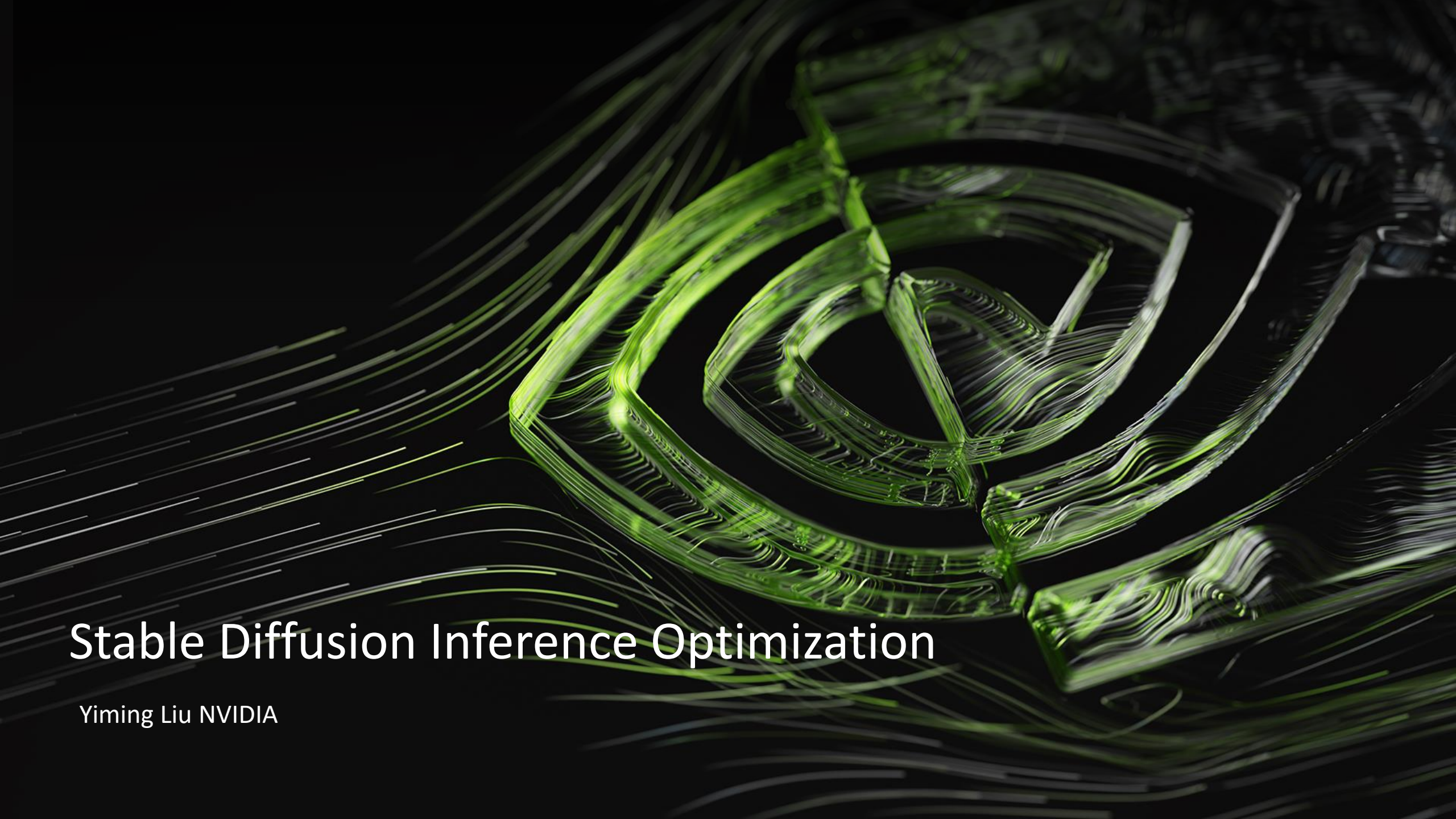
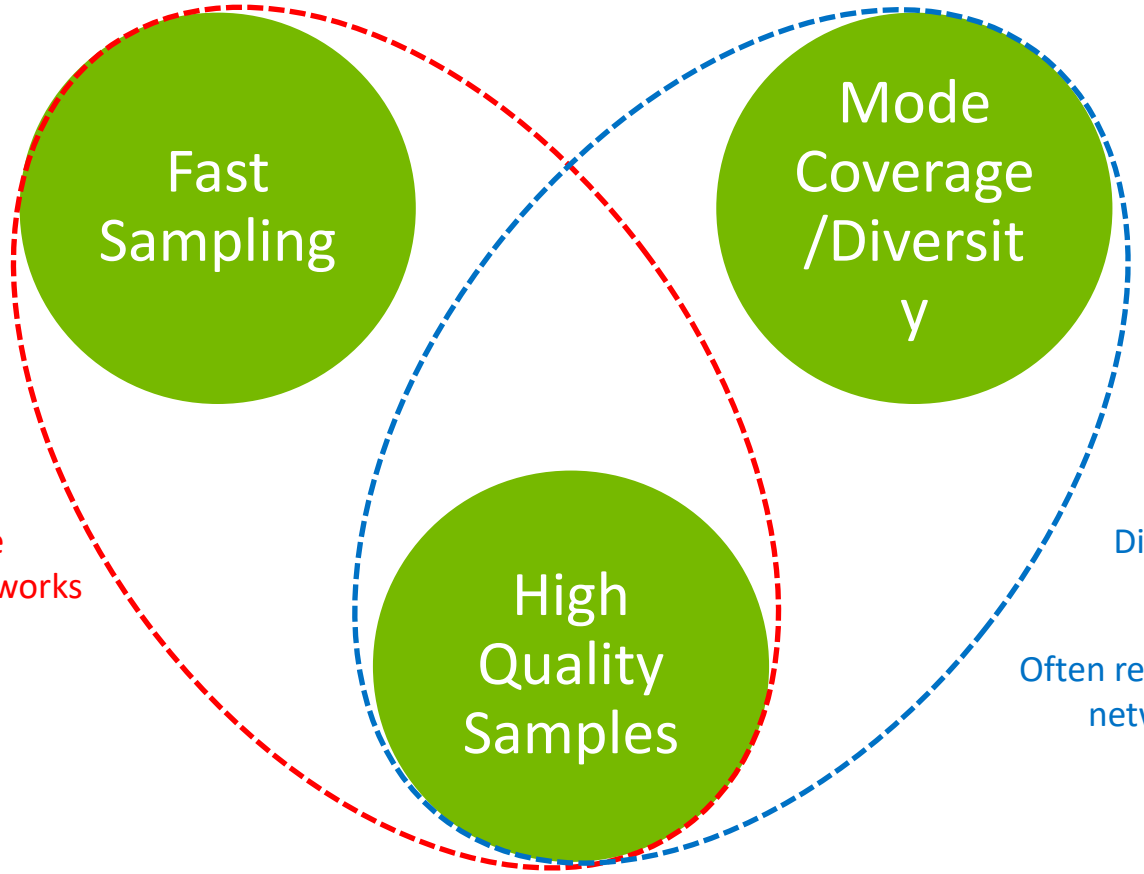


Stable Diffusion Inference Optimization

Yiming Liu NVIDIA

What makes a good generative model



Generative Adversarial Networks (GANs)

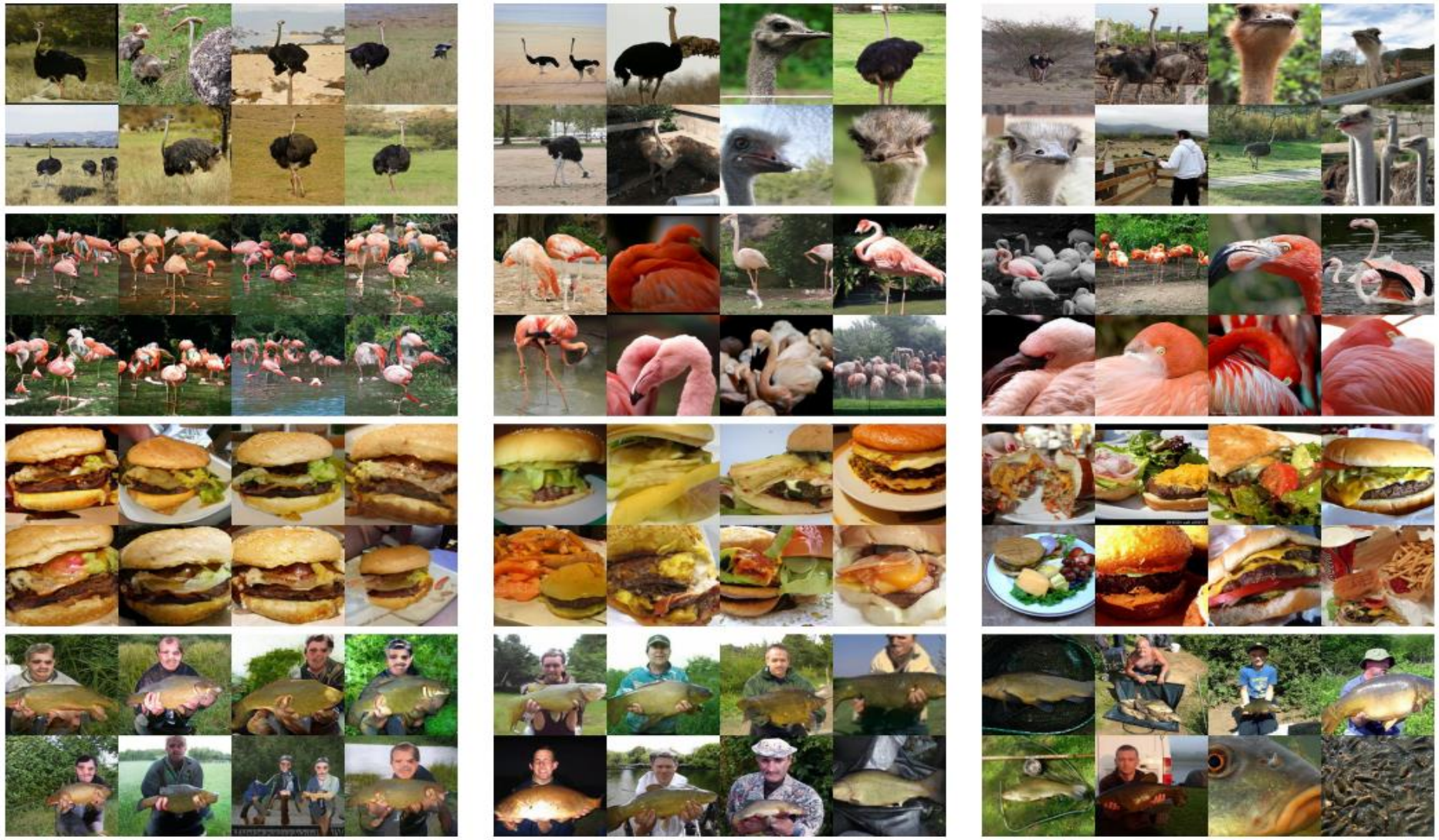
Diffusion Models

Often requires up to 1000s of network evaluations

Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

Sample Diversity

GANs | Diffusion | Real Data



Diffusion Models Beat GANs on Image Synthesis

Diffusion Process

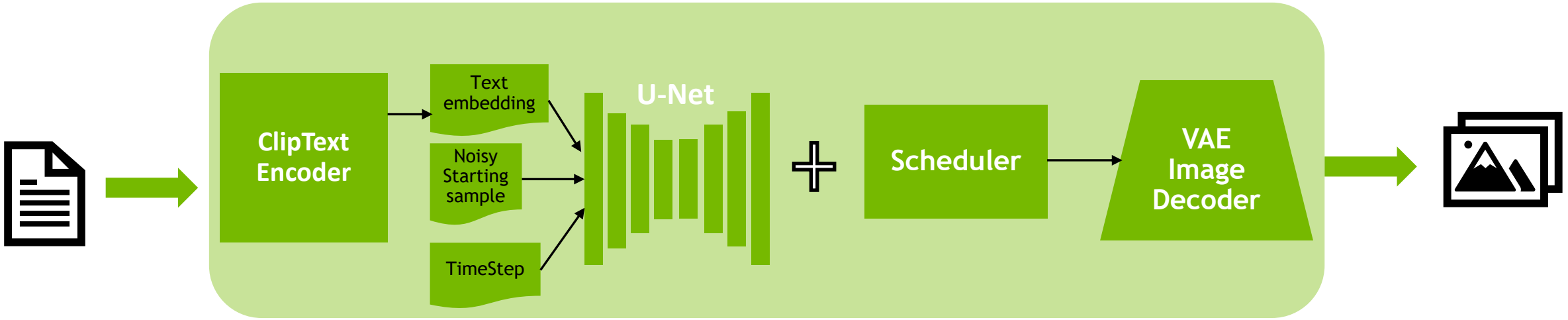
Simple forward diffusion process slowly maps data to noise

Data
x_0

Noise
x_T

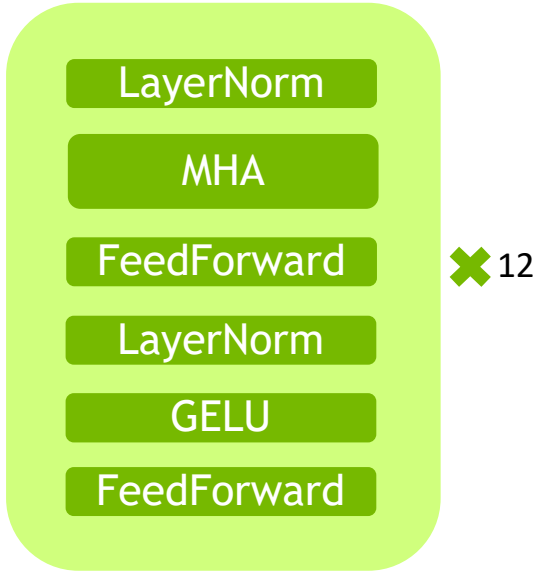
Reverse denoising process maps noise back to data where diffusion model is trained

Stable Diffusion Inference Pipeline



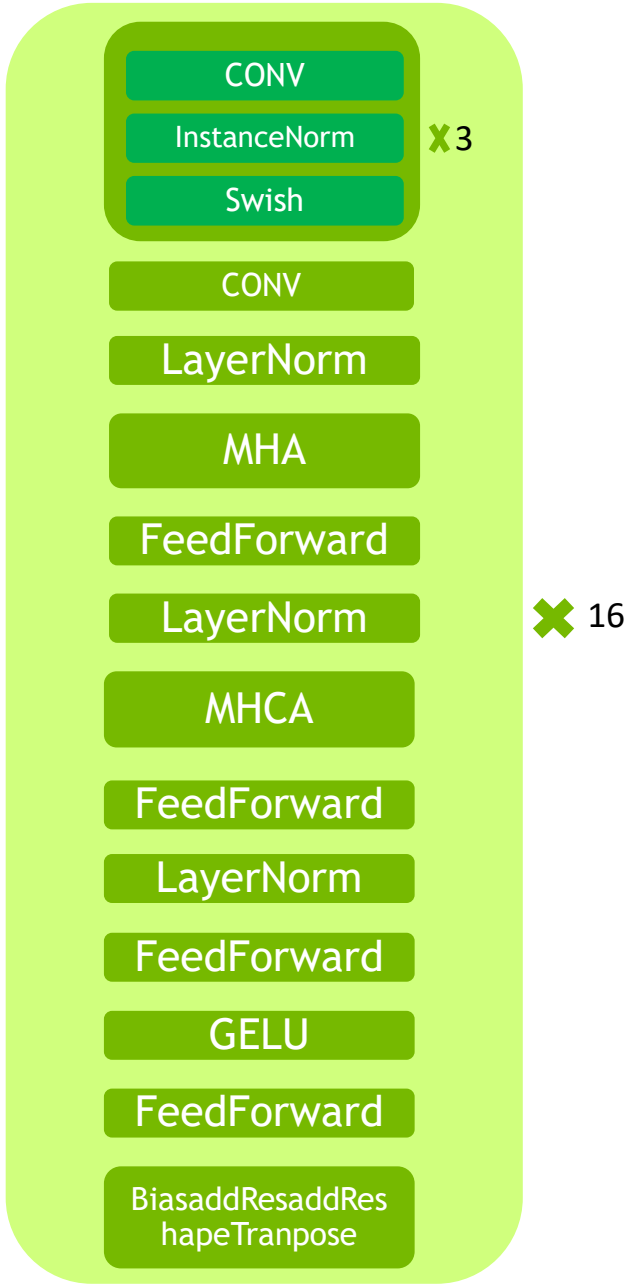
- **ClipText** for text encoding.
  Input: text.
  Output: 77 token embeddings vectors, each in 768 dimensions
- **U-Net + Scheduler** to gradually perform denoising in the information (latent) space.
  Input: text embeddings and a starting multi-dimensional array (structured lists of numbers, also called a *tensor*) made up of noise.
  Output: A processed information array
- **Autoencoder Decoder** that paints the final image using the processed information array.
  Input: The processed information array (dimensions: (4,64,64))
  Output: The resulting image (dimensions: (3, 512, 512) which are (red/green/blue, width, height))

CLIP



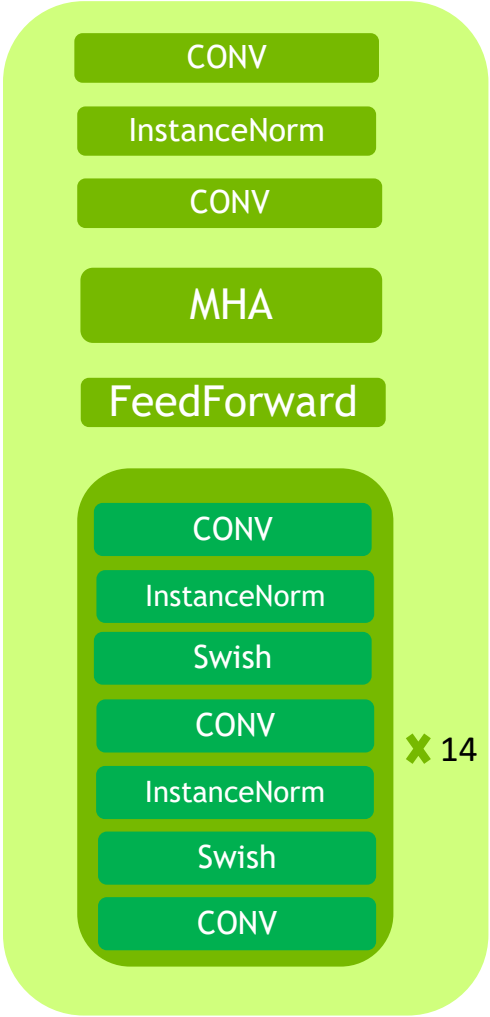
123M parameters

U-NET



859M parameters

VAE



49M parameters

Floating point operations measured under:

- **B=1**
- **H=W=64**

Starting Latent State: Sample Gaussian Noise from **random seed**

For **classifier-free guidance**

User prompt

2Bx4xHxW, Latent

Update/denoise Latent

Floating point operations: **0.02714**x10^{12}

Clip text encoder

Negative prompt: Empty token if not provided

Scheduler

2Bx77x768, text embeddings

TimeStep:t

U-Net

Repeat T=50 times

Floating point operations: **1.67952**x10^{12}

Floating point operations: **83.976**x10^{12}

2Bx4xHxW, **predicted Noise**

Output image:Bx3x8Hx8W

Bx4xHxW

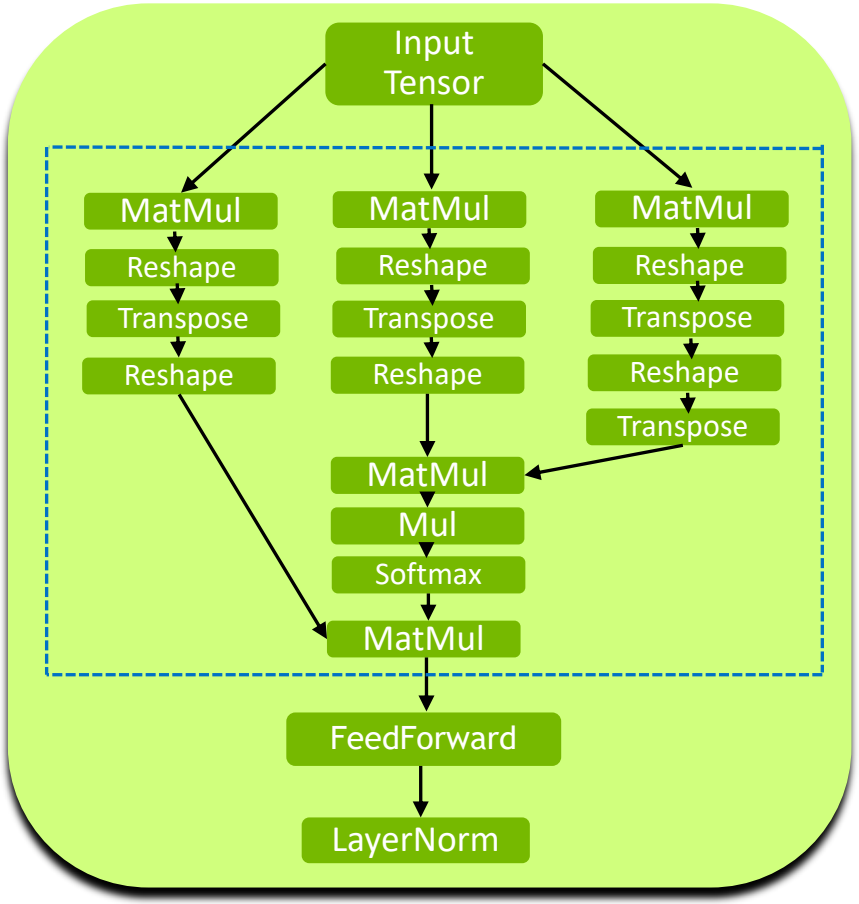
Final denoised latent

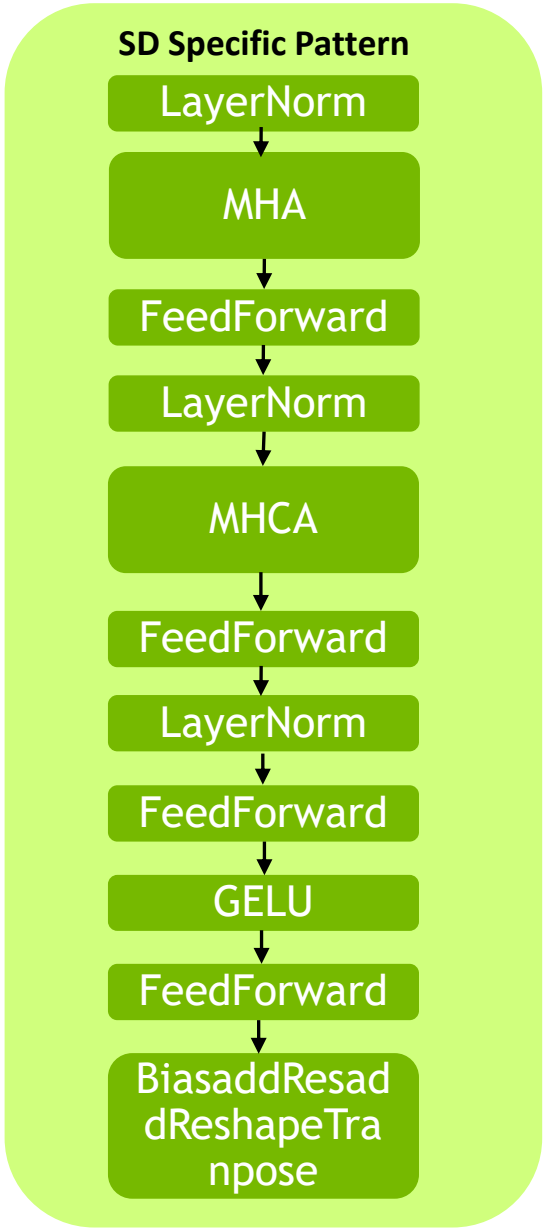
VAE Image Decoder

Floating point operations: **2.53933**x10^{12}

NVIDIA

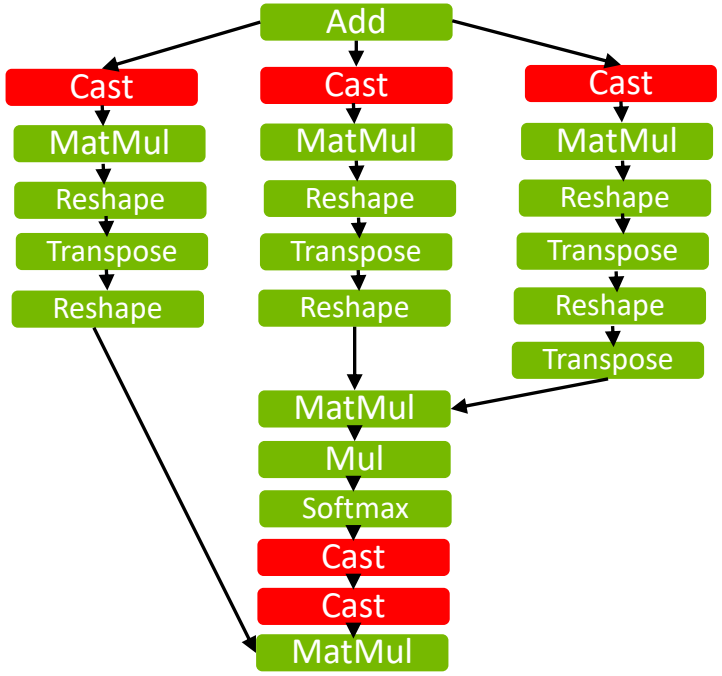
How Does TensorRT Handle Attentions(since 8.0)

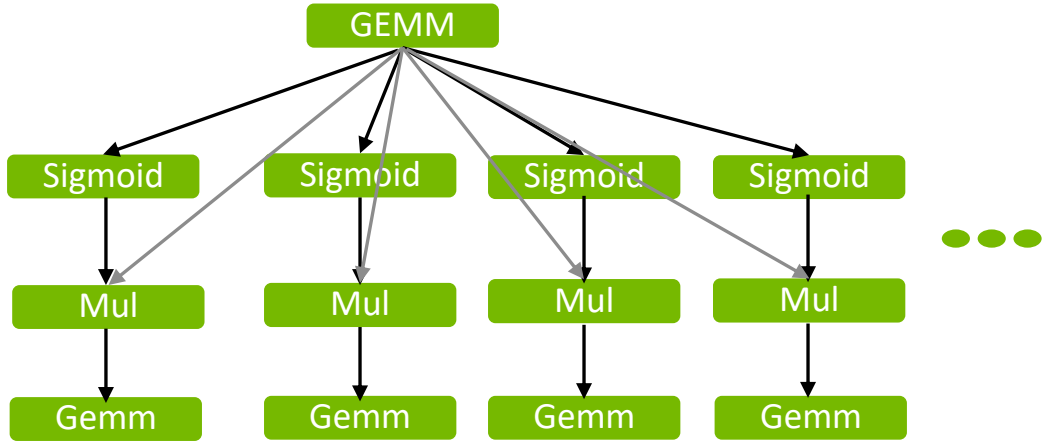




- Detect the MHA pattern
- Try to offload MHA and its surrounding structures to **Myelin** as much as possible

Utilize Myelin





- Remove redundant Cast onnx nodes to help TensorRT's **Myelin** backend compiler recognize the pattern as an MHA(not needed after TRT8.5.2)
- Merge repeated Swish ops

Before Simplification

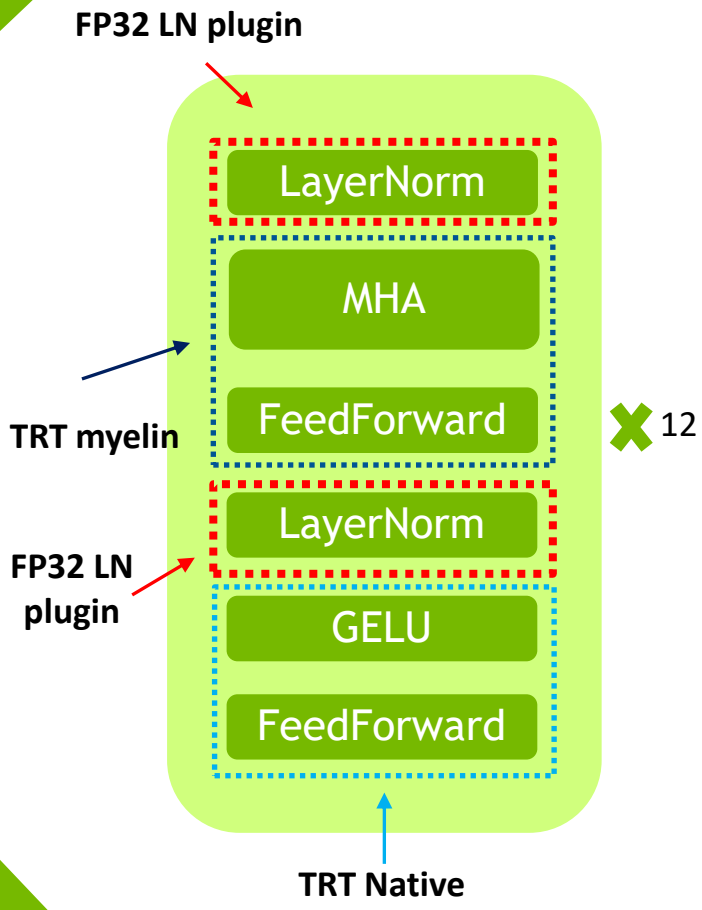

```
    "Dimensions": [1024,640,1,1],
    "Format/Datatype": "Row major linear FP16 format"
  }],
  "TacticValue": "0x0000000000000000"
},{
  "Name": "reshape_after_MatMul_5396 + Reshape_5419 + Transpose_5420",
  "LayerType": "Shuffle",
  "Inputs": [
  {
    "Name": "Reformatted Input Tensor 0 to reshape_after_MatMul_5396 + Reshape_5419 + Transpose_5420",
    "Location": "Device",
    "Dimensions": [1024,640,1,1],
    "Format/Datatype": "Row major linear FP16 format"
  }],
  "Outputs": [
  {
    "Name": "onnx::Reshape_6685",
    "Location": "Device",
    "Dimensions": [1,8,1024,80],
    "Format/Datatype": "Row major linear FP16 format"
  }],
  "ParameterType": "Shuffle",
  "FirstTranspose": [0,1,2,3],
  "Reshape": [1,1024,8,80],
  "SecondTranspose": [0,2,1,3],
  "ZeroIsPlaceholder": 1,
  "TacticValue": "0x0000000000000000"
},{
  "Name": "Reshape_5434",
  "LayerType": "NoOp",
  "Inputs": [
  {
    "Name": "onnx::Reshape_6685",
    "Location": "Device",
    "Dimensions": [1,8,1024,80],
    "Format/Datatype": "Row major linear FP16 format"
  }],
  "Outputs": [
  {
    "Name": "query.171",
    "Location": "Device",
    "Dimensions": [8,1024,80],
    "Format/Datatype": "Row major linear FP16 format"
```


After Simplification

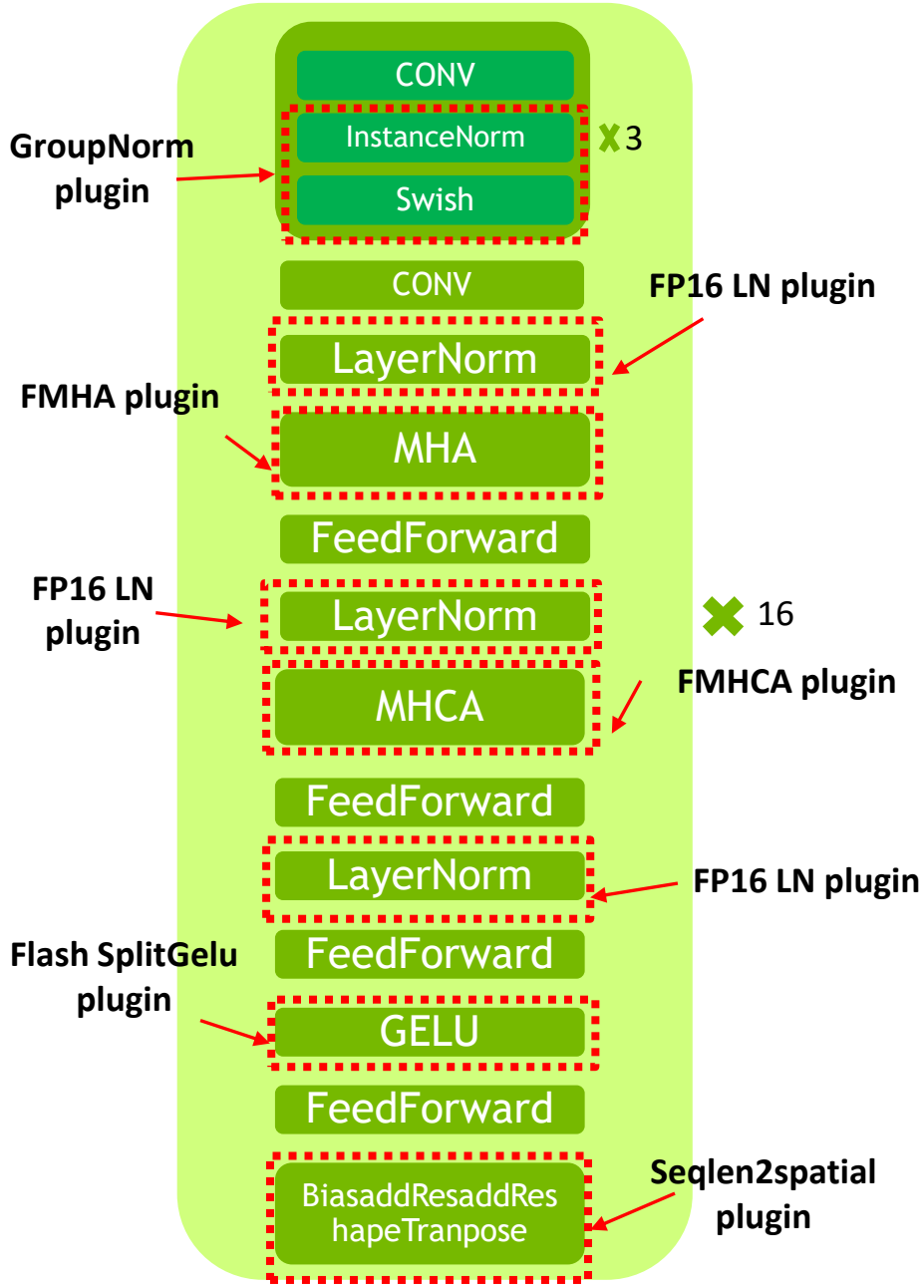

```
  "Activation": "NONE",
  "HasBias": 1,
  "HasReLU": 0,
  "TacticName": "ampere_h16816cudnn_128x128_ldg8_relu_exp_stages_64x3_interior_nhwc_tn_v1",
  "TacticValue": "0xdbe058db40209ee1"
},{
  "Name": "{ForeignNode[onnx::MatMul_9584 + (Unnamed Layer* 1263) [Shuffle]...Reshape_637 + Transpose_638]}",
  "LayerType": "Myelin",
  "Inputs": [
  {
    "Name": "onnx::Shape_808",
    "Location": "Device",
    "Dimensions": [1,320,64,64],
    "Format/Datatype": "Channel major FP16 format where channel % 8 == 0"
  },
  {
    "Name": "encoder_hidden_states",
    "Location": "Device",
    "Dimensions": [1,77,768],
    "Format/Datatype": "Row major linear FP16 format"
  }],
  "Outputs": [
  {
    "Name": "input.64",
    "Location": "Device",
    "Dimensions": [1,320,64,64],
    "Format/Datatype": "Channel major FP16 format where channel % 8 == 0"
  }],
  "TacticValue": "0x0000000000000000"
},{
  "Name": "InstanceBiasV-3",
  "LayerType": "Constant",
  "Inputs": [],
  "Outputs": [
  {
    "Name": "(Unnamed Layer* 1564) [Constant]_output",
    "Location": "Device",
    "Dimensions": [1,32,1],
    "Format/Datatype": "Row major linear FP16 format"
  }],
  "ParameterType": "Constant",
  "weights": {"Type": "Float", "Count": 32},
  "dimensions": [1,32,1],
```


To obtain the layer info json, you can build the engine with trtexec(https://github.com/NVIDIA/TensorRT/tree/main/samples/trtexec), with --exportLayerInfo=layerinfo.json --profilingVerbosity=detailed flags enabled.

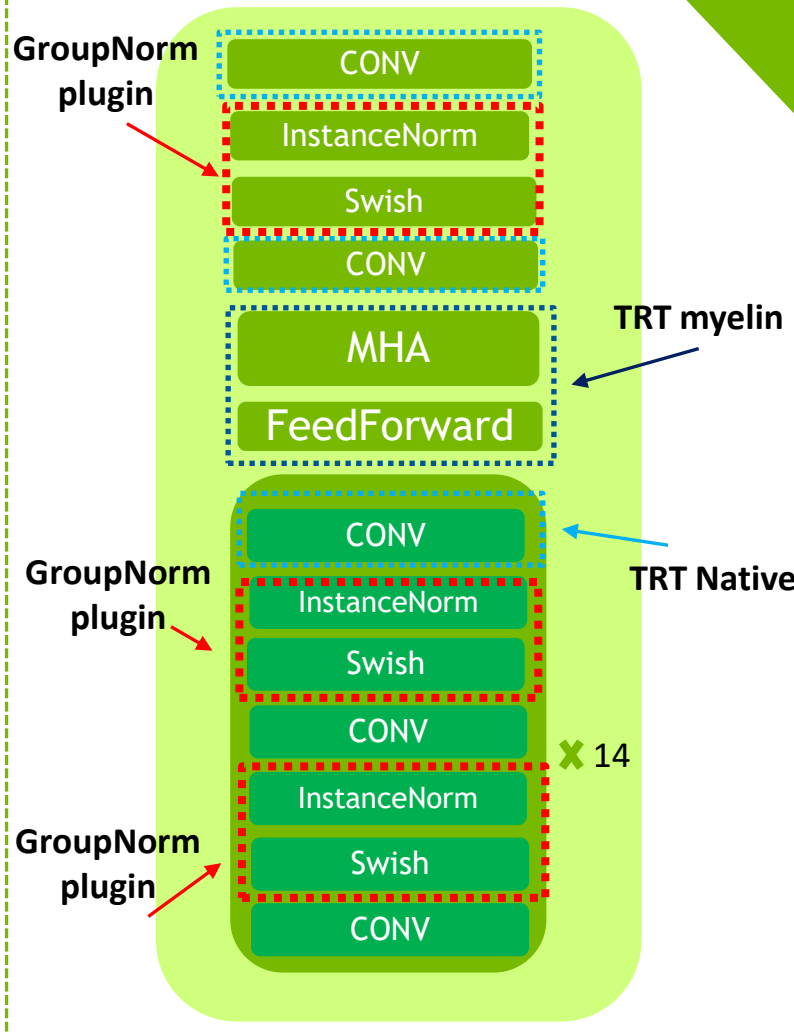
CLIP



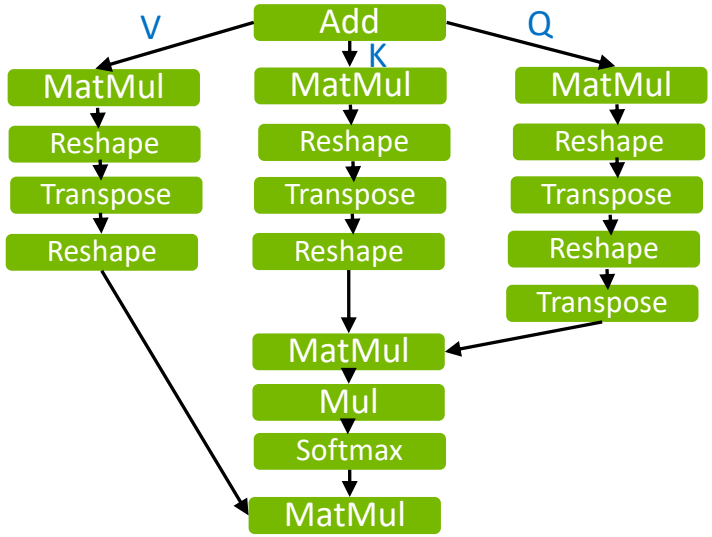
U-NET

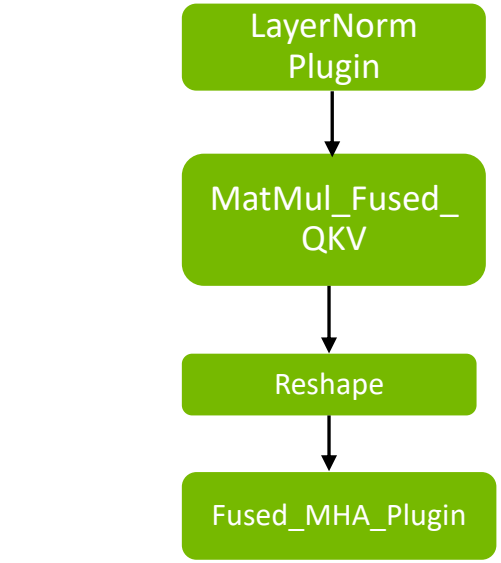


VAE



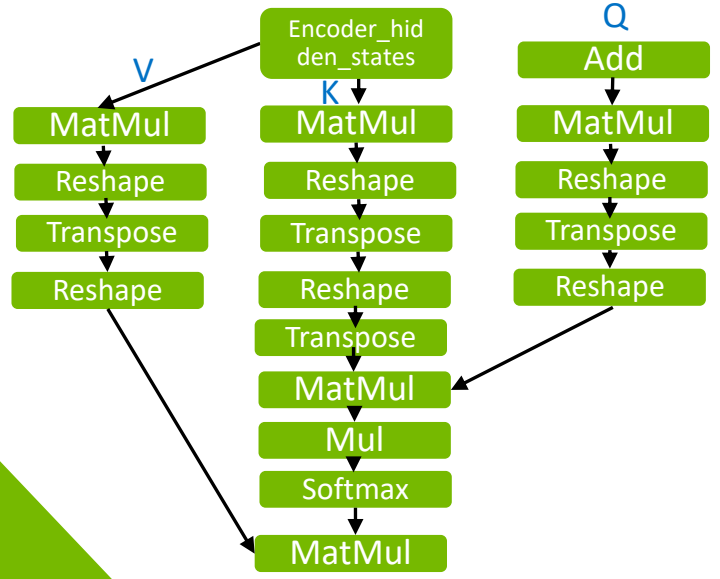
Attentions

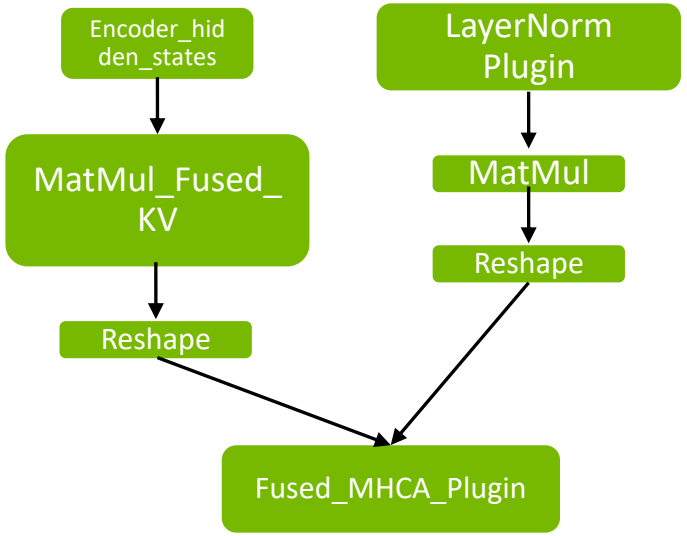




Convert Attention blocks to either

- **Fused multi-head self attention**(https://github.com/NVIDIA/TensorRT/tree/main/plugin/multiHeadFlashAttentionPlugin)
- **Fused multi-head cross attention** (https://github.com/NVIDIA/TensorRT/tree/main/plugin/multiHeadCrossAttentionPlugin)

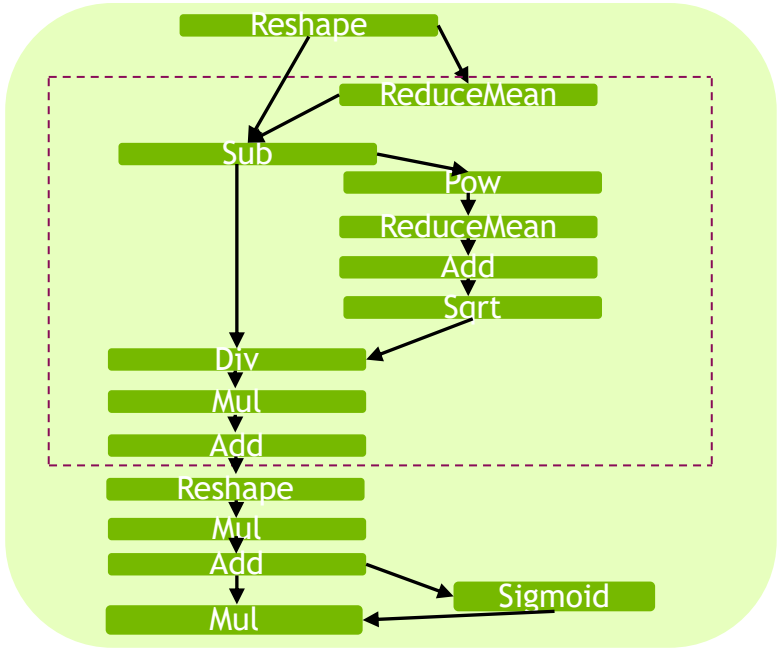




Normalization

Breakup InstanceNormalization into primitive ops

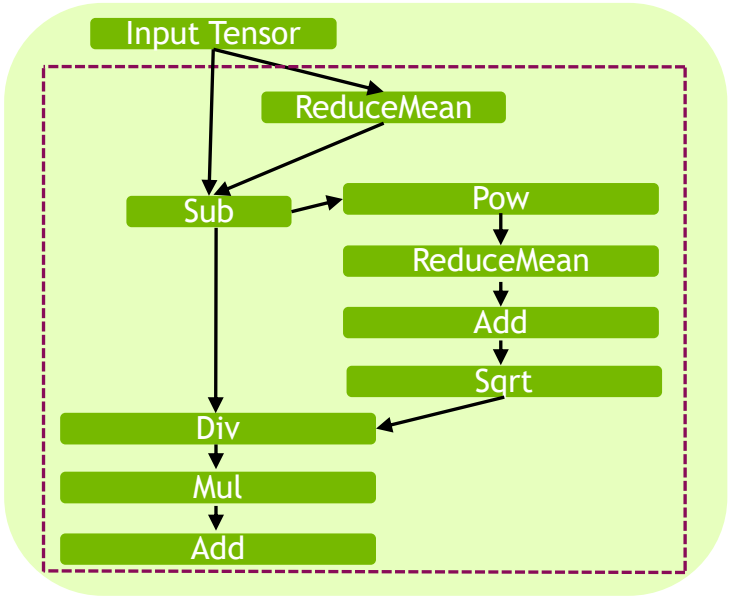
InstanceNormalization



Replace the whole pattern with GroupNorm Plugin

https://github.com/NVIDIA/TensorRT/tree/main/plugin/groupNormPlugin

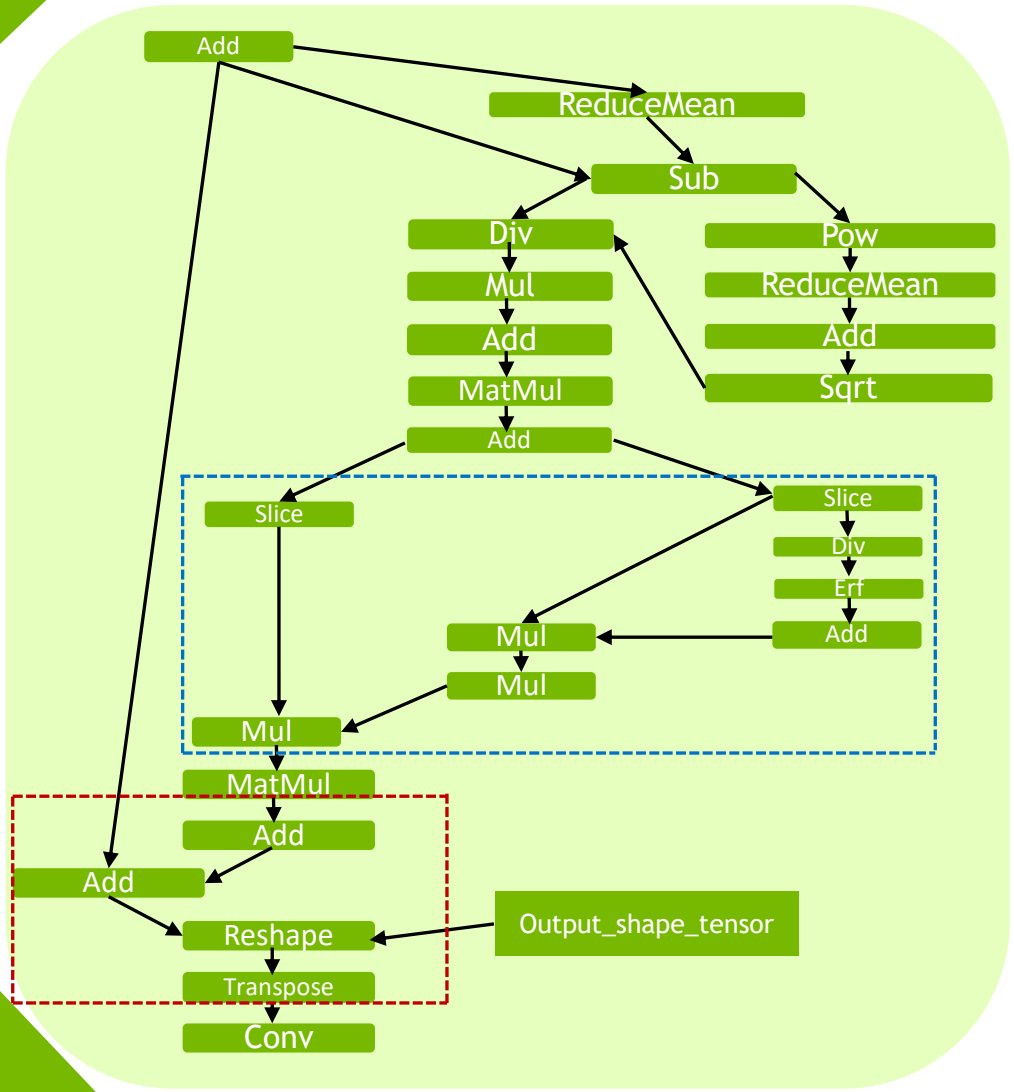
LayerNorm

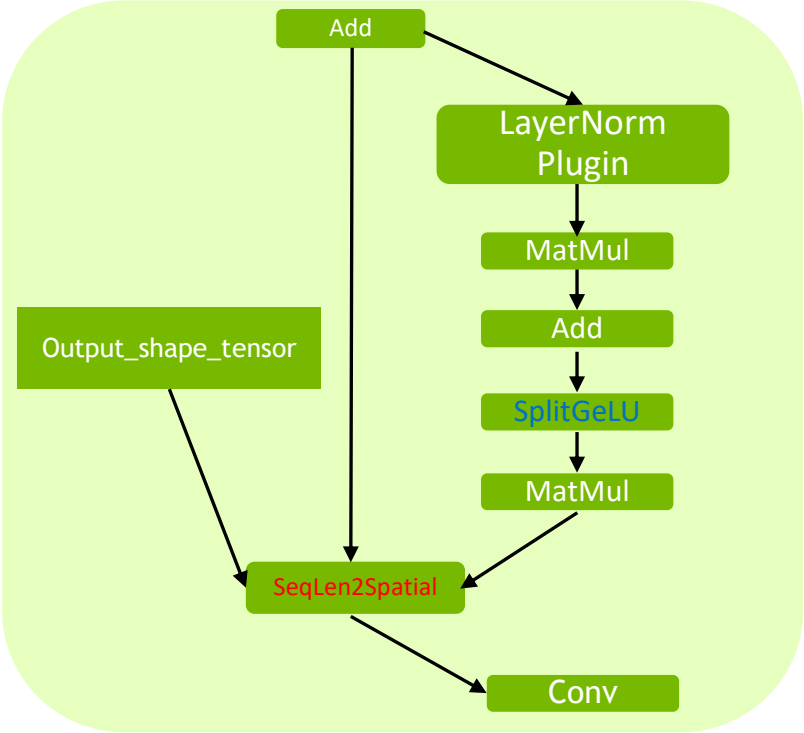


LayerNorm was also picked up by a plugin, since Myelin backend compiler was not used to optimize the MHA pattern

https://github.com/NVIDIA/TensorRT/tree/main/plugin/layerNormPlugin

Other Small Fusions(SplitGeLU+SeqLen2Spatial)





*Additionally, we also employ SplitGelu plugin to fuse ops around Erf and seq2spatial plugin to fuse the biasadd+residualadd+reshape+transpose pattern

DemoDiffusion(https://github.com/NVIDIA/TensorRT/tree/main/demo/Diffusion)

Decoupled design and in an OOP style

Utilities.py

Provide basic utilities, and non-DL related function

- Base engine class
- Scheduler
- Cuda utilities

Models.py

Provide model specific onnx graph optimization strategies

- Base optimizer class and base model class
- CLIP, UNet and VAE specific optimization strategies

demo_diffusion.py

Main entry point of the demo

- CLI interface
- Main DemoDiffusion class to initiate end2end optimization of all three models, build respective engines, and finally run inference with perf profiling support

For instance, to run a simple demo

```
LD_PRELOAD=${PLUGIN_LIBS} python3 demo-diffusion.py "a beautiful photograph of Mt. Fuji during cherry blossom" --hf-token=$HF_TOKEN -v
```

DemoDiffusion Benchmark

Optimization Level	UNETx50
OOTB(out of the box)	2288.14ms
Myelin (RemoveCast + RemoveInstanceNorm)	1282.32ms
Above + fMHA +fMHCA	1047.26ms
Above + LayerNorm	979.72ms
Above + GroupNorm	874.51ms
Above + SplitGeLU + Seq2Spatial	856.47ms
Above + Graph Optimization	805.97ms
Above + Preview Feature 0805	792.26ms

Optimization Level	CLIP
OOTB(out of the box)	3.01ms
Myelin (RemoveCast + RemoveInstanceNorm)	2.96ms
Above + LayerNorm	4.38ms

Optimization Level	VAE
OOTB(out of the box)	20.73ms
Myelin (RemoveCast + RemoveInstanceNorm)	20.97ms
Above + GroupNorm	18.54ms

TensorRT8.5.2.2, batch=1, image size=512x512, GPU=A100-80Gib-PCIE

Pipeline Total Time

815.18ms

Support dynamic shapes image generation up to 1024x1024

The optimized pipeline was mainly developed by Devtech China team, Nikita K and Julien D.

TensorRT8.5.2, batch=1, image size=512x512,GPU=L40

Optimization Level	CLIP	UNETx50	VAE	Pipeline
ALL Plugins	2.42ms	973.56ms	23.98ms	999.98ms

TensorRT8.5.1, batch=1, image size=512x512,GPU=T4

Optimization Level	CLIP	UNETx50	VAE	Pipeline
ALL Plugins	9.70ms	4536.414ms	104.944ms	4651.088ms

TensorRT8.5.1, batch=1, image size=512x512,GPU=A10

Optimization Level	CLIP	UNETx50	VAE	Pipeline
ALL Plugins	5.14ms	1892.144ms	40.180ms	1937.497ms

TensorRT8.5.1, batch=1, image size=512x512,GPU=A30

Optimization Level	CLIP	UNETx50	VAE	Pipeline
ALL Plugins	4.97ms	1368.08ms	30.34ms	1403.43ms