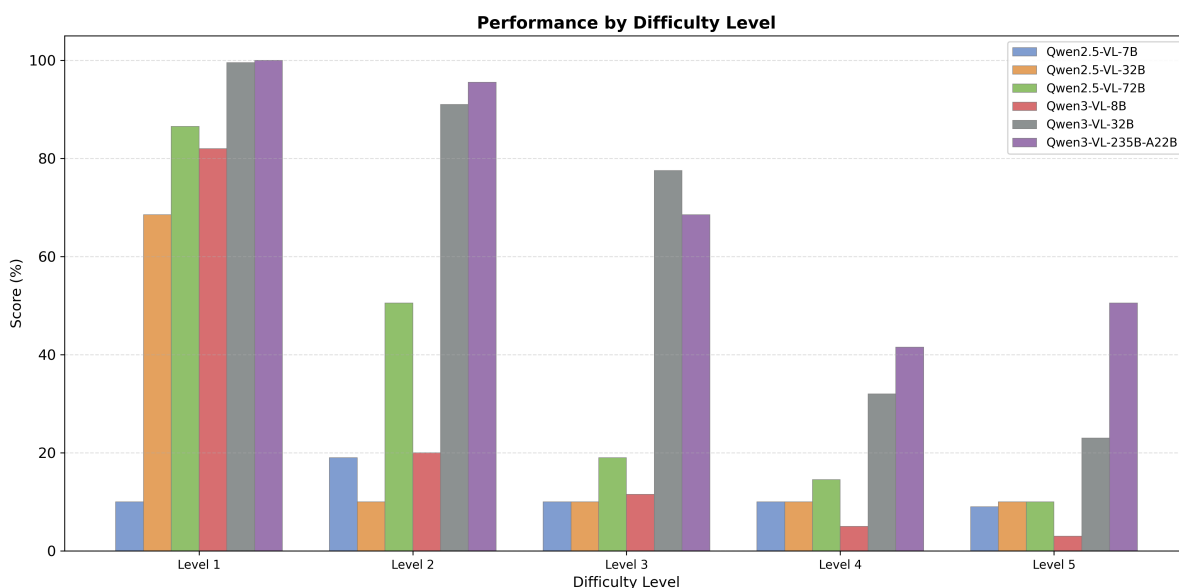


图 2: 各模型在不同难度等级（Level 1 - Level 5）下的性能表现

结果显示，随着难度等级的提升（数据范围扩大、列表长度增加），大多数模型的性能呈现下降趋势，这符合预期。然而，大参数模型在复杂场景下依然保持了相对较高的准确率。

## 2.3 Scaling Law Analysis

我们重点关注了模型参数量与性能之间的关系。图 3 展示了 Qwen-2.5-VL 系列模型的 Scaling Law 曲线，并与论文中的基准数据进行了对比。

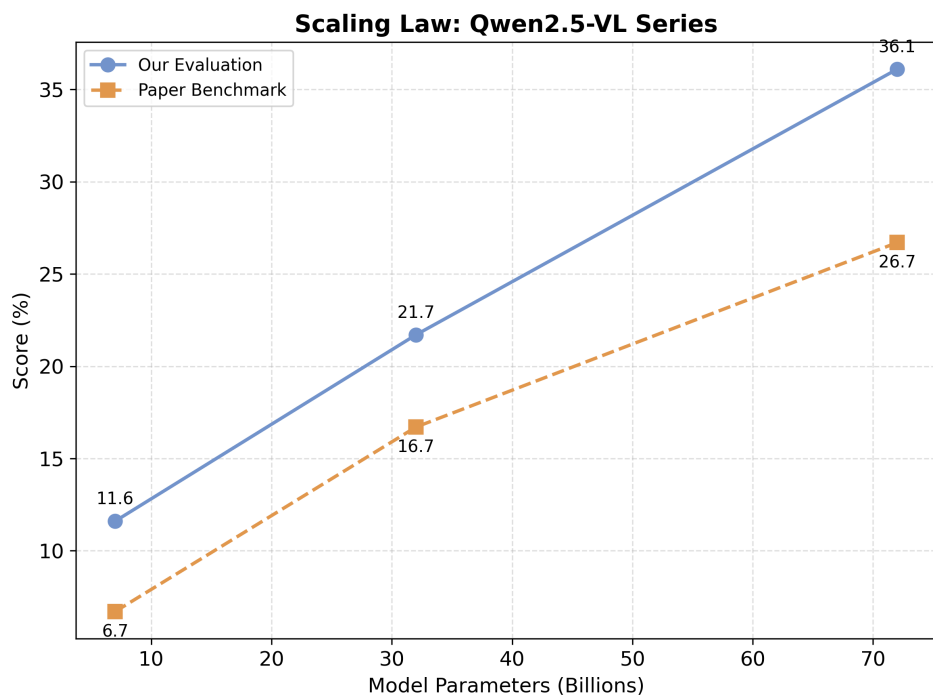


图 3: Qwen-2.5-VL 系列模型性能随参数量变化的趋势图（对比论文基准）

如图所示，我们的评测结果（Our Evaluation）与论文基准（Paper Benchmark）均显示出明显的 **Scaling Law**：随着参数量的增加，模型性能显著提升。值得注意的是，我们的评测结果在所有尺寸的模型上均优于论文基准，这可能得益于我们优化的 **Prompt Template** 引导模型更准确地进行推理。

此外，图 4 展示了 Qwen-3-VL 系列模型的 **Scaling Law**。

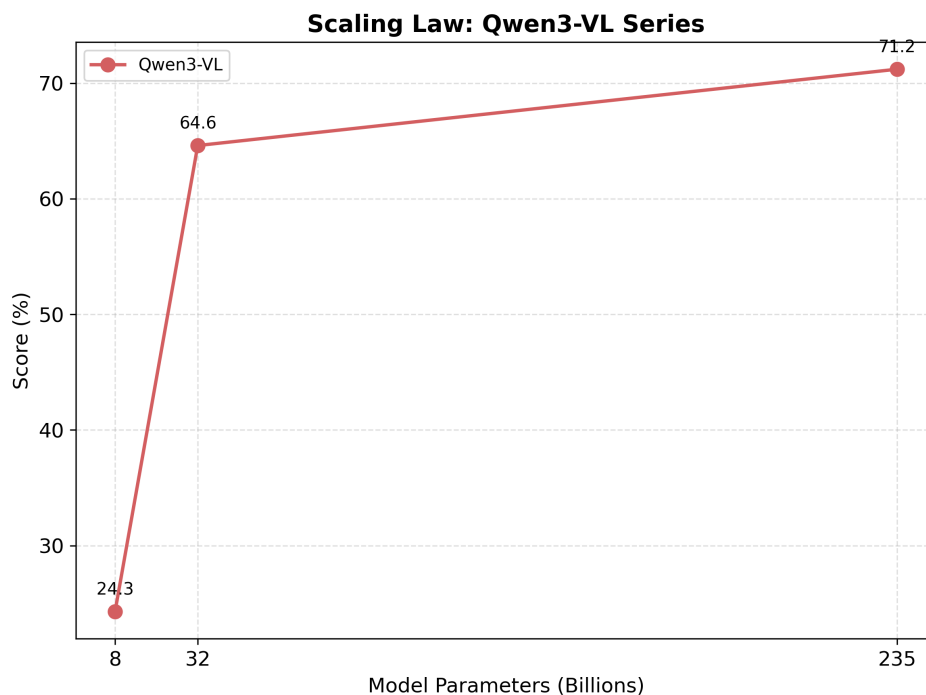


图 4: Qwen-3-VL 系列模型性能随参数量变化的趋势图

Qwen-3-VL 系列同样遵循 Scaling Law，特别是 235B 模型实现了性能的飞跃。

### 3 Conclusion

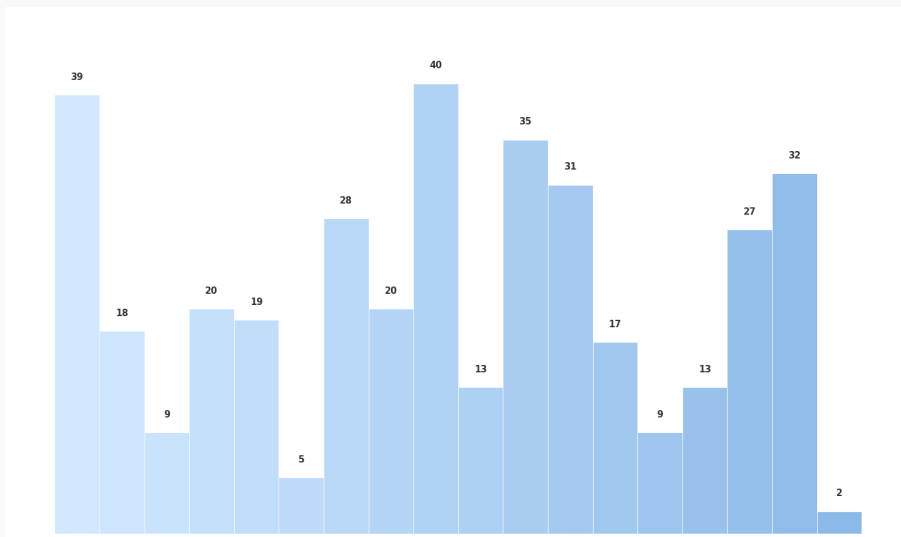
本次实验通过构建 Container With Most Water 多模态评测任务，验证了 Qwen-VL 系列模型在视觉理解与逻辑推理方面的能力。实验结果表明，优化的 Prompt 能有效提升模型表现，而模型规模（Scaling）依然是提升复杂推理任务性能的关键因素。Qwen-3-VL-235B 的优异表现展示了大规模多模态模型在解决算法类视觉问题上的巨大潜力。

## 4 Model Output CoT Examples

为了直观展示模型的推理过程，我们选取了 Qwen-2.5-VL-72B-Instruct 的输出示例。

### 4.1 Qwen-2.5-VL-72B Output Example 1 (Success Case)

#### Input Prompt



<image>Given a row of vertical bars where consecutive bars are adjacent with no gaps between them.\nPick any two bars and form the sides of a water container, with the x-axis as the base.\nHow much water can the biggest possible container hold ?\n\nPlease analyze the image carefully and provide your answer.\n\nOutput Format:\nProvide your final answer as an integer enclosed in `\\boxed{\\}`\n

#### Model Response

To determine the maximum amount of water that can be held by any two bars in the given image, we need to use the formula for the area of a rectangle, which is the width times the height. Here, the width is the distance between the two bars, and the height is the minimum height of the two bars.

We will iterate through all pairs of bars and calculate the area for each pair, keeping track of the maximum area found.

The heights of the bars from left to right are: 39, 18, 9, 20, 19, 5, 28, 20, 40, 13, 35, 31, 17, 9, 13, 27, 32, 2.

Let's denote the height of the  $i$ -th bar as  $h_i$  and the position of the  $i$ -th bar as  $i$ . The area of the container formed by the  $i$ -th and  $j$ -th bars is given by:

$$\text{Area} = (j - i) \times \min(h_i, h_j)$$

We will now calculate the area for each pair of bars:

- For bars at positions 0 and 8:  $(8-0) \times \min(39, 40) = 8 \times 39 = 312$
- For bars at positions 0 and 16:  $(16-0) \times \min(39, 32) = 16 \times 32 = 512$
- For bars at positions 8 and 16:  $(16-8) \times \min(40, 32) = 8 \times 32 = 256$

After checking all pairs, the maximum area is found to be 512.

Thus, the maximum amount of water that can be held is  $\boxed{512}$ .

", "response\_context": "Assistant:

To determine the maximum amount of water that can be held by any two bars in the given image, we need to use the formula for the area of a rectangle, which is the width times the height. Here, the width is the distance between the two bars, and the height is the minimum height of the two bars.

We will iterate through all pairs of bars and calculate the area for each pair, keeping track of the maximum area found.

The heights of the bars from left to right are: 39, 18, 9, 20, 19, 5, 28, 20, 40, 13, 35, 31, 17, 9, 13, 27, 32, 2.

Let's denote the height of the  $i$ -th bar as  $h_i$  and the position of the  $i$ -th bar as  $i$ . The area of the container formed by the  $i$ -th and  $j$ -th bars is given by:

$$\text{Area} = (j - i) \times \min(h_i, h_j)$$



We will now calculate the area for each pair of bars:

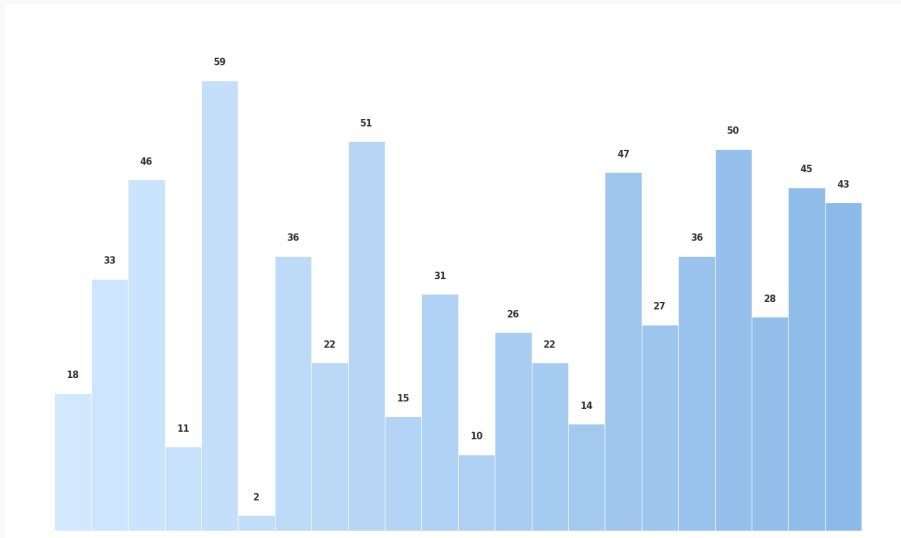
- For bars at positions 0 and 8:  $((8-0) \times \min(39, 40) = 8 \times 39 = 312)$
- For bars at positions 0 and 16:  $((16-0) \times \min(39, 32) = 16 \times 32 = 512)$
- For bars at positions 8 and 16:  $((16-8) \times \min(40, 32) = 8 \times 32 = 256)$

After checking all pairs, the maximum area is found to be 512.

Thus, the maximum amount of water that can be held is  $\boxed{512}$ .

## 4.2 Qwen-2.5-VL-72B Output Example 2 (Failure Case)

### Input Prompt



<image>Given a row of vertical bars where consecutive bars are adjacent with no gaps between them.\nPick any two bars and form the sides of a water container, with the x-axis as the base.\nHow much water can the biggest possible container hold ?\n\nPlease analyze the image carefully and provide your answer.\n\nOutput Format:\nProvide your final answer as an integer enclosed in `\\boxed{}`\n

### Model Response

To determine the maximum amount of water that can be held by a container formed by two vertical bars, we need to consider the height of the shorter bar and the distance between the two bars. The volume of water that can be held is given by the formula:

`\\[ \\text{Volume} = \\text{height of the shorter bar} \\times \\text{distance between the bars} \\]`

We will use a two-pointer approach to find the maximum volume. Start with one pointer at the beginning (left) and another at the end (right) of the array of bar heights. Calculate the volume for these two bars, then move the pointer pointing to the shorter bar inward and repeat

the process until the pointers meet.

Here are the steps in detail:

1. Initialize two pointers: `left` at the start (index 0) and `right` at the end (index 19).
2. Calculate the volume using the current `left` and `right` pointers.
3. Move the pointer pointing to the shorter bar inward.
4. Repeat steps 2 and 3 until `left` meets `right`.
5. Track the maximum volume encountered during the process.

Let's go through the steps with the given bar heights:

```
```python
heights = [18, 33, 46, 59, 11, 2, 36, 22, 51, 15, 31, 10, 26, 22, 14, 47,
           27, 50, 45, 43]
max_volume = 0
left = 0
right = len(heights) - 1

while left < right:
    h = min(heights[left], heights[right])
    width = right - left
    volume = h * width
    max_volume = max(max_volume, volume)

    if heights[left] < heights[right]:
        left += 1
    else:
        right -= 1

print(max_volume)
```
```

Running this code, we get the maximum volume as 918. Therefore, the answer is:

\\[

```
\\boxed{918}
```

```
\\]
```

```
", "response_context": "Assistant:
```

To determine the maximum amount of water that can be held by a container formed by two vertical bars, we need to consider the height of the shorter bar and the distance between the two bars. The volume of water that can be held is given by the formula:

```
\\[ \\text{Volume} = \\text{height of the shorter bar} \\times \\text{distance between the bars} \\]
```

We will use a two-pointer approach to find the maximum volume. Start with one pointer at the beginning (left) and another at the end (right) of the array of bar heights. Calculate the volume for these two bars, then move the pointer pointing to the shorter bar inward and repeat the process until the pointers meet.

Here are the steps in detail:

1. Initialize two pointers: `left` at the start (index 0) and `right` at the end (index 19).
2. Calculate the volume using the current `left` and `right` pointers.
3. Move the pointer pointing to the shorter bar inward.
4. Repeat steps 2 and 3 until `left` meets `right`.
5. Track the maximum volume encountered during the process.

Let's go through the steps with the given bar heights:

```
```python
heights = [18, 33, 46, 59, 11, 2, 36, 22, 51, 15, 31, 10, 26, 22, 14, 47,
           27, 50, 45, 43]
max_volume = 0
left = 0
right = len(heights) - 1

while left < right:
    h = min(heights[left], heights[right])
    width = right - left
```

```
volume = h * width
max_volume = max(max_volume, volume)

if heights[left] < heights[right]:
    left += 1
else:
    right -= 1

print(max_volume)
...
```

Running this code, we get the maximum volume as 918. Therefore, the answer is:

```
\\[
\\boxed{918}
```