生成加速

Ⅰ. webUI 加速

0. 基准速度：stable diffusion webUI 框架生成耗时（条件逐行增加）：

	尺寸	采样器	步数	LoRA	ControlNet	Latent Couple	速度（约）	sdp-attention	&channelslast	Doggettx
基准	512x512	Euler	20	无	无	无	17.7it/s、1.1s			
尺寸	1024x768						5.6it/s、3.5s			
采样器		DPM++ SDE Karras					2.8it/s、7.1s			
步数			30				2.8it/s、10.7s	5.5it/s、5.4s	5.8it/s、5.2s	
LoRA				couple			2.8it/s、10.7s	5.5it/s、5.4s	5.8it/s、5.2s	
ControlNet 1					Seg		2.0it/s、15.0s	3.9it/s、7.7s	4.3it/s、7.0s	
ControlNet 2					Pose		1.6it/s、18.7s	3.3it/s、9.1s	3.4it/s、8.8s	1.6it/s、18.7s
Latent Couple						rectangle	0.8it/s、37.4s	1.7it/s、17.2s	1.7it/s、17.2s	

webUI 性能 benchmark 网站：https://vladmandic.github.io/sd-extension-system-info/pages/benchmark.html

1. webUI 加速：现有工具

```
("--xformers", action='store_true', help="enable xformers for cross attention layers")
("--force-enable-xformers", action='store_true', help="enable xformers for cross attention layers regardless of whether the checking code thinks you can run it; do not make bug
("--xformers-flash-attention", action='store_true', help="enable xformers with Flash Attention to improve reproducibility (supported for SD2.x or variant only)")
("--deepdanbooru", action='store_true', help="does not do anything")
("--opt-split-attention", action='store_true', help="prefer Doggettx's cross-attention layer optimization for automatic choice of optimization")
("--opt-sub-quad-attention", action='store_true', help="prefer memory efficient sub-quadratic cross-attention layer optimization for automatic choice of optimization")
("--sub-quad-q-chunk-size", type=int, help="query chunk size for the sub-quadratic cross-attention layer optimization to use", default=1024)
("--sub-quad-kv-chunk-size", type=int, help="kv chunk size for the sub-quadratic cross-attention layer optimization to use", default=None)
("--sub-quad-chunk-threshold", type=int, help="the percentage of VRAM threshold for the sub-quadratic cross-attention layer optimization to use chunking", default=None)
("--opt-split-attention-invokeai", action='store_true', help="prefer InvokeAI's cross-attention layer optimization for automatic choice of optimization")
("--opt-split-attention-v1", action='store_true', help="prefer older version of split attention optimization for automatic choice of optimization")
("--opt-sdp-attention", action='store_true', help="prefer scaled dot product cross-attention layer optimization for automatic choice of optimization; requires PyTorch 2.*")
("--opt-sdp-no-mem-attention", action='store_true', help="prefer scaled dot product cross-attention layer optimization without memory efficient attention for automatic choice of
("--disable-opt-split-attention", action='store_true', help="prefer no cross-attention layer optimization for automatic choice of optimization")
```

webUI 已经集成了多种类型的加速库

目前公认加速效果最好的是 torch2.x+cu118+sdp，比正常推理方案提速一倍

reference：https://www.reddit.com/r/StableDiffusion/comments/y71q5k/comment/jcm67lu/?utm_source=share&utm_medium=web2x&context=3

2. 加速原理：

现有的加速工具都是对 transformer 部分进行加速（cnn 部分已经做的很好了，[4k,4k] 输入的 UNet 在 4090 上的耗时也不会超过 0.5s），而 transformer 部分包含多个组件：如 Linear、scaled dot-product、LayerNorm 等等，加速一般从内存机制（如何使得各组件之间的 tensor 读写更加合理）、高效 FLOPS（融合某些相邻层权重）、量化等。

以 xformer 为例，做了如下优化：

Key Features

1. Many attention mechanisms, interchangeables
2. Optimized building blocks, beyond PyTorch primitives
    i. Memory-efficient exact attention - up to 10x faster
    ii. sparse attention
    iii. block-sparse attention
    iv. fused softmax
    v. fused linear layer
    vi. fused layer norm
    vii. fused dropout(activation(x+bias))
    viii. fused SwiGLU
3. Benchmarking and testing tools
    i. micro benchnmarks
    ii. transformer block benchmark
    iii. LRA, with SLURM support
4. Programatic and sweep friendly layer and model construction
    i. Compatible with hierarchical Transformers, like Swin or Metaformer
5. Hackable
    i. Not using monolithic CUDA kernels, composable building blocks
    ii. Using Triton for some optimized parts, explicit, pythonic and user-accessible
    iii. Native support for SquaredReLU (on top of ReLU, LeakyReLU, GeLU, ..), extensible activations

from：https://github.com/facebookresearch/xformers#key-features

II . diffusers 加速

（先按之前看到的加速方案开一些可能的脑洞，还未尝试）

1. text_encoder 参数量比较大、且一般无需更改。如果已经确定 prompt，可以预先计算文本的 embeddings tensor，并且存到本地；
2. webUI 的框架如果使用了两个以上的 controlNet condition，会涉及两个 condition 模型的交替 load 和 offload，如果显存足够，可以尝试长期置于 GPU（maybe 优化 1~2s）；
3. 可以尝试生成低分辨率的图，在之后接 Upscaler（这个从原理上讲，其实速度上不会有太多提升，不过可以减少显存占用，而 webUI 里的 sdp attention 优化就是使用更多显存来加速的）# SD-2.x 系；
4. Latent Couple 优化方案；