PR #27026 完整报告

sgl-project/sglang

[diffusion] Add realtime WebUI super resolution controls

合并时间：2026-06-02 20:29

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27026

执行摘要

- 一句话：为实时 WebUI 添加超分辨率与预览缩放控件
- 推荐动作：此 PR 实施质量较高，前端改动有配套样式调整和测试，后端逻辑增强有缓存和动态适配。建议关注点：后处理模块的缓存线程安全性、测试对 monkeypatch 的依赖（是否脆弱）。对于类似需求（如帧插值控制），可以参考此 PR 的控件和测试模式。

功能与动机

根据 PR 描述，目标是为实时 WebUI 增加用户可调节的超分辨率控制，包括启用 / 禁用、缩放比例选择，以及预览缩放和输出尺寸显示，从而提升超分辨率功能在实时场景中的可用性与可视化效果。同时添加单元测试验证实时原始帧上采样材质化路径的正确性，确保在不加载真实 SR 模型的情况下能正确工作。

实现拆解

实现分为以下 5 个步骤：

1. 前端控件与交互：在 index.html 中添加 id="superResolution" 的复选框和 id="upscalingScale" 的选择框，以及预览缩放滑块 id="previewScale" 和输出尺寸显示 id="outputSizeText"。在 app.js 中新增 setPreviewState、readUpscalingScale、readSuperResolutionParams、parseSizeValue、updateOutputSizeText、updateOutputSizeFromHeader、updateSuperResolutionControls、setPreviewScale 等函数，实现控件状态读取、WebSocket 发送参数拼接、预览缩放 CSS 变量更新、从 frame header 解析输出尺寸并更新 UI。
2. 超分辨率后端适配：在 realesrgan_upscaler.py 中新增 _default_model_path_for_scale 函数，根据缩放比例（2/4/8）返回对应的权重文件名（RealESRGAN_x2.pth 等），而非默认的固定 x4 权重。同时新增 _RESOLVED_MODEL_PATH_CACHE 字典，缓存已解析的模型路径，避免重复下载。在 _build_net_from_state_dict 中改进 RRDBNet 的 scale 检测，根据 conv_first 输入通道数（3/12/48）动态确定缩放比例（4/2/1），支持更多预训练变体。
3. 布局与样式重构：styles.css 中网格布局从三栏改为两栏，新增 .workspace、.preview-frame、.preview-overlay、.preview-scale-control 等样式，实现预览帧容器、叠加层、缩放控制条的视觉设计。调整 #viewport 的 max-height 为基于 var(--preview-scale) 的动态值。

4. 测试覆盖：test_realtime_webui.py 增加对新增 HTML 元素、CSS 类、JS 常量和函数调用的断言，确保控件正确渲染且无退化。test_output_materialization.py 新增 test_raw_rgb_frame_batches_apply_realtime_upscaling，使用 monkeypatch 替换真实的 batch_upscale_frames，验证在启用 upscaling 时 build_raw_rgb_frame_batches 正确调用 upscaling 函数并生成预期字节流。
5. 依赖配置：pyproject.toml 中添加一行依赖（具体包未明确，可能为 WebSocket 相关），确保后端实时通信依赖正确声明。

关键文件：

- python/sglang/multimodal_gen/apps/realtime_webui/app.js（模块 实时 WebUI；类别 source；类型 core-logic；符号 setPreviewState, readUpscalingScale, readSuperResolutionParams, parseSizeValue）：核心前端文件，新增了超分辨率和预览缩放的所有交互逻辑，改动量最大（+139/-25）。
- python/sglang/multimodal_gen/runtime/postprocess/realesrgan_upscaler.py（模块 超分辨率；类别 source；类型 core-logic；符号 _default_model_path_for_scale, _RESOLVED_MODEL_PATH_CACHE, _build_net_from_state_dict）：后端超分辨率模块，新增按比例选择权重文件的功能，并优化了模型路径缓存和 scale 自动检测。
- python/sglang/multimodal_gen/test/unit/realtime/test_output_materialization.py（模块 输出材质化；类别 test；类型 test-coverage；符号 test_raw_rgb_frame_batches_apply_realtime_upscaling, fake_batch_upscale_frames, post_process_sample）：覆盖实时上采样材质化路径，验证在启用了 upscaling 时 build_raw_rgb_frame_batches 正确调用 upscaling 函数并产出预期字节。
- python/sglang/multimodal_gen/test/unit/realtime/test_realtime_webui.py（模块 WebUI 测试；类别 test；类型 test-coverage）：验证新增控件在 HTML 和 JS 中的存在性，确保 WebUI 渲染无退化。
- python/sglang/multimodal_gen/apps/realtime_webui/styles.css（模块 前端样式；类别 other；类型 core-logic）：布局重构和新增预览帧、叠加层、缩放控制等样式，支撑前端交互。

关键符号：setPreviewState, readUpscalingScale, readSuperResolutionParams, parseSizeValue, updateOutputSizeText, updateOutputSizeFromHeader, updateSuperResolutionControls, setPreviewScale, _default_model_path_for_scale, _build_net_from_state_dict, test_raw_rgb_frame_batches_apply_realtime_upscaling

关键源码片段

python/sglang/multimodal_gen/apps/realtime_webui/app.js

核心前端文件，新增了超分辨率和预览缩放的所有交互逻辑，改动量最大（+139/-25）。

```
// 新增默认常量
const DEFAULT_UPSCALING_SCALE = 2;
const DEFAULT_PREVIEW_SCALE = 120;

let renderedPreviewFrames = 0;
let previewScaleFrame = 0;
```

```
// 获取新布局元素
const stage = document.querySelector(".stage");
const previewFrame = document.querySelector(".preview-frame");

// 设置预览状态（idle / waiting / active）
function setPreviewState(state) {
  if (!stage) return;
  stage.dataset.previewState = state;
  canvas.setAttribute("aria-busy", state === "waiting" ? "true" : "false");
}

// 重写的 idle 画面（固定 1280x720，更纯粹的抗压占位设计）
function drawIdle() {
  const w = 1280, h = 720;
  if (canvas.width !== w || canvas.height !== h) {
    canvas.width = w;
    canvas.height = h;
  }
  setPreviewState("idle");
  renderedPreviewFrames = 0;
  ctx.fillStyle = "#11140f";
  ctx.fillRect(0, 0, w, h);

  const surface = ctx.createLinearGradient(0, 0, 0, h);
  surface.addColorStop(0, "rgba(238,241,236,0.045)");
  surface.addColorStop(0.5, "rgba(238,241,236,0.012)");
  surface.addColorStop(1, "rgba(0,0,0,0.16)");
  ctx.fillStyle = surface;
  ctx.fillRect(0, 0, w, h);

  ctx.strokeStyle = "rgba(238,241,236,0.11)";
  ctx.lineWidth = 1;
  ctx.strokeRect(0.5, 0.5, w - 1, h - 1);
  ctx.strokeStyle = "rgba(238,241,236,0.08)";
  ctx.beginPath();
  if (ctx.roundRect) {
    ctx.roundRect(w * 0.38, h * 0.42, w * 0.24, h * 0.16, 18);
  } else {
    ctx.rect(w * 0.38, h * 0.42, w * 0.24, h * 0.16);
  }
  ctx.stroke();

  ctx.fillStyle = "rgba(238,241,236,0.22)";
  for (let i = -1; i <= 1; i++) {
    ctx.beginPath();
    ctx.arc(w * 0.5 + i * 22, h * 0.5, 4.5, 0, Math.PI * 2);
    ctx.fill();
  }
}
```

python/sglang/multimodal_gen/test/unit/realtime/test_output_materialization.py

覆盖实时上采样材质化路径，验证在启用了 upscaling 时 build_raw_rgb_frame_batches 正确调用 upscaling 函数并产出预期字节。

```
def test_raw_rgb_frame_batches_apply_realtime_upscaling(monkeypatch):
    calls = []

    # 模拟上采样函数：记录调用参数，并返回缩放后的帧
    def fake_batch_upscale_frames(frames, *, model_path, scale):
        calls.append((model_path, scale, [frame.shape for frame in frames]))
        return [
            np.repeat(np.repeat(frame, scale, axis=0), scale, axis=1)
            for frame in frames
        ]

    from sglang.multimodal_gen.runtime import postprocess
    monkeypatch.setattr(postprocess, "batch_upscale_frames", fake_batch_upscale_frames)

    # 构造启用了 upscaling 的请求
    req = type(
        "Req",
        (),
        {
            "data_type": DataType.VIDEO,
            "fps": 24,
            "output_compression": None,
            "enable_frame_interpolation": False,
            "enable_upscaling": True,
            "upscaling_model_path": "mock-sr",
            "upscaling_scale": 2,
            "request_id": "req",
            "block_idx": 0,
        },
    )()
    output_batch = OutputBatch(audio_sample_rate=None)

    # post_process_sample 中不应再启用 upscaling
    def post_process_sample(_sample, *_args, **kwargs):
        assert kwargs["enable_upscaling"] is False
        return [np.array([[[1, 2, 3]]], dtype=np.uint8)]

    frame_batches, metadata = build_raw_rgb_frame_batches(
        torch.zeros(1, 3, 1, 1, 1),
        req,
        output_batch,
        post_process_sample,
    )
```

```
    # 验证调用参数
    assert calls == [("mock-sr", 2, [(1, 1, 3)])]
    # 验证输出元数据（宽度 / 高度经过了 2x 缩放）
    assert metadata == {
        "format": "rgb24",
        "width": 2,
        "height": 2,
        "channels": 3,
        "bytes_per_frame": 12,
    }
    assert len(frame_batches) == 1
    assert frame_batches[0][0] == bytes([1, 2, 3] * 4)
```

评论区精华

此 PR 没有收到实质性的审查评论。仅有一条 gemini-code-assist 机器人的每日配额提醒和作者自己的 /tag-and-rerun-ci 指令，不涉及技术讨论。

- 暂无高价值评论线程

风险与影响

- 风险：前端风险：布局从三栏变为两栏，可能影响现有用户的界面习惯；预览缩放使用了 CSS 变量 --preview-scale，需要确保与 canvas 渲染兼容。后端风险：_RESOLVED_MODEL_PATH_CACHE 是模块级全局字典，在多线程或多请求并发下可能存在竞态条件（但当前模型加载是单线程？需要确认）。动态 scale 检测如果遇到未预料的输入通道数会抛出 ValueError，可能影响模型加载。测试风险：测试未在本地运行，依赖 CI，可能遗漏与本地环境的兼容性问题。
- 影响：影响范围限定在 SGLang Diffusion 实时 WebUI 和 Real-ESRGAN 后处理模块。用户若使用实时 WebUI 将看到新的控件和布局变更，但功能是向后兼容的（不启用 SR 时无行为变化）。后端增加缓存可减少重复模型解析开销，但缓存粒度可能不够细。新测试确保材质化路径正确，提升回归安全性。
- 风险标记：前端布局变更影响现有交互，后处理缓存线程安全，模型 scale 检测兼容性，测试依赖 CI 未本地执行

关联脉络

- PR #27041 [diffusion] Optimize Cosmos3 lossless hot paths: 同属 diffusion 实时模块的性能优化 PR，与本 PR 功能无直接关联但共享模块基础设施。
- PR #27023 [diffusion] Optimize LingBot realtime transformer path: 同为 diffusion 实时生成的优化，本 PR 的超分辨率控制可与其形成互补。