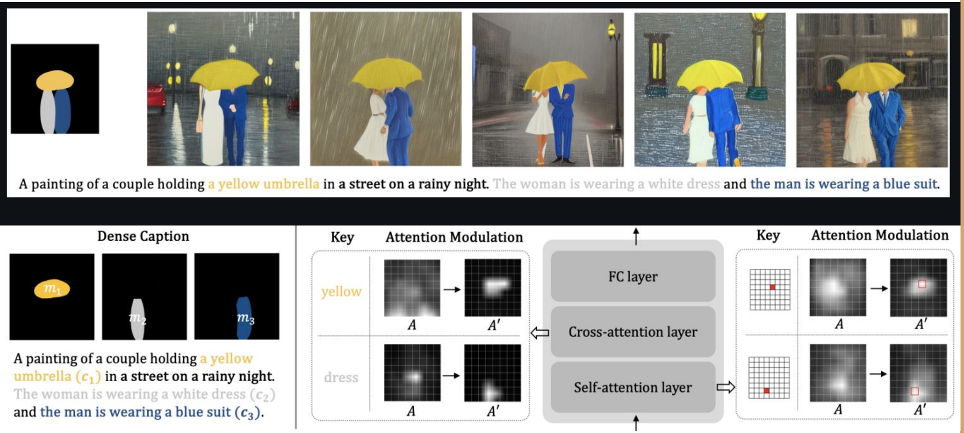
Transfer Dense Diffusion on SD 3.5

Yuehao Wu

December 23, 2024

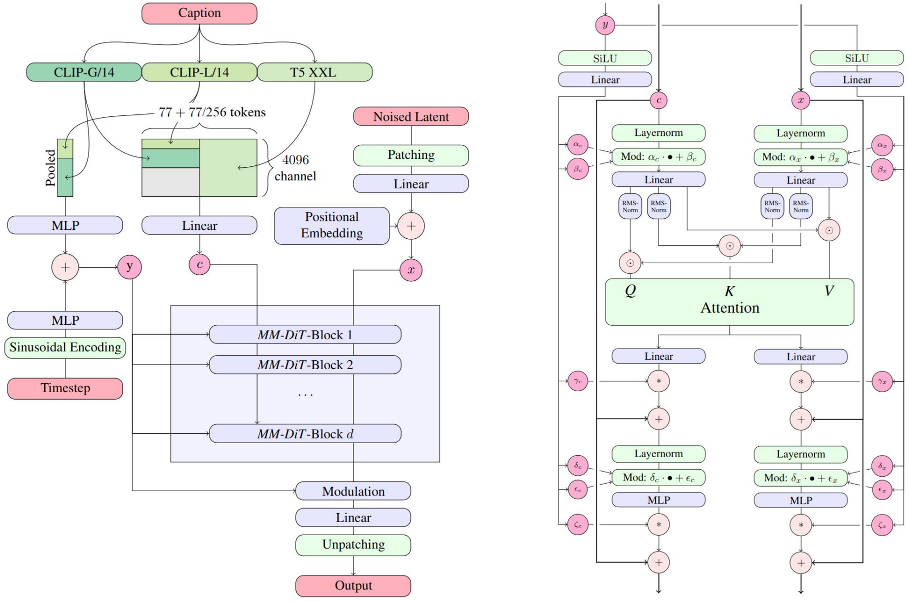
- Dense Diffusion
- Stable Diffusion - v 3.5
- Load and Transfer
- Result Analysis
- Future Work

Dense Diffusion



- Pretrained Model: Stable Diffusion v1.5
- Introduce dynamic cross-attention and self-attention
- Adaptive modulation mechanism

Stable Diffusion v3.5 (Compare to previous version)



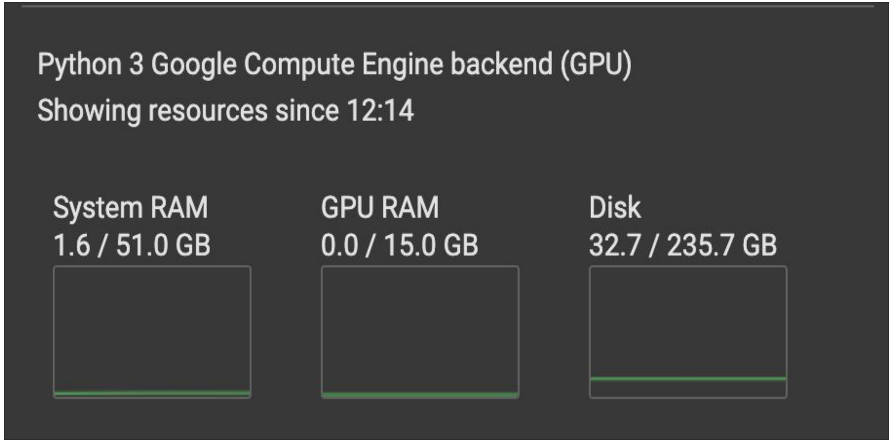
(a) Overview of all components.

(b) One *MM-DiT* block

- Using Transformer instead of U-Net
- Multimodal embedding and compatibility
- Better in details, resolution and semantic consistency

Code Implementation and Experimental Setup

```
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   39C    P8              10W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
```



- **Colab Pro**
- **T4 GPU - High RAM**
- **CUDA 12.2**

Load Stable Diffusion 3.5 model - 1

```
[ ]  import torch
     from diffusers import StableDiffusion3Pipeline
     pipe = StableDiffusion3Pipeline.from_pretrained(
         "stabilityai/stable-diffusion-3.5-large", torch_dtype=torch.float16
     ).to("cuda")
```

```
Loading checkpoint shards: 100%  2/2 [00:22<00:00, 11.91s/it]
---------------------------------------------------------------------------
OutOfMemoryError                          Traceback (most recent call last)
<ipython-input-3-52bb816a5509> in <cell line: 4>()
      4 pipe = StableDiffusion3Pipeline.from_pretrained(
      5     "stabilityai/stable-diffusion-3.5-large", torch_dtype=torch.bfloat16
----> 6 ).to("cuda")

                              8 frames
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in convert(t)
   1324                             memory_format=convert_to_format,
   1325                         )
-> 1326                     return t.to(
   1327                         device,
   1328                         dtype if t.is_floating_point() or t.is_complex() else
None,

OutOfMemoryError: CUDA out of memory. Tried to allocate 68.00 MiB. GPU 0 has a
total capacity of 14.75 GiB of which 9.06 MiB is free. Process 9046 has 14.74 GiB
memory in use. Of the allocated memory 14.51 GiB is allocated by PyTorch, and
128.16 MiB is reserved by PyTorch but unallocated. If reserved but unallocated
memory is large try setting PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True to
avoid fragmentation.  See documentation for Memory Management
(https://pytorch.org/docs/stable/notes/cuda.html#environment-variables)
```

Simply loading the entire Stable Diffusion 3.5 model at once will result in excessive peak resource usage

- **Solution: Loading in steps**

Load Stable Diffusion 3.5 model - 2

```
from transformers import T5EncoderModel
from diffusers import StableDiffusion3Pipeline
import torch

text_encoder_3_4bit = T5EncoderModel.from_pretrained(
    ckpt_4bit_id,
    subfolder="text_encoder_3",
)

pipeline = StableDiffusion3Pipeline.from_pretrained(
    ckpt_id,
    text_encoder_3=text_encoder_3_4bit,
    transformer=None,
    vae=None,
    torch_dtype=torch.float16,
)
pipeline.enable_model_cpu_offload()
```

```
import gc

del pipeline

gc.collect()
torch.cuda.empty_cache()
torch.cuda.reset_max_memory_allocated()
torch.cuda.reset_peak_memory_stats()
```

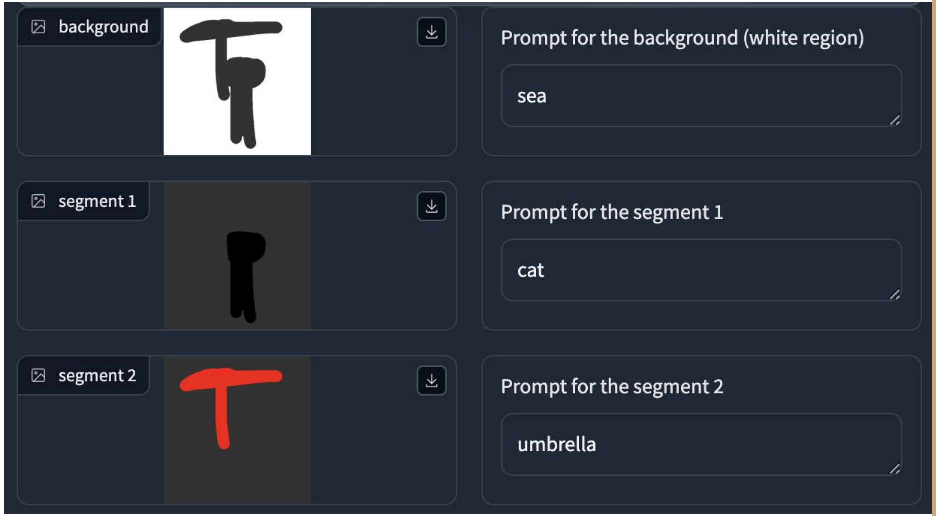
- Load the text encoder and Transformer modules separately; avoid occupying all memory at once
- Reduce the calculation from floating point to half-precision floating point, reducing video memory and RAM usage
- Offload inactive parts from the GPU to the CPU to reduce GPU memory usage

Load Stable Diffusion 3.5 model - 3

```
from diffusers import BitsAndBytesConfig, SD3Transformer2DModel
from diffusers import StableDiffusion3Pipeline
import torch

model_id = "stabilityai/stable-diffusion-3.5-medium"

nf4_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)
model_nf4 = SD3Transformer2DModel.from_pretrained(
    model_id,
    subfolder="transformer",
    quantization_config=nf4_config,
    torch_dtype=torch.bfloat16
)

pipeline = StableDiffusion3Pipeline.from_pretrained(
    model_id,
    transformer=model_nf4,
    torch_dtype=torch.bfloat16
)
pipeline.enable_model_cpu_offload()
```

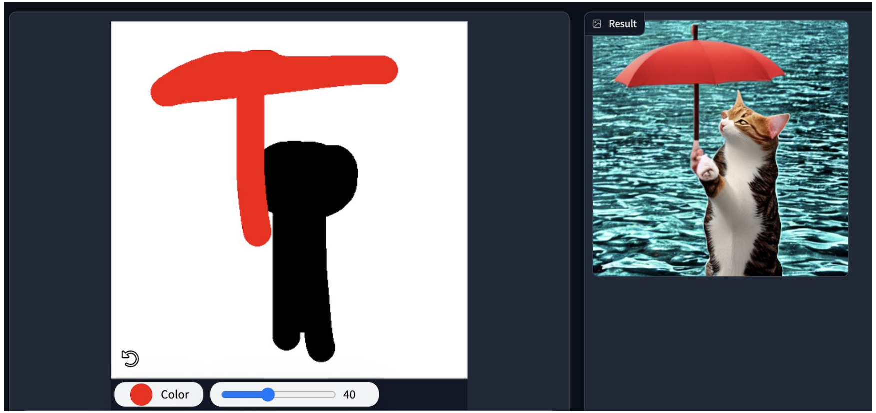
- **SD 3.5 Medium over the Large model for a shorter generation time and lower VRAM requirement**
- **Reduce the storage space required by the model and improve inference efficiency through quantification technology**
- **Offload inactive parts from the GPU to the CPU to reduce GPU memory usage**

Transfer - Improve layout controls-1



One of the core features of Dense Diffusion is segment-based generation through the introduction of an attention mechanism. This approach enables the model to independently generate different parts of an image based on defined image segments and textual cues.

Transfer - Improve layout controls-2



Current goal is to bring the segmented generation capabilities of Dense Diffusion to Stable Diffusion 3.5 to enhance its control over scene layout. This will help achieve more refined generation control while maintaining the efficiency and flexibility of Stable Diffusion 3.5.

Transfer - Improve layout controls-2

```
latent = latent * segment_mask + noise * (1 -
    segment_mask)
```

```
segment_attention_weights =
    compute_attention_weights(segment_size,
    target_complexity)
attn_weights = attn_weights * segment_attention_weights
```

- In the initial stage of diffusion, use mask restriction to each segment of the noise image to ensure that the features are distributed in the specified area

- Dynamically adjusted attention weight so that smaller segments receive higher attention values

Transfer - Dynamic attention modulation mechanism -1

```
for layer in cross_attention_layers:
    sim = compute_attention_scores(Q, K)
    sim = sim + layout_mask * pos_weight - (1 -
        layout_mask) * neg_weight
    attention_probs = softmax(sim)
    layer_output = attention_probs @ V
```

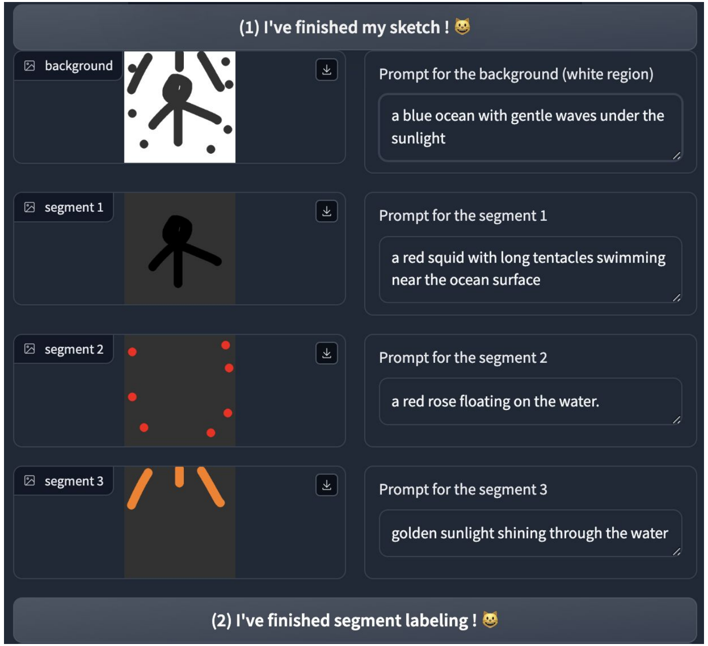
Current goal is to bring the segmented generation capabilities of Dense Diffusion to Stable Diffusion 3.5 to enhance its control over scene layout. This will help achieve more refined generation control while maintaining the efficiency and flexibility of Stable Diffusion 3.5.

Transfer - Dynamic attention modulation mechanism -2

```
for layer in self_attention_layers:
    sim = compute_attention_scores(Q, K)
    size_reg = compute_size_reg(layout_region)
    sim = sim + size_reg * attention_mod
    attention_probs = softmax(sim)
    layer_output = attention_probs @ V
```

Current goal is to bring the segmented generation capabilities of Dense Diffusion to Stable Diffusion 3.5 to enhance its control over scene layout. This will help achieve more refined generation control while maintaining the efficiency and flexibility of Stable Diffusion 3.5.

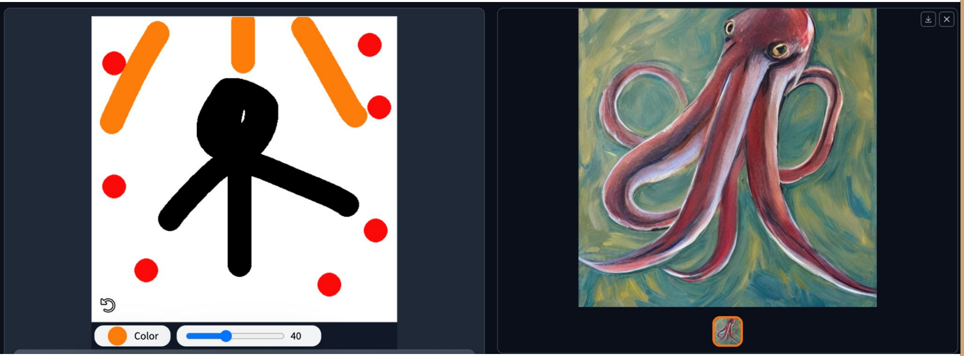
Result - Dense Diffusion - 1



Give more Complex Prompt for each part:

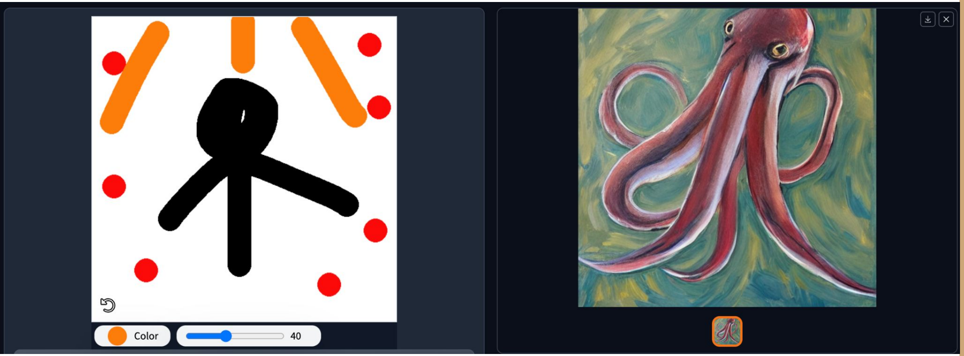
- **Background: a blue ocean with gentle waves under the sunlight**
- **Segment 1: a red squid with long tentacles swimming near the ocean surface**
- **Segment 2: a red rose floating on the water**
- **Segment 3: golden sunlight shining through the water**

Result - Dense Diffusion - 2



The results show that Dense Diffusion may have difficulty in allocating attention evenly in multi-segment scenarios. Segment 1 (squid) has a more complex content and clear semantics, so it has a higher weight in attention allocation, while the attention of segments 2 and 3 may be ignored.

Result - Dense Diffusion - 3



The model generated a red squid that basically meets the description, but "rose" does not appear in the generated results, and elements related to "ocean" and "sunlight" are not significantly reflected, which shows that Dense Diffusion dynamic attention modulation has attention dispersion problems when dealing with multi-segment and multi-semantic cues.

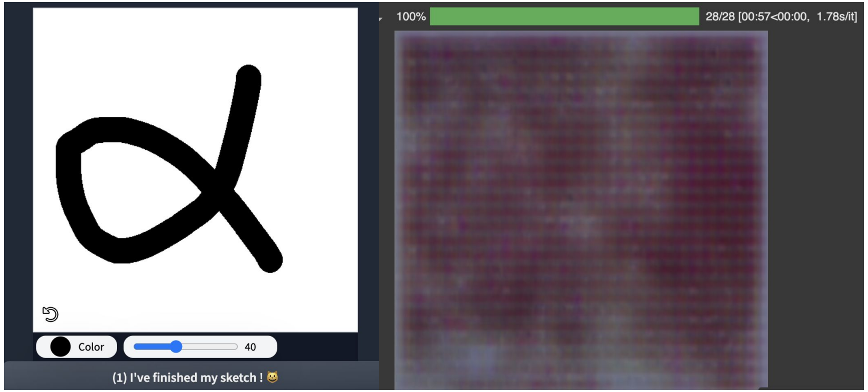
Result - Stable Diffusion v3.5 - 1

The images generated by Stable Diffusion 3.5 under the same prompt words show that the fineness and accuracy have been improved, and all the provided keywords have been better covered.

Result - Stable Diffusion v3.5 - 2

The image generated with stable diffusion 3.5-medium shows that the quality is still very high, even though there are some unreasonable places, such as the suspension and fracture of the octopus tentacles.

Result - Transfer onto SD 3.5



- **The model struggled to eliminate noise interference during image regeneration, leading to low consistency in the denoising process.**

- **The findings suggest defects in the attention mechanism, affecting the final output.**

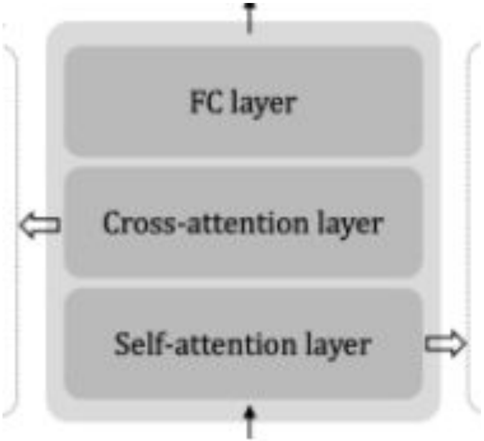
Future Work - 1

```
# import diffusers
# import transformers
# print("Diffusers version:", diffusers.__version__)
# print("Transformers version:", transformers.__version__)
```

```
Diffusers version: 0.31.0
Transformers version: 4.47.1
```

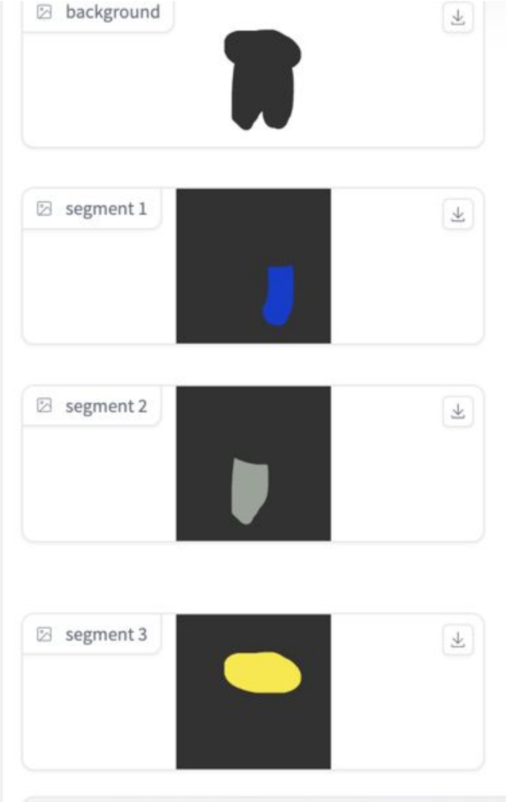
- **Diffusion/Transformer Package Version different**
- **Original Dense Diffusion Model based on diffusers==0.20.2 which not support SD v3.5**
- **Further explore suitable package version to improve stable and compatibility**

Future Work - 2



- Attention modulation logic problem
- Since Stable Diffusion already have aligning text and image features through cross attention and self-attention requires more detailed adjustments to this mechanism to adapt to the SD3.5 architecture

Future Work - 3



- The representation of the segment mask in the latent variable space must be consistent with the generation process.

- If masks are not used correctly when generating latent variables, the resulting image will not perform well.

Question Time

Anything that need further explanation?

https://blog.slido.com/open-ended-questions/

Thanks for your attention! :)

... and looking forward to sharing more findings with you in the future!

References

https://huggingface.co/stabilityai/stable-diffusion-3.5-medium

https://stability.ai/news/introducing-stable-diffusion-3-5

https://huggingface.co/stabilityai/stable-diffusion-3.5-large/blob/main/mmdit.png

https://huggingface.co/spaces/stabilityai/stable-diffusion-3.5-large

https://github.com/naver-ai/DenseDiffusion

https://arxiv.org/abs/2308.12964