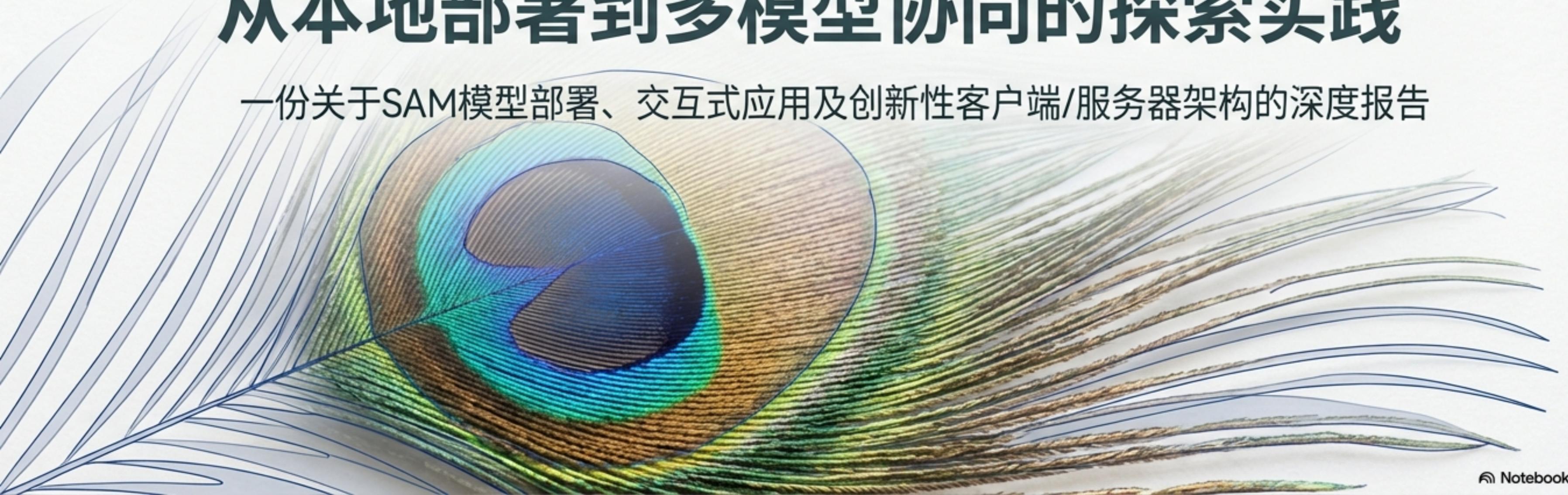


# 万物皆可分割：Segment Anything 模型 从本地部署到多模型协同的探索实践

一份关于SAM模型部署、交互式应用及创新性客户端/服务器架构的深度报告



# 重新定义图像分割：为何 Segment Anything (SAM) 如此重要？

图像分割是计算机视觉的关键任务。Meta AI提出的SAM模型，旨在构建一个通用的分割基础模型，其核心优势体现在三个方面：



**1. 任务通用性 (Task-Agnostic)**：统一各类分割任务范式，无需为特定任务重新训练。



**2. 交互灵活性 (Interactive Flexibility)**：支持多种提示输入（如点、框）来引导分割，实现人机协同。



**3. 零样本泛化 (Zero-Shot Generalization)**：能够直接迁移至未见过的目标与场景，展现强大的泛化能力。



# 从理论到现实：我们的本地部署蓝图

核心目标：在个人计算机硬件限制下，实现模型的稳定运行与高效推理。

## 硬件配置



- CPU: Intel(R) Core(TM) i7-10870H @ 2.20GHz
- 内存 (RAM) : 16GB

## 关键软件栈

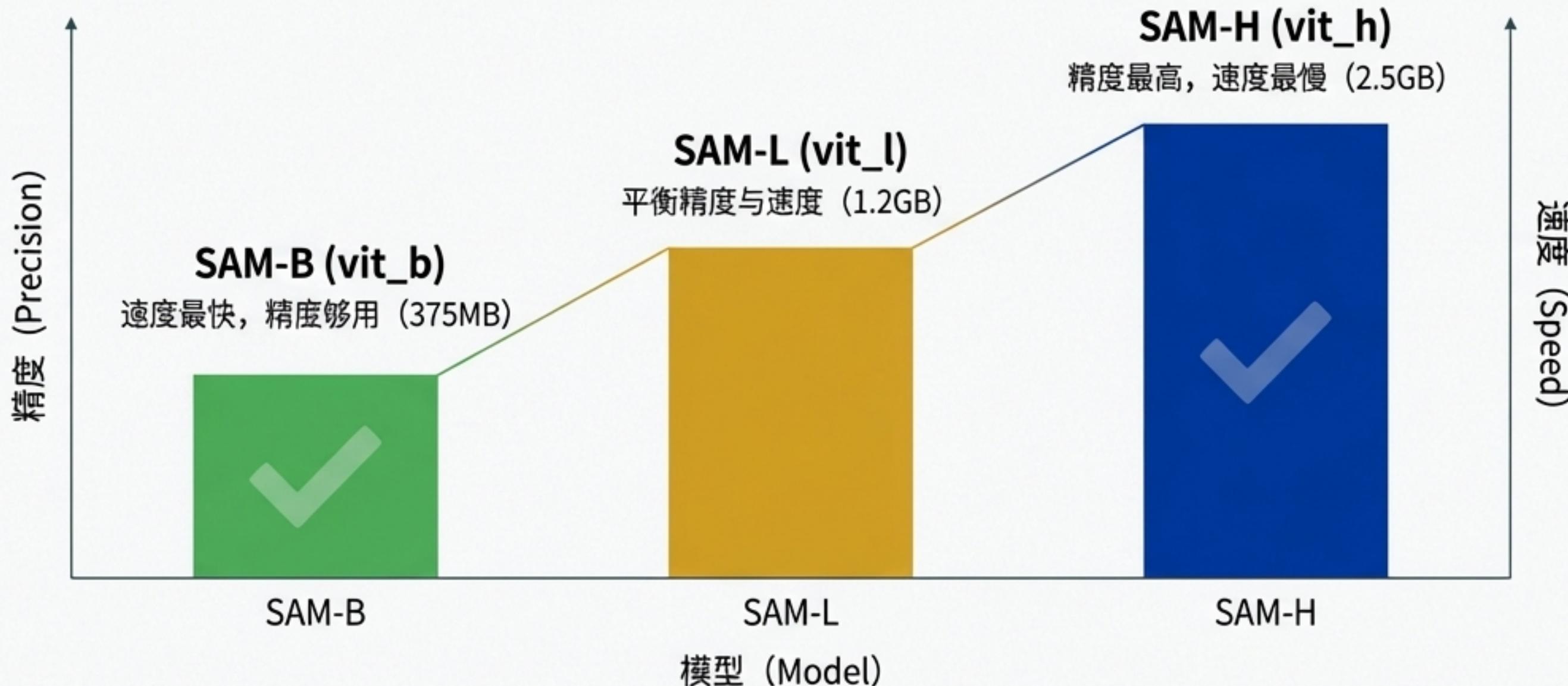


- 操作系统 (OS) : Windows 11 专业版
- 开发环境 (Environment) : Python 3.10, PyTorch 2.9
- 核心依赖库 (Core Libraries) :  
`segment\_anything` 1.0, OpenCV 4.12.0, Gradio 4.13.0



PyTorch

# 选择合适的工具：在精度与速度之间权衡



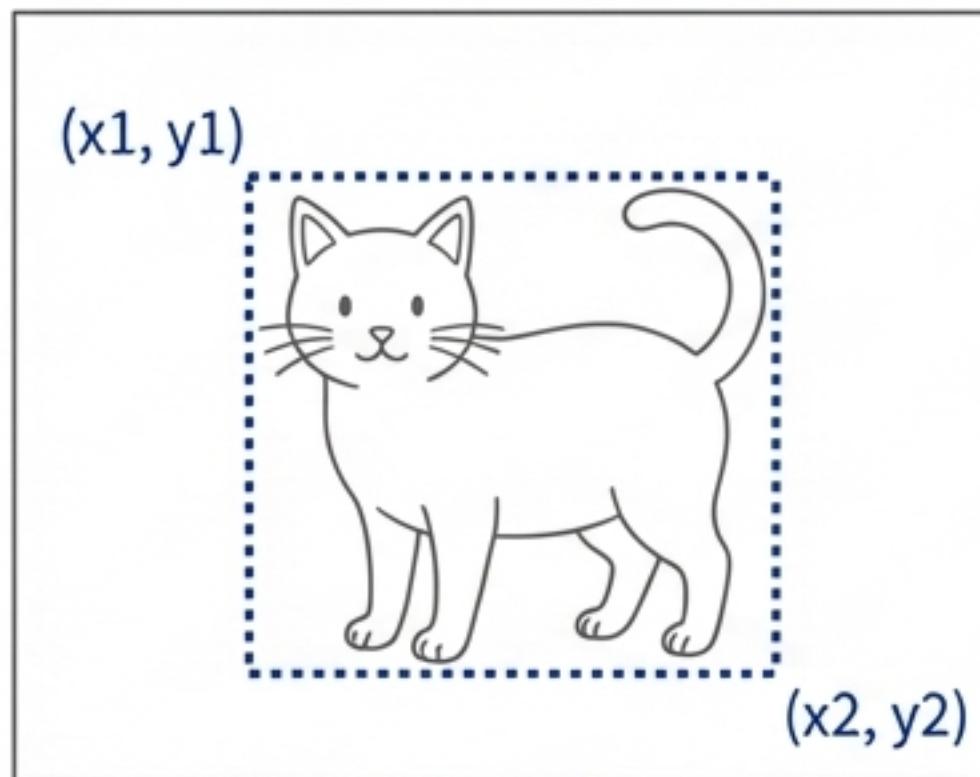
## 本项目的选择 (Our Project's Choice)

我们同时部署了 SAM-H 和 SAM-B，以便在后续实验中对比高精度和高速度模型的效果。

# 实验一：基础交互 - 使用边界框（Box Prompt）精准引导

## 概念（Concept）

`box prompt`通过用户输入一个矩形边界框的坐标  $(x_1, y_1, x_2, y_2)$  来锁定目标的大致范围，引导SAM在此区域内完成精准分割。



## 关键代码（`sam.py`）

```
1 import numpy as np  
2  
3 # ... other code ...  
4 masks, scores, logits = predictor.predict(  
5     point_coords=None,  
6     point_labels=None,  
7     box=np.array([425, 600, 700, 875])(None, :), # Bounding box  
8     multimask_output=False  
9 )  
10 # ... other code ...
```



**代码注解：**第29行代码是核心，定义了包围前景物体的矩形框坐标。

# Box Prompt 效果对比：高低精度模型表现

原图 (Original Image)



低精度模型 (SAM-B) 输出



高精度模型 (SAM-H) 输出



Source Han Sans CN Regular  
'sam\_vit\_b\_01ec64.pth' - 速度快，精度够用

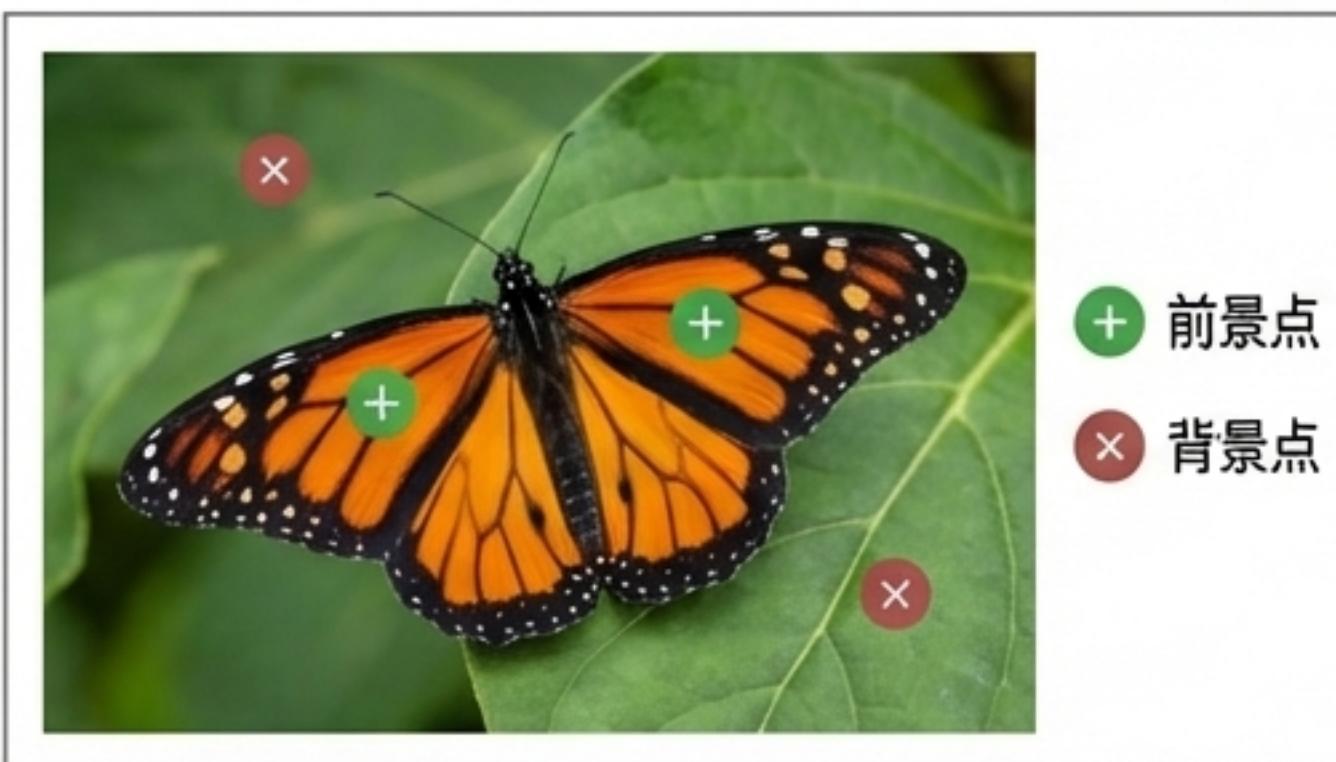
Source Han Sans CN Regular  
'sam\_vit\_h\_4b8939.pth' - 细节更完整，边缘更清晰

# 进一步精调：使用点提示（Point Prompt）进行细化分割

## 概念（Concept）

`point prompt` 允许用户通过标注多个点来更精细地定义分割区域。

- 前景点（Foreground Points）：指示模型需要分割的目标。
- 背景点（Background Points）：指示模型需要排除的区域。
- 本实验配置（Our Experiment's Configuration）：使用两个前景点和两个背景点进行引导。



## 关键代码（`sam.py`）

```
1 import numpy as np  
2  
3 # ... other code ...  
4 input_point = np.array([500, 375], [700, 500], [600, 200], [250, 750])  
5 input_label = np.array([1, 1, 0, 0])  
6  
7 masks, _, _ = predictor.predict(  
8     point_coords=input_point,  
9     point_labels=input_label,  
10    multimask_output=False,  
11 )  
12 # ... other code ...
```

代码注解：通过 `input\_point` 传入坐标，`input\_label`（1为前景，0为背景）区分点类型。

# Point Prompt 效果对比：精细引导下的分割结果

原图 (Original Image)



低精度模型 (SAM-B) 输出



高精度模型 (SAM-H) 输出



## 观察结论 (Observation)

点提示提供了更强的控制力，即使是复杂的边界，高精度模型也能实现优秀的分割效果。

# 实验二：从命令行到用户界面 - 构建交互式分割应用

**核心理念 (Core Idea)**：“用户引导 + 模型响应”是SAM交互式分割的精髓。为提升用户体验，我们将静态的‘box prompt’功能迁移至网页端，实现真正的可视化、实时交互。

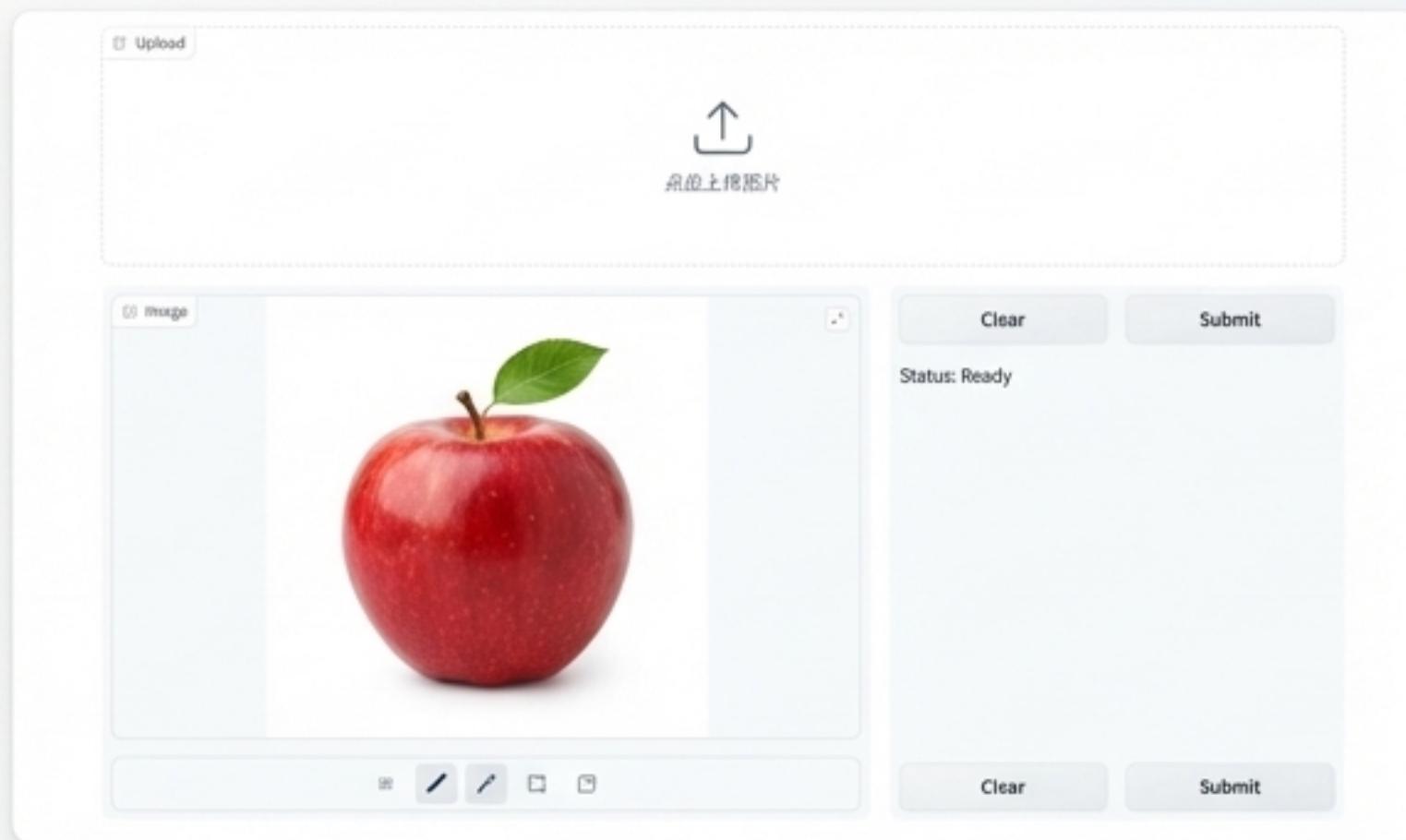
## 技术实现 (Technology Implementation)

- **框架 (Framework)**：使用轻量级Web框架 **Gradio 4.13.0** 快速搭建可视化界面。  Gradio
- **脚本 (Script)**：samwebplus.py
- **交互方式 (Interaction Method)**：用户直接在网页上用鼠标左键拖拽，标记矩形框的左上角和右下角。



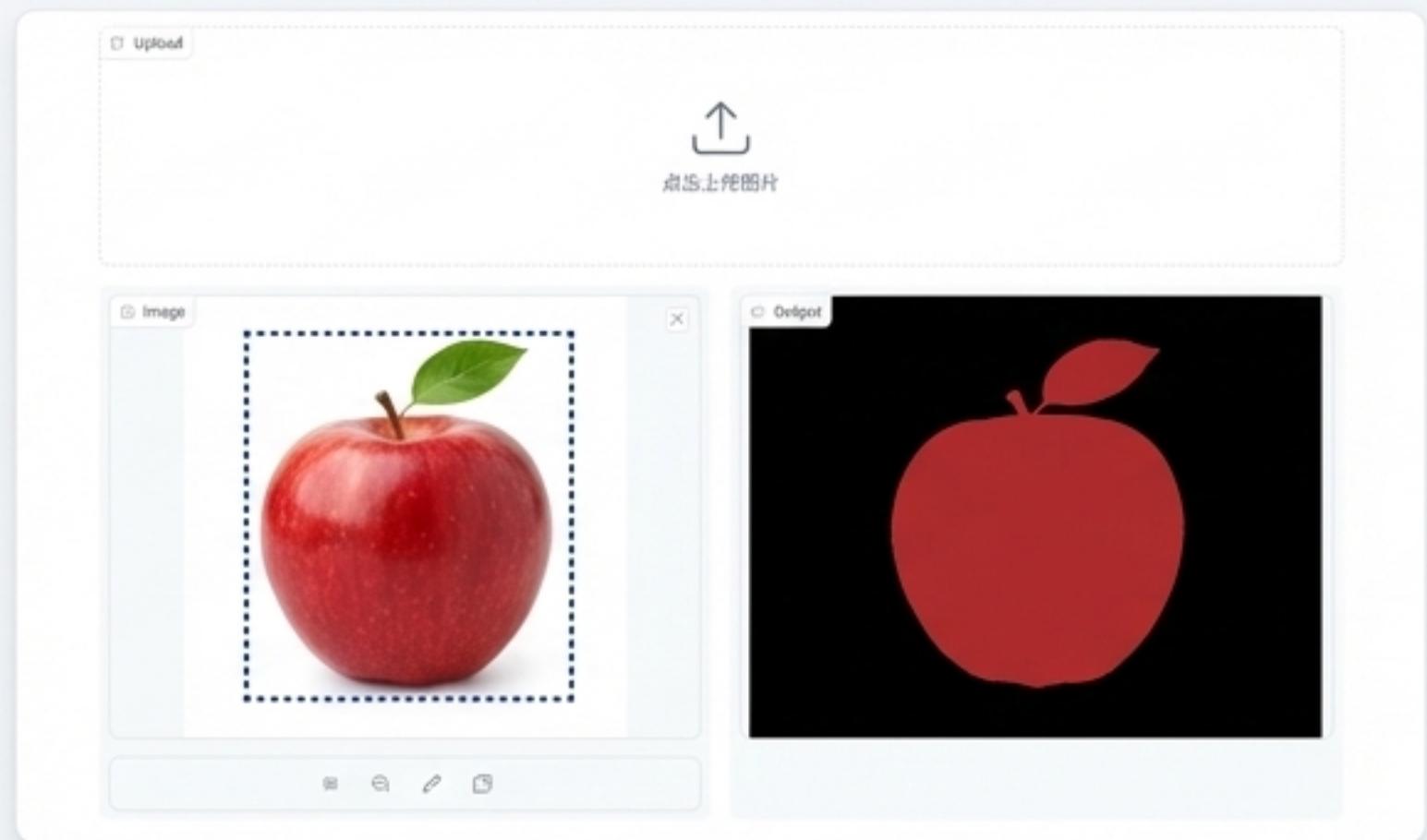
# 实时交互体验：网页版 Box Prompt 操作演示

左侧：操作界面



说明（Caption）：用户上传图片后，可在界面上直接用鼠标绘制选框。

右侧：分割结果



说明（Caption）：模型根据用户绘制的选框，实时生成并展示分割后的前景图像。

# 实验三：范式突破 - 构建 Qwen3 与 SAM 的跨设备协同架构

## 实验背景 (Experiment Context)

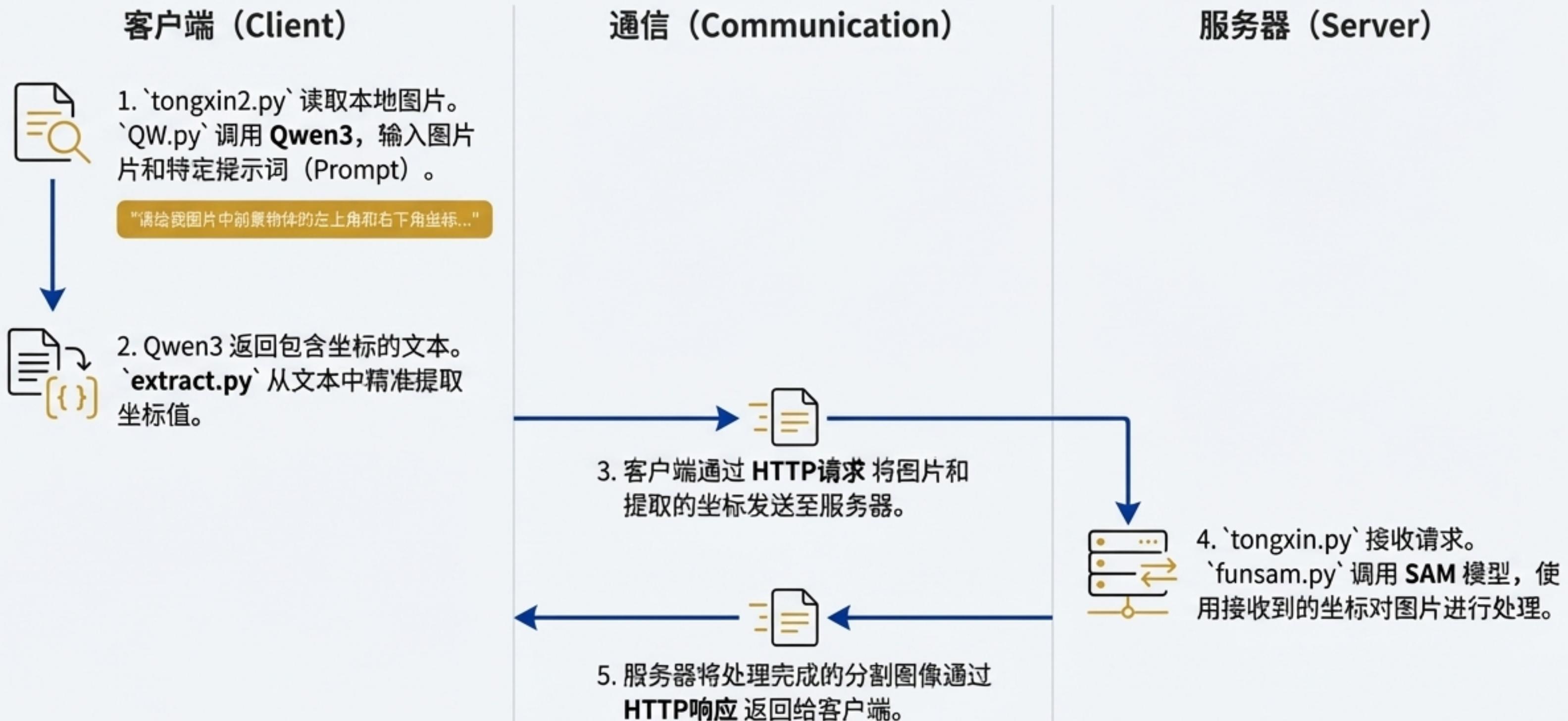
能否让一个大语言模型 (LLM) 自动为SAM提供分割提示？我们设计了一个实验，让部署在电脑A上的Qwen3模型为部署在电脑B上的SAM模型提供坐标。

## 核心挑战 (Core Challenge)

实现两个独立部署的大模型之间的无缝通信与任务协作。



# 协同工作流详解：从坐标生成到图像分割



# 代码中的协同：客户端与服务器的关键实现

## 客户端 (Qwen3)

### \*\*调用模型 (`QW.py`) & 提取坐标 (`extract.py`)

```
# QW.py - Simplified
response = qwen.Generation.call(
    model="qwen-turbo",
    messages=[{'role': 'user', 'content': prompt}],
    result_format='message'
)
return response.output.choices[0].message.content
```

→ 展示如何向Qwen3提问并从其自然语言回答中解析出结构化坐标数据。

### \*\*发起通信 (`tongxin2.py`)

```
# tongxin2.py - Simplified
url = "http://server_ip:port/upload"
files = {'file': open(image_path, 'rb')}
data = {'coords': json.dumps(coordinates)}
response = requests.post(url, files=files, data=data)
```

→ 重点展示构建HTTP POST请求，将图片和坐标数据打包发送的部分。

## 服务器 (SAM)

### \*\*接收通信 (`tongxin.py`)

```
# tongxin.py - Simplified (using Flask)
@app.route('/upload', methods=['POST'])
def upload_file():
    file = request.files['file']
    coords = json.loads(request.form['coords'])
    # ... process with funsam.py ...
    return send_file(result_path)
```

展示如何启动  
HTTP服务器，解析  
传入的请求数据。



### \*\*执行SAM (`funsam.py`)

```
# funsam.py - Simplified
def process_with_sam(image, coords):
    predictor.set_image(image)
    input_point = np.array(coords['points'])
    input_label = np.array(coords['labels'])
    masks, _, _ = predictor.predict(...)
    return masks
```

→ 展示如何调用SAM模型库，并将客户端传来的坐标作为`point prompt`输入。

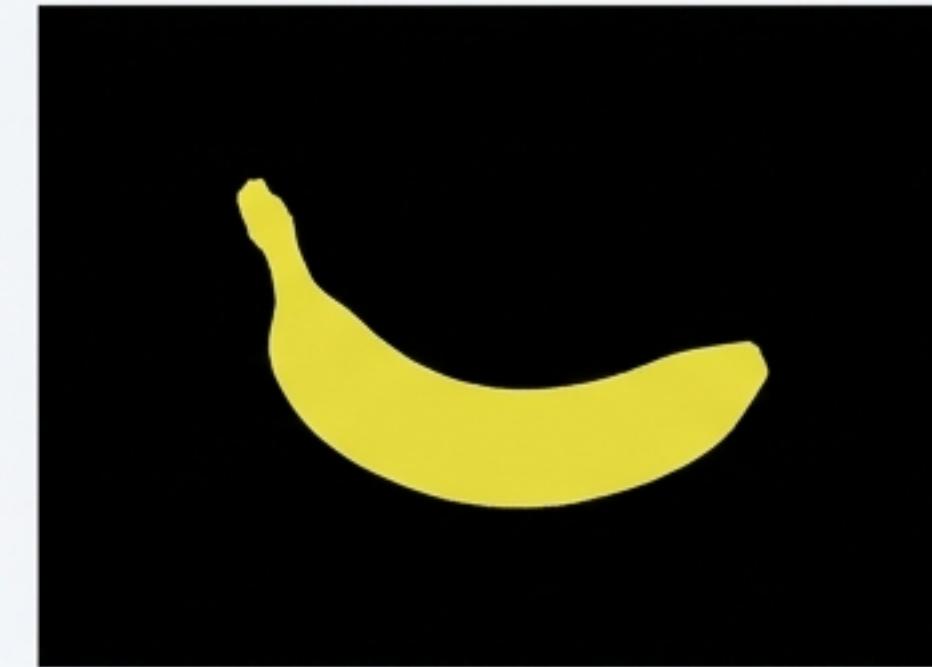
# 最终成果：由 Qwen3 驱动的全自动图像分割

左侧：原图 (Left: Original Image)



说明 (Caption)：客户端输入的原始图像。

右侧：SAM 分割图 (Right: SAM Segmented Image)



说明 (Caption)：服务器SAM根据Qwen3自动生成的坐标完成的前景分割图。

## 客户端

```
[INFO] Loading image: banana.jpg...
[INFO] Querying Qwen3 for coordinates...
[SUCCESS] Coordinates received: [[210, 350], [650, 480]]
[INFO] Sending data to server...
[SUCCESS] Segmented image received and saved.
```

## 服务器

```
[INFO] Server listening on 0.0.0.0:5000
[INFO] Received POST request from client.
[INFO] Processing image with SAM using coordinates...
[SUCCESS] Segmentation complete.
[INFO] Sending result back to client.
```

说明 (Caption)：程序运行时客户端与服务器的通信日志。

# 总结与展望：一次成功的架构验证与未来方向

## 成果与反思 (Achievements & Reflections)



**核心收获：**成功验证了跨设备、多模型 (LLM+Vision Model) 协同工作的技术路径和可行性。这是本次探索最有价值的成果。



**当前局限：**由于使用的Qwen3模型为最小体量版本，其生成的坐标精度有待提升，导致部分分割效果未达最优。

## 未来展望 (Future Outlook)

- 随着大语言模型能力的不断增强，未来有望利用更强大的模型为SAM提供更高质量、甚至更富语义的分割提示（例如，通过自然语言直接命令“分割出图片中的所有车辆”）。
- 本次实验为探索更智能、更自动化的视觉任务处理流程奠定了坚实基础。

