PR #27279 完整报告

sgl-project/sglang

[Multimodal] Add Ideogram 4 FP8 generation support

合并时间：2026-06-05 23:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27279

执行摘要

- 一句话：为 multimodal_gen 添加 Ideogram 4 FP8 文生图支持
- 推荐动作：值得精读。该 PR 展示了在 SGLang 框架中系统性地添加新扩散模型的完整流程，包括配置、模型、pipeline 和测试，尤其是通过继承基类减少重复代码和共享工具函数的设计思路值得学习。

功能与动机

根据 PR body，该 PR 旨在为 multimodal_gen 添加对 Ideogram 4 FP8 文生图模型的原生支持。讨论中 Fatemanx 说明当前有两个开源变体（NF4 和 FP8），本 PR 集中支持 FP8 版本，以利用其权重压缩且不需要 FP8 张量核心硬件的特性。

实现拆解

- 配置层：新增 configs/pipeline_configs/ideogram.py 定义 Ideogram4PipelineConfig 和 latent 统计信息；configs/models/dits/ideogram.py 定义 Ideogram4DiTConfig；configs/models/encoders/ideogram.py 定义 Ideogram4TextEncoderConfig；configs/sample/ideogram.py 定义采样预设和 Ideogram4SamplingParams。
- 文本编码器：在 runtime/models/encoders/ideogram.py 实现 IdeogramQwen3VLTextEncoder，复用现有 Qwen3VLTextModel，并通过 swap_linears_to_weight_only_fp8 将线性层替换为存储型 FP8；encode_ideogram_features 方法提取指定层的 hidden states 拼接为 Ideogram 特征。
- DiT 模型：在 runtime/models/dits/ideogram.py 构建 Ideogram4TransformerBlock，使用共享的 USPAttention 和 Qwen3VLTextRotaryEmbedding；_linear 辅助函数调用 WeightOnlyFP8Linear，实现权重以 e4m3 FP8 存储、运算时反量化到 bf16。
- Pipeline 阶段：新建 pipelines_core/stages/model_specific_stages/ideogram.py，包含 Ideogram4TextEncodingStage（继承 TextEncodingStage）和 Ideogram4DenoisingStage（继承 DenoisingStage）；LogitNormalSchedule 控制去噪噪声计划；Ideogram4TextEncodingFingerprint 用于请求去重。
- Pipeline 组装：runtime/pipelines/ideogram.py 中的 Ideogram4Pipeline.create_pipeline_stages 组装各阶段，在 ImageDiffusionPipeline 中注册 ideogram-ai/ideogram-4-fp8。

- 共享工具重构：从 cosmos3video.py 删除原内联的 qwen3_apply_rotary_pos_emb 和 Qwen3VLTextRotaryEmbedding，提取到 runtime/layers/rotary_embedding/mrope.py，同时供 Ideogram4 和 Cosmos3 使用。
- 测试配套：test/unit/test_ideogram4.py 含 749 行单元测试，覆盖配置加载、FP8 线性层、采样预设和 pipeline 行为；test/server/gpu_cases.py 注册 e2e GPU 用例，但性能与一致性检查暂未启用。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/ideogram.py（模块 流水线阶段；类别 source；类型 data-contract；符号 LogitNormalSchedule, call, Ideogram4TextEncodingFingerprint, get_schedule_for_resolution）：新增的 pipeline 阶段核心文件，包含噪声调度、文本编码指纹和去噪阶段实现，是整体功能的主干。
- python/sglang/multimodal_gen/runtime/models/dits/ideogram.py（模块 DiT 模型；类别 source；类型 data-contract；符号 Ideogram4RMSNorm, init, forward, _linear）：新增的 Ideogram4 DiT 模型实现，包括注意力、MLP、Transformer Block 和正弦位置编码，复用了共享的 USPAttention 和 rotary embedding。
- python/sglang/multimodal_gen/runtime/models/encoders/ideogram.py（模块 文本编码器；类别 source；类型 data-contract；符号 IdeogramQwen3VLTextEncoder, init, forward, encode_ideogram_features）：实现 IdeogramQwen3VLTextEncoder，封装 Qwen3VLTextModel 且支持 FP8 权重替换，是功能关键入口。
- python/sglang/multimodal_gen/runtime/layers/quantization/weight_only_fp8.py（模块 量化层；类别 source；类型 dependency-wiring；符号 dequantize_rowwise_fp8_weight, WeightOnlyFP8Linear, init, forward）：新增的存储型 FP8 线性层实现，仅权重用 e4m3 存储，计算时反量化到 bf16，支撑 Ideogram4 的 FP8 权重加载。
- python/sglang/multimodal_gen/runtime/models/dits/cosmos3video.py（模块 DiT 模型；类别 source；类型 data-contract；符号 qwen3_apply_rotary_pos_emb, Qwen3VLTextRotaryEmbedding, init, apply_interleaved_mrope）：重构文件，移除了内联的 Qwen3MRoPE 函数和类，改为从共享模块导入，体现代码复用和清理。

关键符号：LogitNormalSchedule.call, get_schedule_for_resolution, Ideogram4TextEncodingFingerprint, Ideogram4Scheduler.set_begin_index, IdeogramQwen3VLTextEncoder.forward, IdeogramQwen3VLTextEncoder.encode_ideogram_features, IdeogramQwen3VLTextEncoder.load_weights, Ideogram4Attention.forward, Ideogram4TransformerBlock.forward, WeightOnlyFP8Linear.forward, swap_linears_to_weight_only_fp8

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/ideogram.py

新增的 pipeline 阶段核心文件，包含噪声调度、文本编码指纹和去噪阶段实现，是整体功能的主干。

```
# 导入所需组件
import math
from dataclasses import dataclass

import torch

# ... 其他导入 ...

# frozen dataclass 定义不可变的 LogitNormal 噪声调度
@dataclass(frozen=True)
class LogitNormalSchedule:
    mean: float
    std: float = 1.0
    logsnr_min: float = -15.0
    logsnr_max: float = 18.0

    def __call__(self, t: torch.Tensor) -> torch.Tensor:
        # 输入 t 为 [0,1] 区间的均匀采样，将其映射到噪声时间步
        t = t.to(torch.float64)
        z = torch.special.ndtri(t) # 正态分位函数
        y = self.mean + self.std * z
        t_ = 1 - torch.special.expit(y) # 逆 sigmoid 后再映射
        t_min = 1.0 / (1 + math.exp(0.5 * self.logsnr_max))
        t_max = 1.0 / (1 + math.exp(0.5 * self.logsnr_min))
        return t_.clamp(t_min, t_max).to(torch.float32)


# 根据图像分辨率调整噪声调度均值（参考 512x512 基准）
def get_schedule_for_resolution(image_resolution, known_mean: float, std: float):
    num_pixels = image_resolution[0] * image_resolution[1]
    known_pixels = 512 * 512
    mean = known_mean + 0.5 * math.log(num_pixels / known_pixels)
    return LogitNormalSchedule(mean=mean, std=std)
```

python/sglang/multimodal_gen/runtime/models/dits/ideogram.py

新增的 Ideogram4 DiT 模型实现，包括注意力、MLP、Transformer Block 和正弦位置编码，复用了共享的 USPAttention 和 rotary embedding。

```
import math
from typing import Any

import torch
import torch.nn as nn
import torch.nn.functional as F

from sglang.multimodal_gen.configs.models.dits.ideogram import Ideogram4DiTConfig
```

```
from sglang.multimodal_gen.runtime.layers.attention import (
    USPAttention,
    build_varlen_mask_meta,
)
from sglang.multimodal_gen.runtime.layers.quantization.weight_only_fp8 import (
    WeightOnlyFP8Linear,
)
from sglang.multimodal_gen.runtime.layers.rotary_embedding import (
    Qwen3VLTextRotaryEmbedding,
    qwen3_apply_rotary_pos_emb,
)
from sglang.multimodal_gen.runtime.models.dits.base import BaseDiT

OUTPUT_IMAGE_INDICATOR = 2
LLM_TOKEN_INDICATOR = 3


class Ideogram4RMSNorm(nn.Module):
    def __init__(self, dim: int, eps: float = 1e-6) -> None:
        super().__init__()
        self.weight = nn.Parameter(torch.ones(dim))
        self.eps = eps

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        return F.rms_norm(x, self.weight.shape, self.weight, self.eps)


# 工厂函数：默认使用 WeightOnlyFP8Linear
# WeightOnlyFP8Linear 存储 FP8 权重，计算时反量化到 compute_dtype
def _linear(in_features: int, out_features: int, bias: bool = True):
    return WeightOnlyFP8Linear(in_features, out_features, bias=bias)


class Ideogram4Attention(nn.Module):
    def __init__(
        self,
        hidden_size: int,
        num_heads: int,
        eps: float,
        supported_attention_backends,
    ) -> None:
        super().__init__()
        self.hidden_size = hidden_size
        self.num_heads = num_heads
        self.head_dim = hidden_size // num_heads
        # 复用 weight_only_fp8 线性层
        self.qkv = _linear(hidden_size, hidden_size * 3, bias=False)
        self.norm_q = Ideogram4RMSNorm(self.head_dim, eps=eps)
        self.norm_k = Ideogram4RMSNorm(self.head_dim, eps=eps)
```

```
        # 使用 USPAttention 支持多种后端，此处非因果注意力
        self.attn = USPAttention(
            num_heads=num_heads,
            head_size=self.head_dim,
            dropout_rate=0,
            softmax_scale=None,
            causal=False,
            supported_attention_backends=supported_attention_backends,
        )
        self.o = _linear(hidden_size, hidden_size, bias=False)

    def forward(self, x, cos, sin, attn_mask, attn_mask_meta):
        batch_size, seq_len, _ = x.shape
        qkv = self.qkv(x).view(batch_size, seq_len, 3, self.num_heads, self.head_dim)
        q, k, v = qkv.unbind(dim=2)
        q = self.norm_q(q)
        k = self.norm_k(k)
        # 应用共享的 Qwen3-style RoPE
        q, k = qwen3_apply_rotary_pos_emb(q, k, cos, sin)
        out = self.attn(q, k, v, attn_mask=attn_mask, attn_mask_meta=attn_mask_meta)
        out = out.reshape(batch_size, seq_len, self.hidden_size)
        return self.o(out)


class Ideogram4TransformerBlock(nn.Module):
    # ... 此处省略完整实现，参考源码
```

评论区精华

- weight_only_fp8 文件位置：mickqian 指出 weight_only_fp8.py 应放在 layers/quantization/ 目录下，Fatemanx 随即移动并更新导入。
- Ideogram4DiT attention 后端选择：mickqian 询问为何硬编码某些后端，Fatemanx 解释因当前使用稠密 padding mask，故仅支持 SDPA，并将后端选择配置化。
- TextEncodingStage 继承：mickqian 建议 Ideogram4TextEncodingStage 继承自现有的 TextEncodingStage，Fatemanx 采纳并保留 tokenization 等差异逻辑。
- DenoisingStage 继承：mickqian 同样要求 Ideogram4DenoisingStage 继承 DenoisingStage，Fatemanx 实现，并使用 unconditional_transformer 组件名映射。
- 测试图像上传与检查：mickqian 要求上传 GT 图像到 ci-data 并启用 run_perf_check 和 run_consistency_check，Fatemanx 已创建 ci-data PR，表示后续开启。
  - weight_only_fp8 文件位置 (design): 文件移至 runtime/layers/quantization/weight_only_fp8.py，导入更新。
  - Ideogram4DiT attention 后端选择 (design): 将 supported_attention_backends 从硬编码改为从 config 传入，解耦。
  - TextEncodingStage 继承 (design): Ideogram4TextEncodingStage 现在继承 TextEncodingStage，保留 tokenization 差异。

- DenoisingStage 继承 (design): Ideogram4DenoisingStage 继承 DenoisingStage，使用 unconditional_transformer 名称。
- 测试图像上传与检查 (testing): GT 图像 PR 已创建，但性能和一致性检查暂未启用，计划后续开启。

风险与影响

- 风险：
  - 自定义量化路径：WeightOnlyFP8Linear 存储 FP8 权重但计算反量化到 bf16，可能导致数值精度损失或性能瓶颈，需与官方 Diffusers 实现对比验证。
  - 继承基类约束：Ideogram4DenoisingStage 继承 DenoisingStage，未来基类变更可能引入不兼容。
  - 测试未完全启用：e2e 测试的性能和一致性检查暂关闭，可能遗漏回归。
  - 共享工具影响：从 Cosmos3Video 提取的 rotary embedding 模块若出错会影响两个模型，需回归。
- 影响：
  - 用户：可直接使用 DiffGenerator.from_pretrained("ideogram-ai/ideogram-4-fp8") 生成图像，降低使用门槛。
  - 系统：增加约 2.5k 行代码，但主要通过复用现有架构减少冲击；存储型 FP8 减少模型加载内存。
  - 团队：后续维护需关注 FP8 量化路径的特殊性，以及与其他 DiT 模型的兼容性。
  - 风险标记：自定义量化路径，继承基类约束，测试未完全启用

关联脉络

- 暂无明显关联 PR