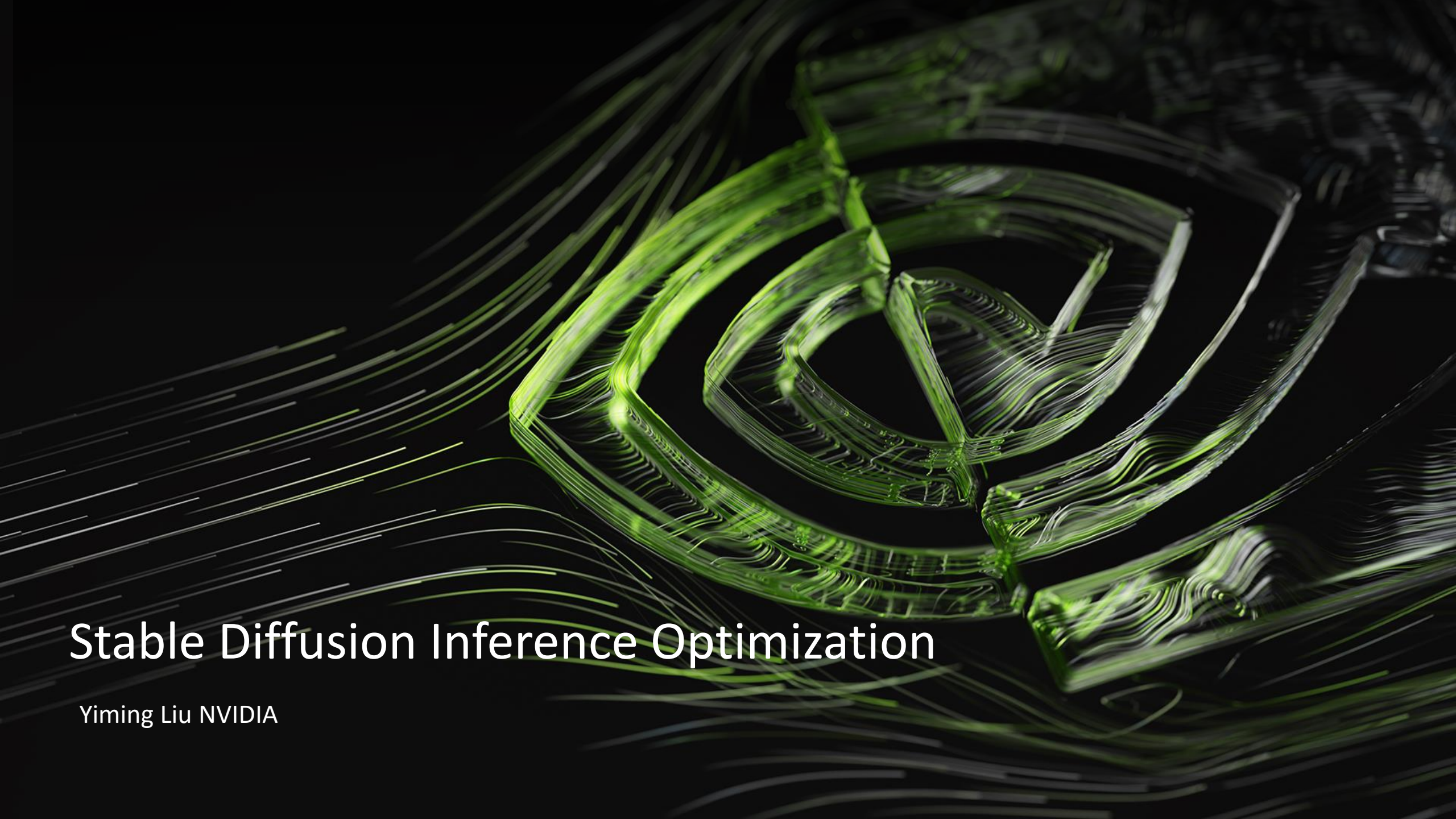
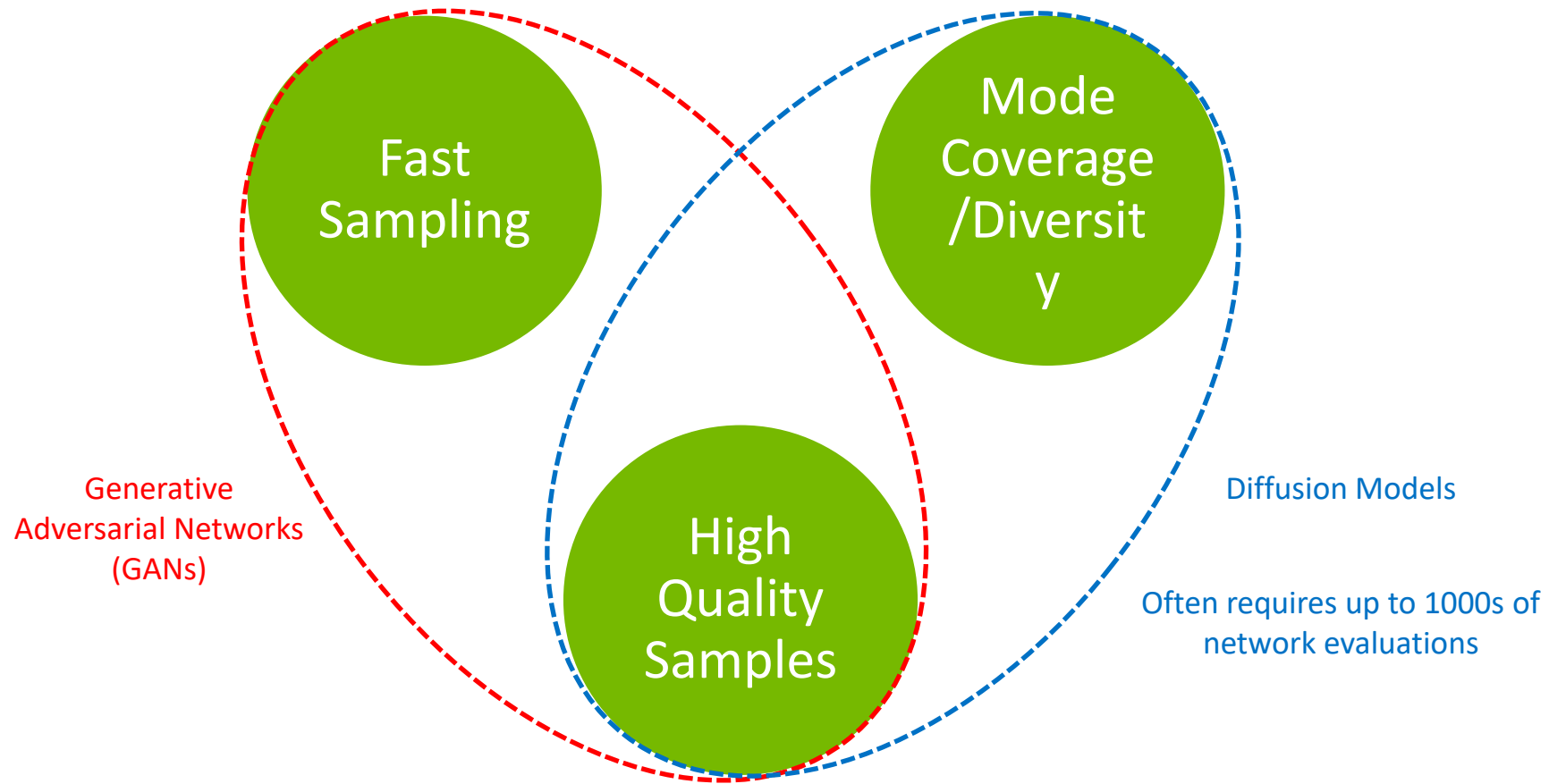


The background is a dark, almost black, space filled with numerous thin, bright green lines that flow and swirl in various directions, creating a sense of motion and energy. In the center-right area, there is a more complex, glowing green structure that resembles a stylized, multi-layered geometric shape or a molecular model, with sharp edges and internal details that catch the light.

Stable Diffusion Inference Optimization

Yiming Liu NVIDIA

What makes a good generative model



Sample Diversity

GANs



Diffusion

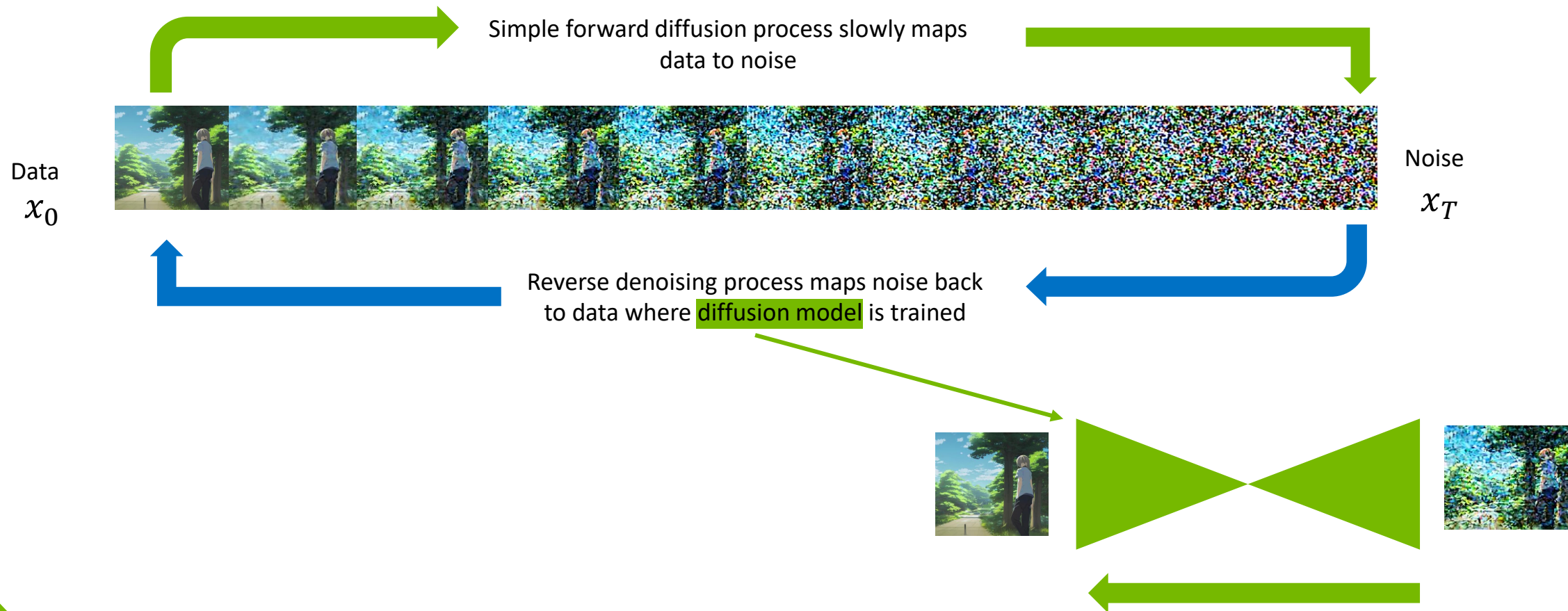


Real Data

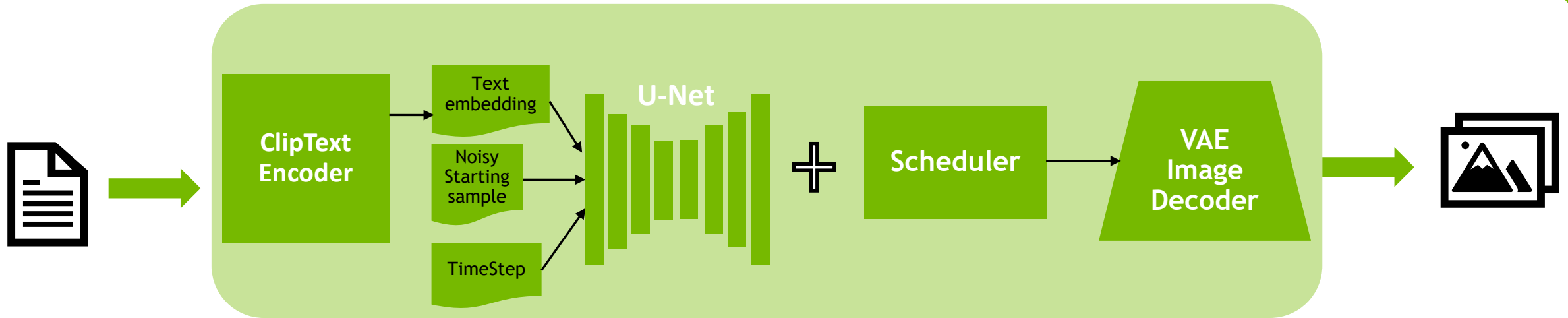


Diffusion Models Beat GANs on Image Synthesis

Diffusion Process

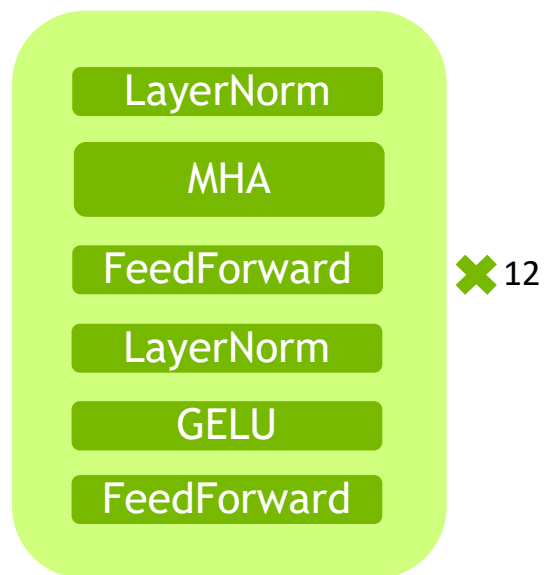


Stable Diffusion Inference Pipeline



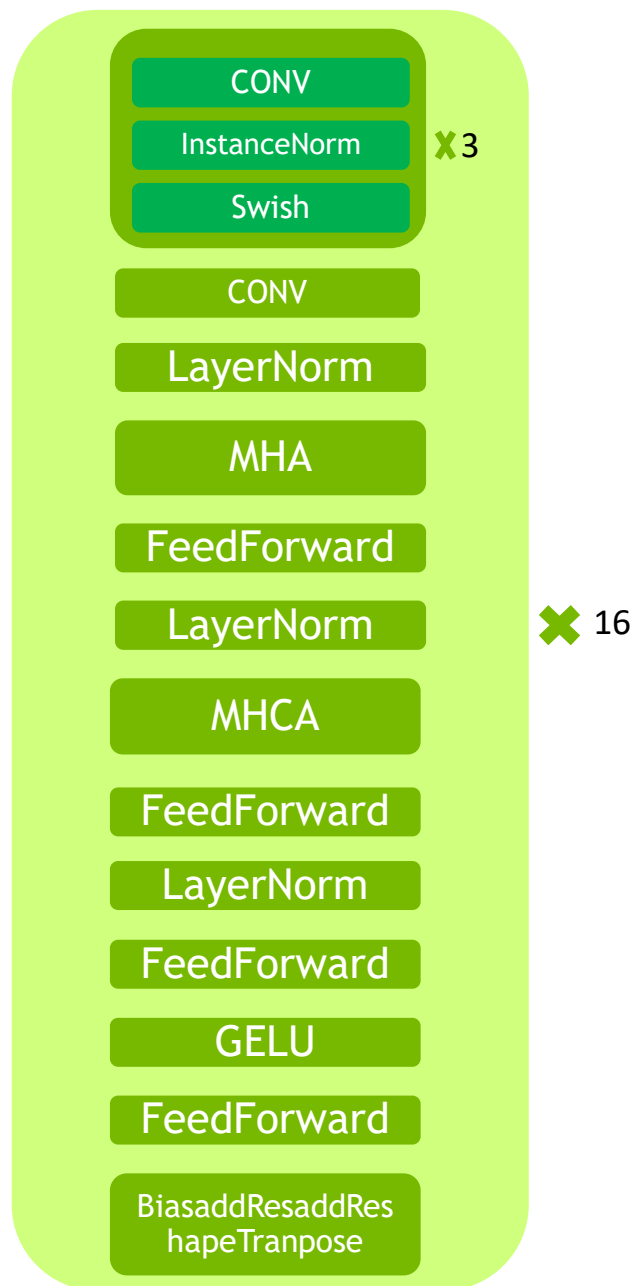
- **ClipText** for text encoding.
Input: text.
Output: 77 token embeddings vectors, each in 768 dimensions
- **U-Net + Scheduler** to gradually perform denoising in the information (latent) space.
Input: text embeddings and a starting multi-dimensional array (structured lists of numbers, also called 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



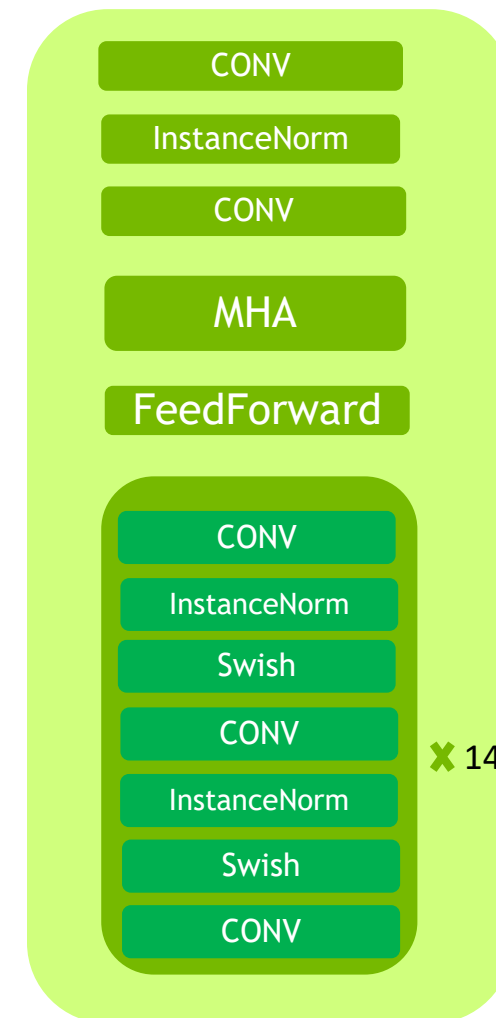
123M parameters

U-NET



859M parameters

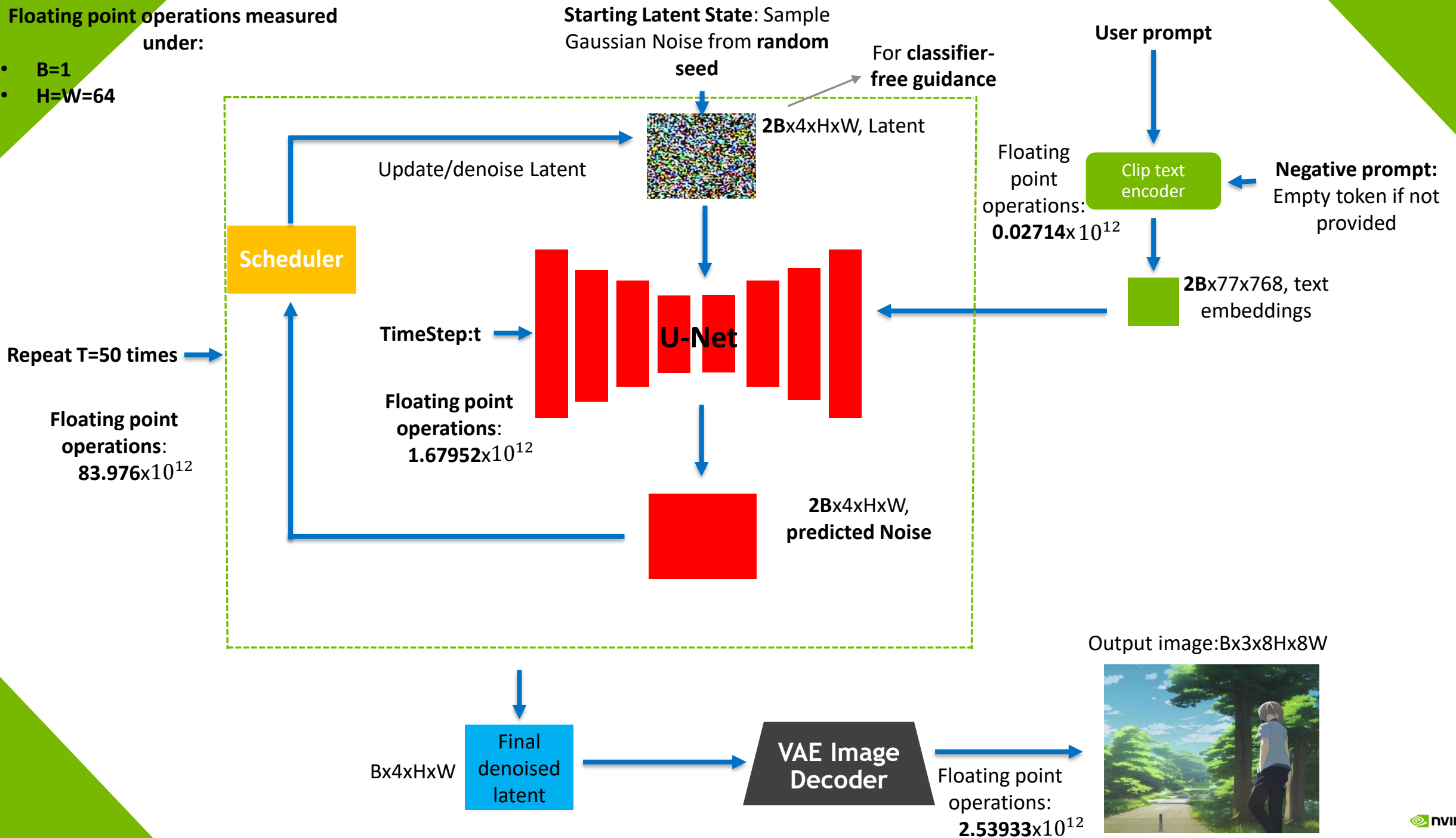
VAE



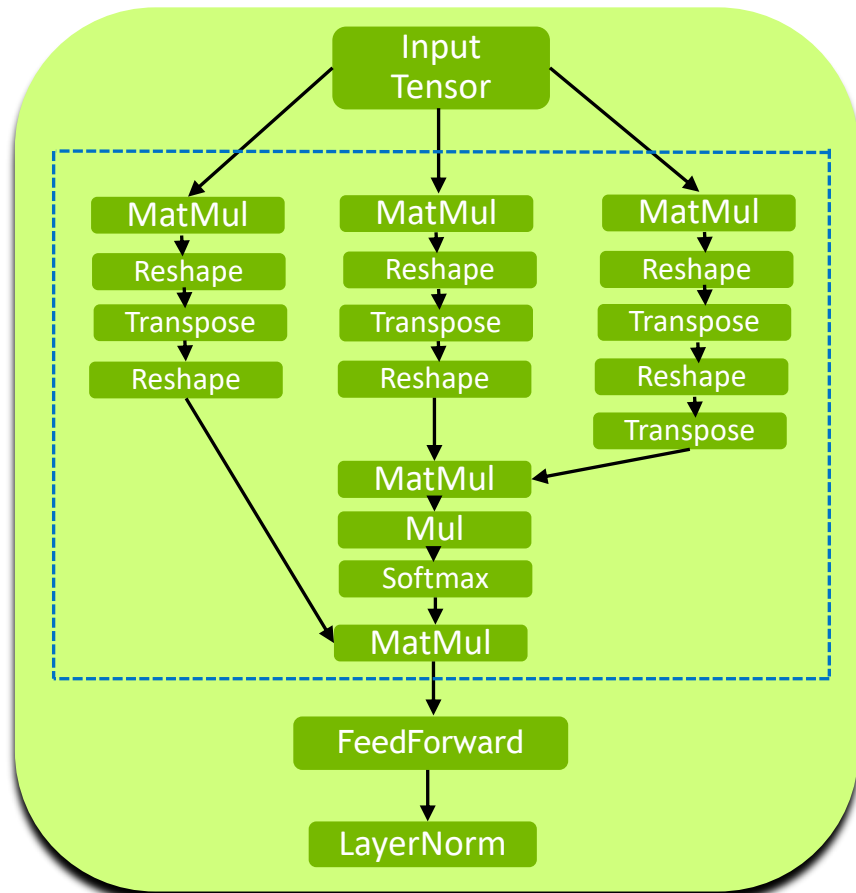
49M parameters

Floating point operations measured under:

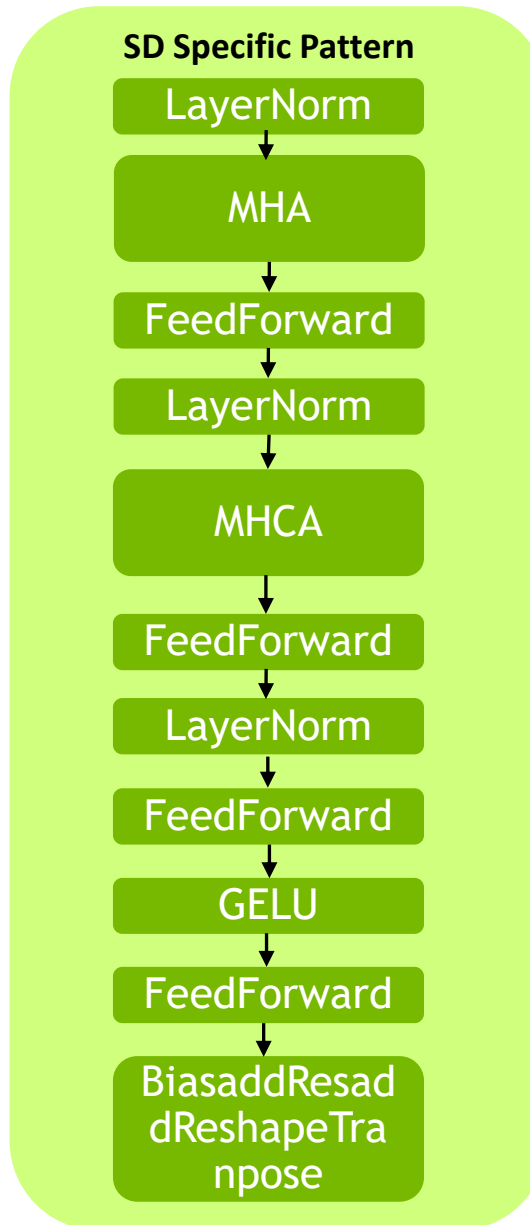
- $B=1$
- $H=W=64$



How Does TensorRT Handle Attentions(since 8.0)

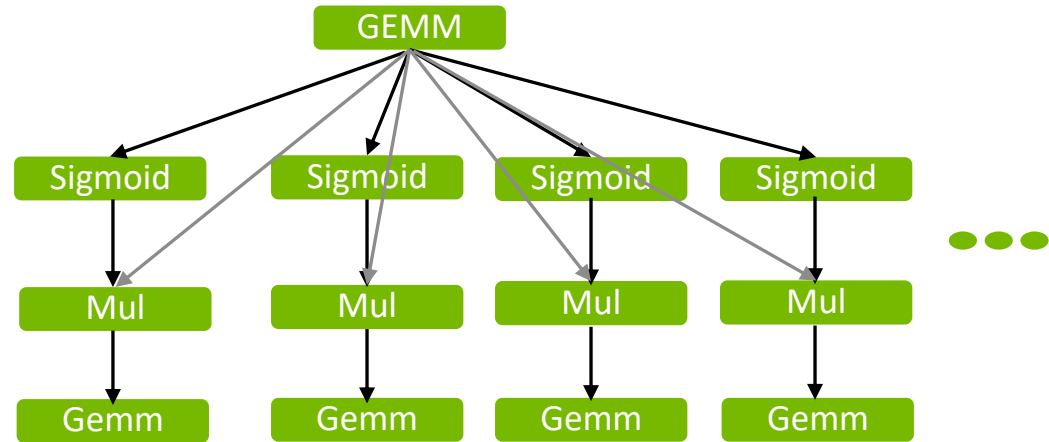
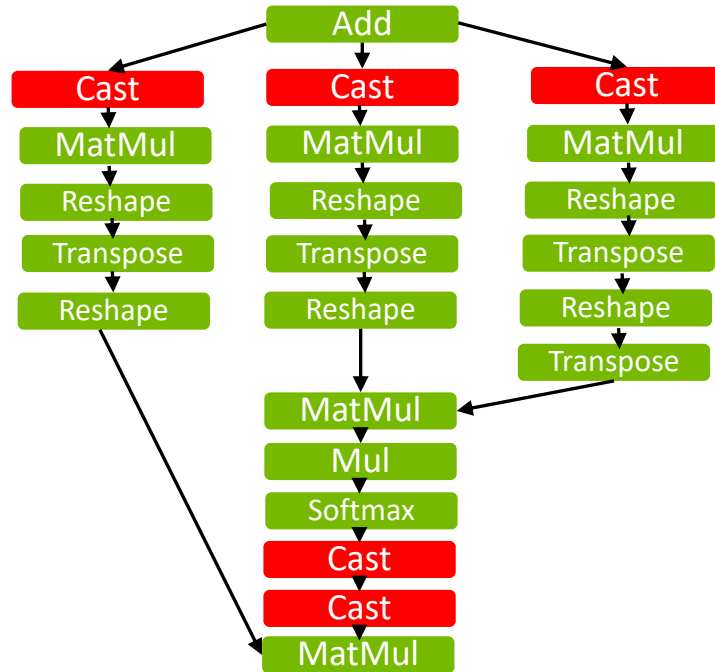


- Detect the MHA pattern
- Try to offload MHA and its surrounding structures to **Myelin** as much as possible



Myelin
ForeignNode

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": "0xdb058db40209ee1"
},{
"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

FP32 LN plugin

TRT myelin

FP32 LN plugin

TRT Native

× 12

U-NET

GroupNorm plugin

FMHA plugin

FP16 LN plugin

Flash SplitGelu plugin

× 3

FP16 LN plugin

× 16

FMHCA plugin

FP16 LN plugin

SeqLen2spatial plugin

VAE

GroupNorm plugin

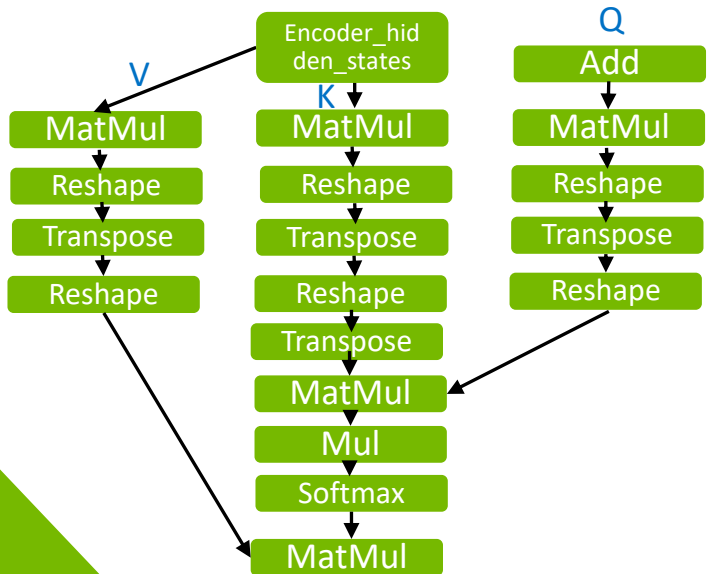
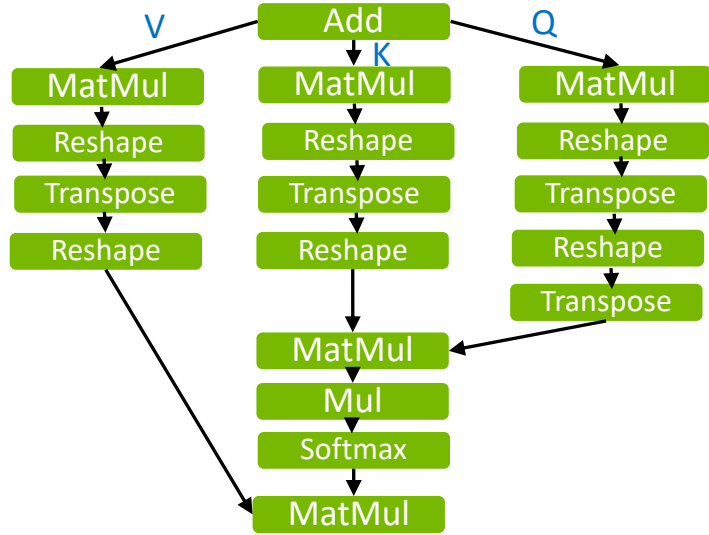
TRT myelin

GroupNorm plugin

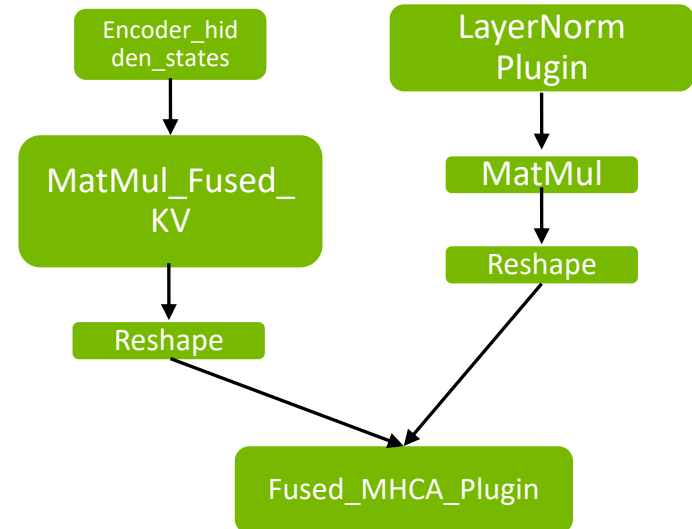
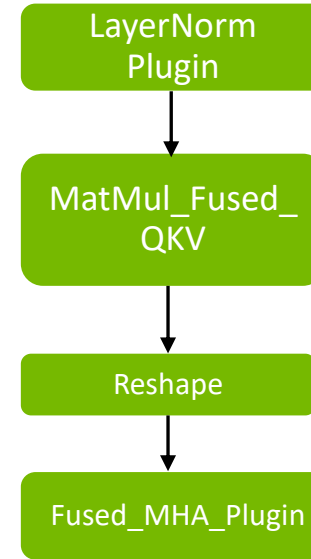
TRT Native

× 14

Attentions

[illegible]

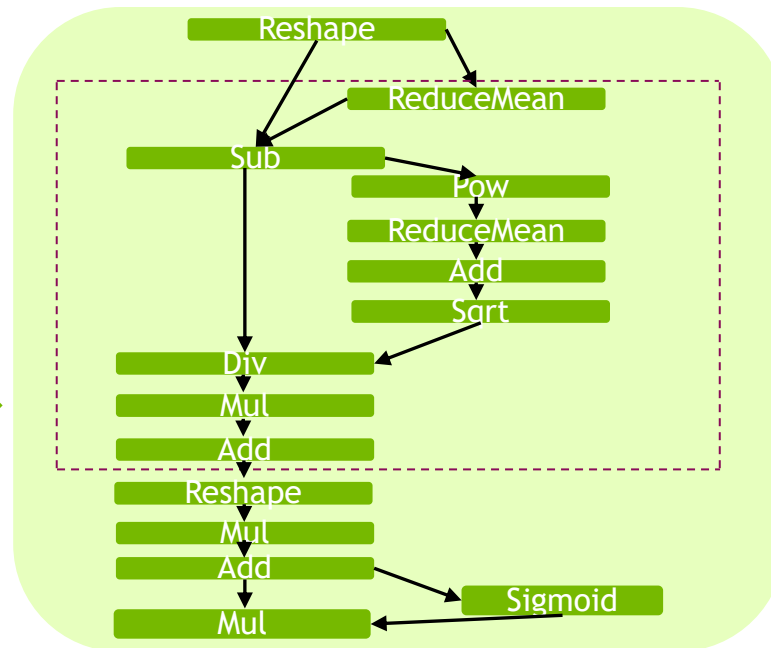
- **Fused multi-head self attention**(<https://github.com/NVIDIA/TensorRT/tree/main/plugin/multiHeadFlashAttentionPlugin>)
- **Fused multi-head cross attention** (<https://github.com/NVIDIA/TensorRT/tree/main/plugin/multiHeadCrossAttentionPlugin>)



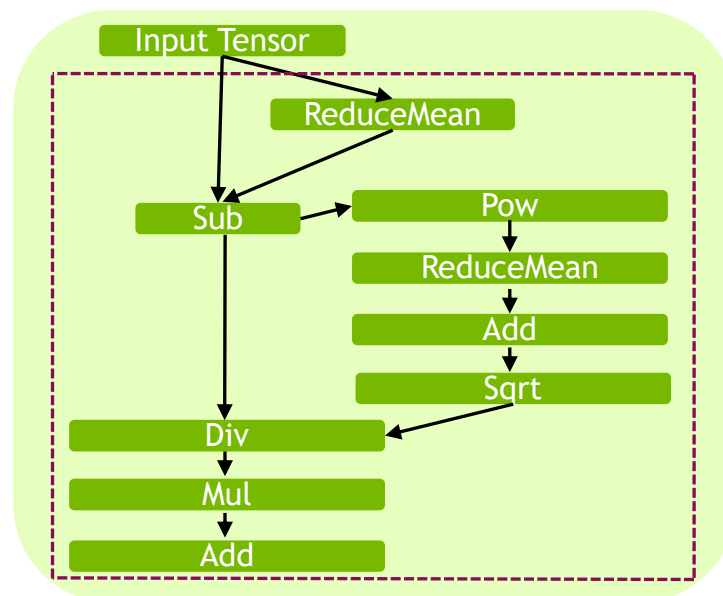
Normalization

Breakup
InstanceNormalization into
primitive ops

InstanceNormalization



LayerNorm



Replace the whole pattern
with GroupNorm Plugin



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

```

#include "groupNormKernel.h"
#include <cuda/cub.cuh>

static inline __device__ __host__ float sigmoid(float x)
{
    return 1.F / (1.F + expf(-x));
}

struct GroupSums
{
    // Is it the 1st element of the group?
    int32_t flag;
    // The sum.
    float sum;
    // The sum of squares.
    float sumSq;
};

struct GroupSumsOp
{
    inline __device__ GroupSums operator()(GroupSums const& a, GroupSums const& b)
    {
        GroupSums dst;
        dst.sum = b.flag ? b.sum : (a.sum + b.sum);
        dst.sumSq = b.flag ? b.sumSq : (a.sumSq + b.sumSq);
        dst.flag = a.flag + b.flag;
        return dst;
    }
};

template <int32_t tTHREADS_PER_BLOCK>
__global__ void groupNormKernel(GroupNormParams params)
{
    // The object in charge of doing the sums for the different blocks.
    typedef cub::BlockScan<GroupSums, tTHREADS_PER_BLOCK> BlockScan;
    // Allocate shared memory for BlockScan.
    __shared__ typename BlockScan::TempStorage tempStorage;
    // Allocate shared memory for the groups. We could reduce the amount of shared
    // memory reserved.
    __shared__ float2 smem[tTHREADS_PER_BLOCK];

    // The instance in the batch.
    int32_t ni = blockIdx.z;
    // The channel loaded by that thread (2 channels per thread for F16x2).
    int32_t ci = blockIdx.x * params.cPerBlock + threadIdx.x * 2;

    // The first activation loaded by that block.
    int32_t niBegin = blockIdx.y * params.nPerBlock;
    // The last activation loaded by that block.
    int32_t niEnd = min(niBegin + params.nPerBlock, params.ni);

    GroupSumsOp op;
    GroupSums a, b;
    for (int i = niBegin; i < niEnd; i++)
    {
        a = GroupSums();
        b = GroupSums();
        for (int j = ci; j < ci + params.cPerBlock; j++)
        {
            float x = smem[i * params.cPerBlock + j];
            float y = smem[i * params.cPerBlock + j + 1];
            a.sum += x * y;
            a.sumSq += x * x + y * y;
        }
        b = op(a, b);
    }
    smem[i * params.cPerBlock] = b.sum;
    smem[i * params.cPerBlock + 1] = b.sumSq;
}
    
```

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>

```

#include "common/common.cuh"
#include "layerNormKernel.h"

template <typename T>
using kvp = cub::KeyValuePair<T, T>;

template <typename T>
struct mySum
{
    __host__ __device__ __forceinline__ kvp operator()(kvp const& a, kvp const& b) const
    {
        return kvp(a.key + b.key, a.value + b.value);
    }
};

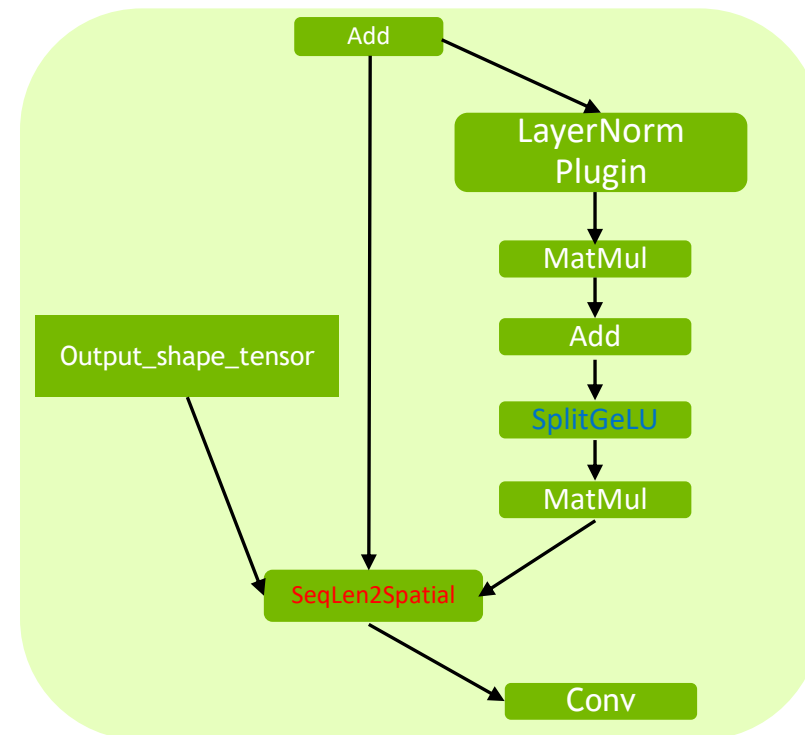
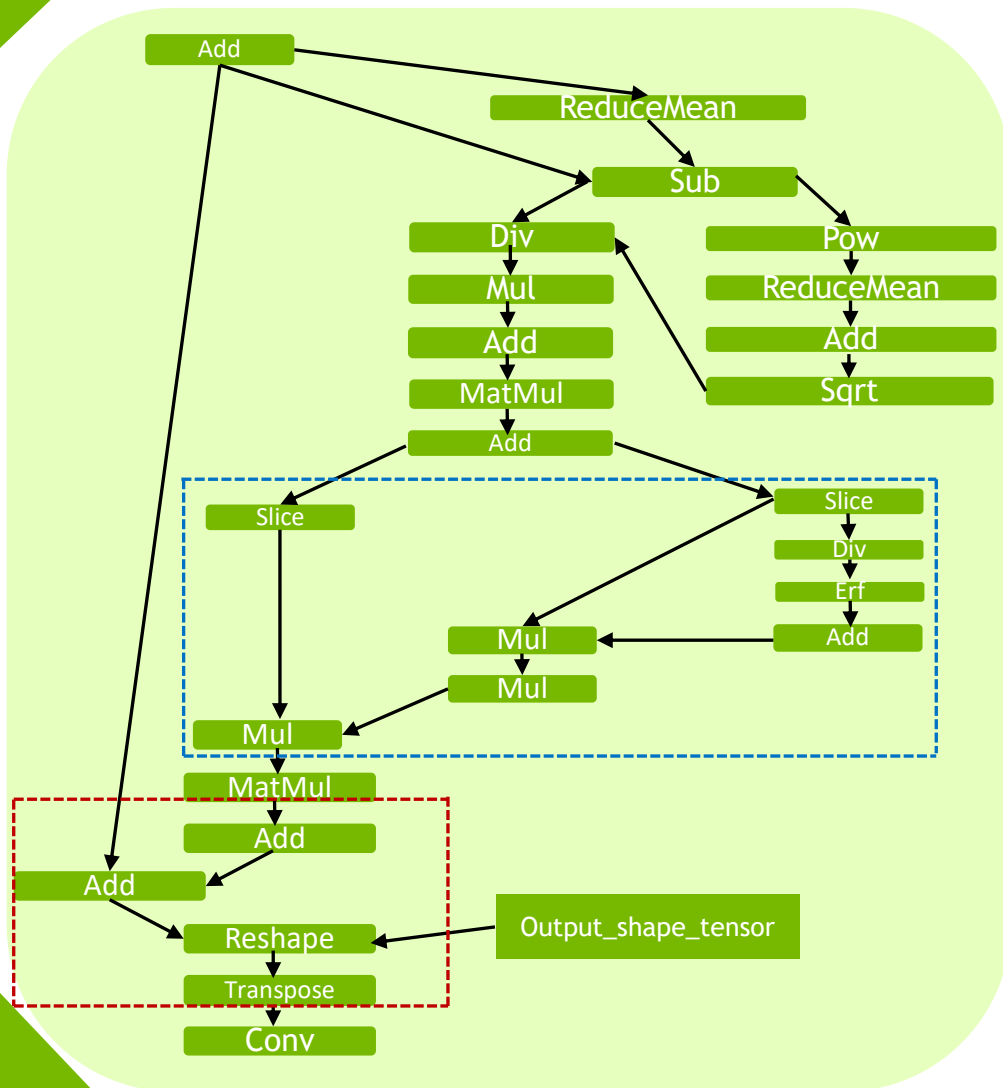
template <typename T, typename OP_T, int32_t TPB>
__global__ void layerNormKernel(
    int32_t const nHiddenDimension, T const* input, T const* gamma, T const* beta, T* output, float const epsilon)
{
    int32_t const index = blockIdx.x * nHiddenDimension + threadIdx.x;
    T const denominator = 1.f / (nHiddenDimension);
    OP_T val = 0;
    kvp<OP_T> threadData(0, 0);

    if (threadIdx.x < nHiddenDimension)
    {
        val = input[index] * denominator;
        OP_T tmp0 = val * static_cast<OP_T>(denominator);
        OP_T tmp1 = val * tmp0;
        threadData = mySum<OP_T>(threadData, kvp<OP_T>(tmp0, tmp1));
    }

    using MyReduce = cub::WarpReduce<kvp<OP_T>, TPB>;
    __shared__ typename MyReduce::TempStorage temp;
    __shared__ OP_T mu, rSigma;

    auto const sumKV = MyReduce(temp).Reduce(threadData, mySum<OP_T>());
    if (threadIdx.x == 0)
    {
        mu = sumKV.key;
        rSigma = rsqrt(sumKV.value - mu * mu + static_cast<OP_T>(epsilon));
    }
}
    
```

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

```

return b2vec;
}

// =====
// @param [in] p1: Vector1
// @param [in] p2: Vector2
// @param [in] p3: Vector3
// @param [in] p4: Vector4
// @param [in] p5: Vector5
// @param [in] p6: Vector6
// @param [in] p7: Vector7
// @param [in] p8: Vector8
// @param [in] p9: Vector9
// @param [in] p10: Vector10
// @param [in] p11: Vector11
// @param [in] p12: Vector12
// @param [in] p13: Vector13
// @param [in] p14: Vector14
// @param [in] p15: Vector15
// @param [in] p16: Vector16
// @param [in] p17: Vector17
// @param [in] p18: Vector18
// @param [in] p19: Vector19
// @param [in] p20: Vector20
// @param [in] p21: Vector21
// @param [in] p22: Vector22
// @param [in] p23: Vector23
// @param [in] p24: Vector24
// @param [in] p25: Vector25
// @param [in] p26: Vector26
// @param [in] p27: Vector27
// @param [in] p28: Vector28
// @param [in] p29: Vector29
// @param [in] p30: Vector30
// @param [in] p31: Vector31
// @param [in] p32: Vector32
// @param [in] p33: Vector33
// @param [in] p34: Vector34
// @param [in] p35: Vector35
// @param [in] p36: Vector36
// @param [in] p37: Vector37
// @param [in] p38: Vector38
// @param [in] p39: Vector39
// @param [in] p40: Vector40
// @param [in] p41: Vector41
// @param [in] p42: Vector42
// @param [in] p43: Vector43
// @param [in] p44: Vector44
// @param [in] p45: Vector45
// @param [in] p46: Vector46
// @param [in] p47: Vector47
// @param [in] p48: Vector48
// @param [in] p49: Vector49
// @param [in] p50: Vector50
// @param [in] p51: Vector51
// @param [in] p52: Vector52
// @param [in] p53: Vector53
// @param [in] p54: Vector54
// @param [in] p55: Vector55
// @param [in] p56: Vector56
// @param [in] p57: Vector57
// @param [in] p58: Vector58
// @param [in] p59: Vector59
// @param [in] p60: Vector60
// @param [in] p61: Vector61
// @param [in] p62: Vector62
// @param [in] p63: Vector63
// @param [in] p64: Vector64
// @param [in] p65: Vector65
// @param [in] p66: Vector66
// @param [in] p67: Vector67
// @param [in] p68: Vector68
// @param [in] p69: Vector69
// @param [in] p70: Vector70
// @param [in] p71: Vector71
// @param [in] p72: Vector72
// @param [in] p73: Vector73
// @param [in] p74: Vector74
// @param [in] p75: Vector75
// @param [in] p76: Vector76
// @param [in] p77: Vector77
// @param [in] p78: Vector78
// @param [in] p79: Vector79
// @param [in] p80: Vector80
// @param [in] p81: Vector81
// @param [in] p82: Vector82
// @param [in] p83: Vector83
// @param [in] p84: Vector84
// @param [in] p85: Vector85
// @param [in] p86: Vector86
// @param [in] p87: Vector87
// @param [in] p88: Vector88
// @param [in] p89: Vector89
// @param [in] p90: Vector90
// @param [in] p91: Vector91
// @param [in] p92: Vector92
// @param [in] p93: Vector93
// @param [in] p94: Vector94
// @param [in] p95: Vector95
// @param [in] p96: Vector96
// @param [in] p97: Vector97
// @param [in] p98: Vector98
// @param [in] p99: Vector99
// @param [in] p100: Vector100
// @param [in] p101: Vector101
// @param [in] p102: Vector102
// @param [in] p103: Vector103
// @param [in] p104: Vector104
// @param [in] p105: Vector105
// @param [in] p106: Vector106
// @param [in] p107: Vector107
// @param [in] p108: Vector108
// @param [in] p109: Vector109
// @param [in] p110: Vector110
// @param [in] p111: Vector111
// @param [in] p112: Vector112
// @param [in] p113: Vector113
// @param [in] p114: Vector114
// @param [in] p115: Vector115
// @param [in] p116: Vector116
// @param [in] p117: Vector117
// @param [in] p118: Vector118
// @param [in] p119: Vector119
// @param [in] p120: Vector120
// @param [in] p121: Vector121
// @param [in] p122: Vector122
// @param [in] p123: Vector123
// @param [in] p124: Vector124
// @param [in] p125: Vector125
// @param [in] p126: Vector126
// @param [in] p127: Vector127
// @param [in] p128: Vector128
// @param [in] p129: Vector129
// @param [in] p130: Vector130
// @param [in] p131: Vector131
// @param [in] p132: Vector132
// @param [in] p133: Vector133
// @param [in] p134: Vector134
// @param [in] p135: Vector135
// @param [in] p136: Vector136
// @param [in] p137: Vector137
// @param [in] p138: Vector138
// @param [in] p139: Vector139
// @param [in] p140: Vector140
// @param [in] p141: Vector141
// @param [in] p142: Vector142
// @param [in] p143: Vector143
// @param [in] p144: Vector144
// @param [in] p145: Vector145
// @param [in] p146: Vector146
// @param [in] p147: Vector147
// @param [in] p148: Vector148
// @param [in] p149: Vector149
// @param [in] p150: Vector150
// @param [in] p151: Vector151
// @param [in] p152: Vector152
// @param [in] p153: Vector153
// @param [in] p154: Vector154
// @param [in] p155: Vector155
// @param [in] p156: Vector156
// @param [in] p157: Vector157
// @param [in] p158: Vector158
// @param [in] p159: Vector159
// @param [in] p160: Vector160
// @param [in] p161: Vector161
// @param [in] p162: Vector162
// @param [in] p163: Vector163
// @param [in] p164: Vector164
// @param [in] p165: Vector165
// @param [in] p166: Vector166
// @param [in] p167: Vector167
// @param [in] p168: Vector168
// @param [in] p169: Vector169
// @param [in] p170: Vector170
// @param [in] p171: Vector171
// @param [in] p172: Vector172
// @param [in] p173: Vector173
// @param [in] p174: Vector174
// @param [in] p175: Vector175
// @param [in] p176: Vector176
// @param [in] p177: Vector177
// @param [in] p178: Vector178
// @param [in] p179: Vector179
// @param [in] p180: Vector180
// @param [in] p181: Vector181
// @param [in] p182: Vector182
// @param [in] p183: Vector183
// @param [in] p184: Vector184
// @param [in] p185: Vector185
// @param [in] p186: Vector186
// @param [in] p187: Vector187
// @param [in] p188: Vector188
// @param [in] p189: Vector189
// @param [in] p190: Vector190
// @param [in] p191: Vector191
// @param [in] p192: Vector192
// @param [in] p193: Vector193
// @param [in] p194: Vector194
// @param [in] p195: Vector195
// @param [in] p196: Vector196
// @param [in] p197: Vector197
// @param [in] p198: Vector198
// @param [in] p199: Vector199
// @param [in] p200: Vector200
// @param [in] p201: Vector201
// @param [in] p202: Vector202
// @param [in] p203: Vector203
// @param [in] p204: Vector204
// @param [in] p205: Vector205
// @param [in] p206: Vector206
// @param [in] p207: Vector207
// @param [in] p208: Vector208
// @param [in] p209: Vector209
// @param [in] p210: Vector210
// @param [in] p211: Vector211
// @param [in] p212: Vector212
// @param [in] p213: Vector213
// @param [in] p214: Vector214
// @param [in] p215: Vector215
// @param [in] p216: Vector216
// @param [in] p217: Vector217
// @param [in] p218: Vector218
// @param [in] p219: Vector219
// @param [in] p220: Vector220
// @param [in] p221: Vector221
// @param [in] p222: Vector222
// @param [in] p223: Vector223
// @param [in] p224: Vector224
// @param [in] p225: Vector225
// @param [in] p226: Vector226
// @param [in] p227: Vector227
// @param [in] p228: Vector228
// @param [in] p229: Vector229
// @param [in] p230: Vector230
// @param [in] p231: Vector231
// @param [in] p232: Vector232
// @param [in] p233: Vector233
// @param [in] p234: Vector234
// @param [in] p235: Vector235
// @param [in] p236: Vector236
// @param [in] p237: Vector237
// @param [in] p238: Vector238
// @param [in] p239: Vector239
// @param [in] p240: Vector240
// @param [in] p241: Vector241
// @param [in] p242: Vector242
// @param [in] p243: Vector243
// @param [in] p244: Vector244
// @param [in] p245: Vector245
// @param [in] p246: Vector246
// @param [in] p247: Vector247
// @param [in] p248: Vector248
// @param [in] p249: Vector249
// @param [in] p250: Vector250
// @param [in] p251: Vector251
// @param [in] p252: Vector252
// @param [in] p253: Vector253
// @param [in] p254: Vector254
// @param [in] p255: Vector255
// @param [in] p256: Vector256
// @param [in] p257: Vector257
// @param [in] p258: Vector258
// @param [in] p259: Vector259
// @param [in] p260: Vector260
// @param [in] p261: Vector261
// @param [in] p262: Vector262
// @param [in] p263: Vector263
// @param [in] p264: Vector264
// @param [in] p265: Vector265
// @param [in] p266: Vector266
// @param [in] p267: Vector267
// @param [in] p268: Vector268
// @param [in] p269
```

Models.py

Provide model specific onnx graph optimization strategies

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

[illegible]

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
```

```

# Create the first DataFrame
df = pd.DataFrame({
    'id': 1,
    'name': 'John Doe',
    'age': 30,
    'gender': 'Male',
    'height': 1.8,
    'weight': 75,
    'blood_pressure': '120/80',
    'heart_rate': 70,
    'cholesterol': '150',
    'sugar': '100'
})

# Create the second DataFrame
df2 = pd.DataFrame({
    'id': 2,
    'name': 'Jane Smith',
    'age': 25,
    'gender': 'Female',
    'height': 1.6,
    'weight': 60,
    'blood_pressure': '110/70',
    'heart_rate': 65,
    'cholesterol': '140',
    'sugar': '90'
})

# Concatenate the two DataFrames
df_combined = pd.concat([df, df2])

# Print the combined DataFrame
print(df_combined)

```

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

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

Optimization Level

VAE

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

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