IML 2022 Talk: Optimized Deep Learning Inference on High Level Trigger at the LHC: Computing time and Resource assessment

Nadezda Chernyavskaya[2], **Syed Hasan**[1,4], *Pratik Jawahar*[5], *Maurizio Pierini*[2], *Kinga Anna Wozniak*[2,3]

[1] Scuola Normale Superiore Pisa, [2] CERN , [3] University of Vienna , [4] ETH Zurich, [5] Worcester Polytechnic Institute

Table of Contents

Motivation for HLT anomaly detection algorithms and optimized fast inference

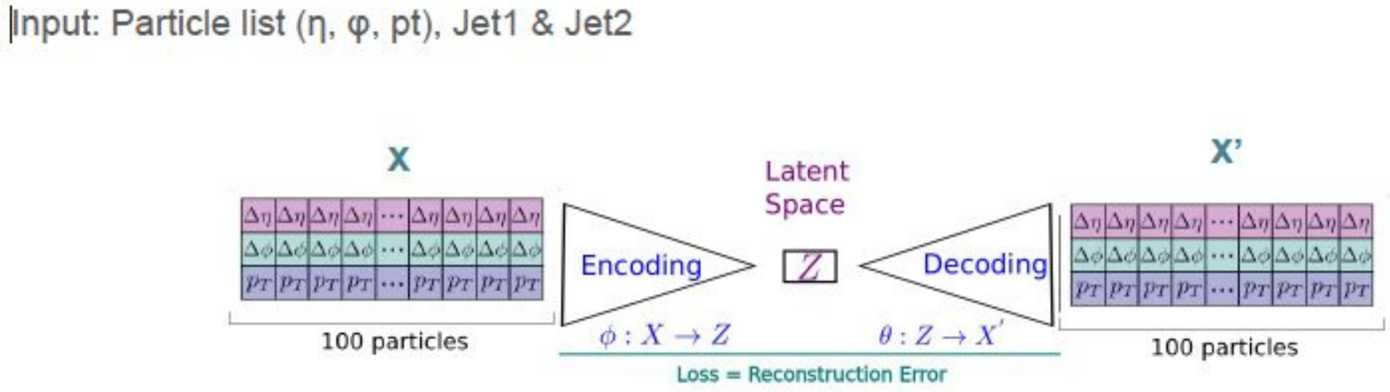
Data Flow at the Large Hadron Collider, CERN

- Wide application of variational autoencoders (VAE) and Graph-based VAE models
  - Beyond the standard model events anomaly detection
  - New Physics searches for the LHC at L1, Jets-based AE for anomaly detection
  - New Autoencoders design for High-level Trigger (HLT) online trigger

- Inference studies with HLT anomaly detection algorithms on CPU and GPU hardware **guide decisions** on GPU farm requirements for LHC Run 3 and beyond

Machine learning model architectures for anomaly detection for the High-level trigger at the LHC

Dataset Representation: Jet-level

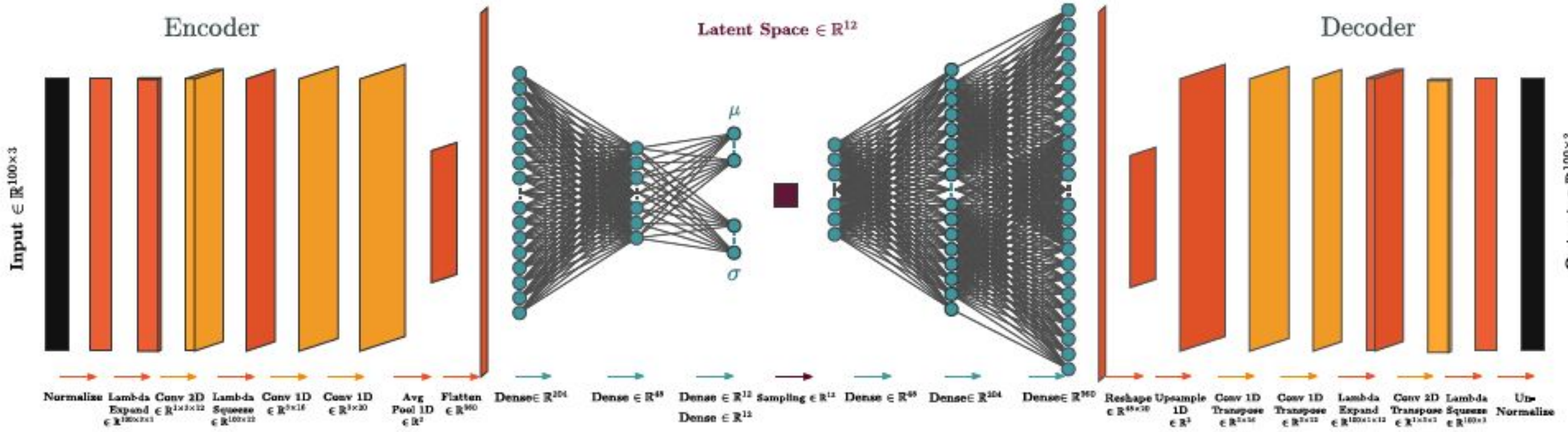


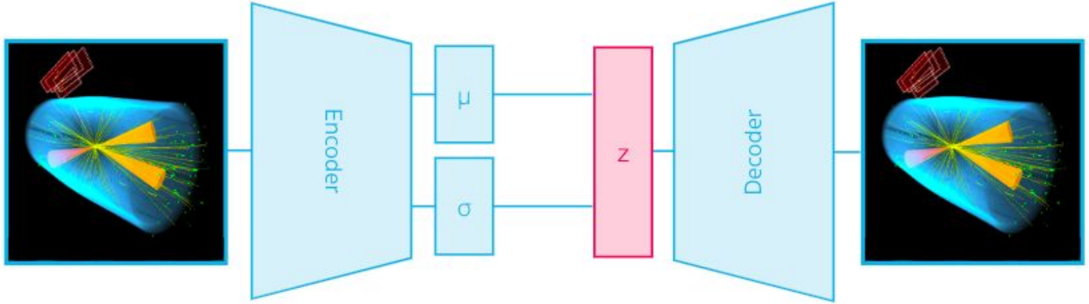
- Jet-level VAE model (Tensorflow)
    - Input Shape: (# of jets, # of constituents, # input features equal particle momentum in cylindrical coordinates)
    - Example: (100000, 100, 3)

- Unsupervised Learning method
    - Can be trained directly on data
    - Used Monte Carlo simulations for this use case-study

Convolutional VAE (Conv-VAE) architecture: Jet-level

Pictorial representation of the architecture of the variational autoencoder (VAE) used for jet anomaly detection





Conv-VAE architecture (Jet-level): Training configuration

- Machine learning libraries: Keras and Tensorflow 2.4.1
- Optimizer: Adam
- Initial learning rate: 0.001, beta = 0.0005
- Latent dimensions: 12 (also tried other sizes: 6, 8)
- learning rate decay and early stopping procedure enforced
- Loss function = Reconstruction loss + Kullback Leibler (KL) divergence (loss on latent space)

$$\text{Loss} = L_{\text{RECO}} + \beta \cdot D_{KL}$$

Conv-VAE (Jet-level): Model Summary example (with latent space dimension: 8)

Model: "encoder"

Layer (type)	Output Shape	Param #
encoder_input (InputLayer)	[(None, 100, 3)]	0
Std_Normalize (StdNormalizat	(None, 100, 3)	0
lambda (Lambda)	(None, 100, 3, 1)	0
conv2d (Conv2D)	(None, 98, 1, 16)	160
lambda_1 (Lambda)	(None, 98, 16)	0
conv1d (Conv1D)	(None, 96, 20)	980
conv1d_1 (Conv1D)	(None, 94, 24)	1464
average_pooling1d (AveragePo	(None, 47, 24)	0
flatten (Flatten)	(None, 1128)	0
dense (Dense)	(None, 136)	153544
dense_1 (Dense)	(None, 32)	4384
z (Dense)	(None, 8)	264

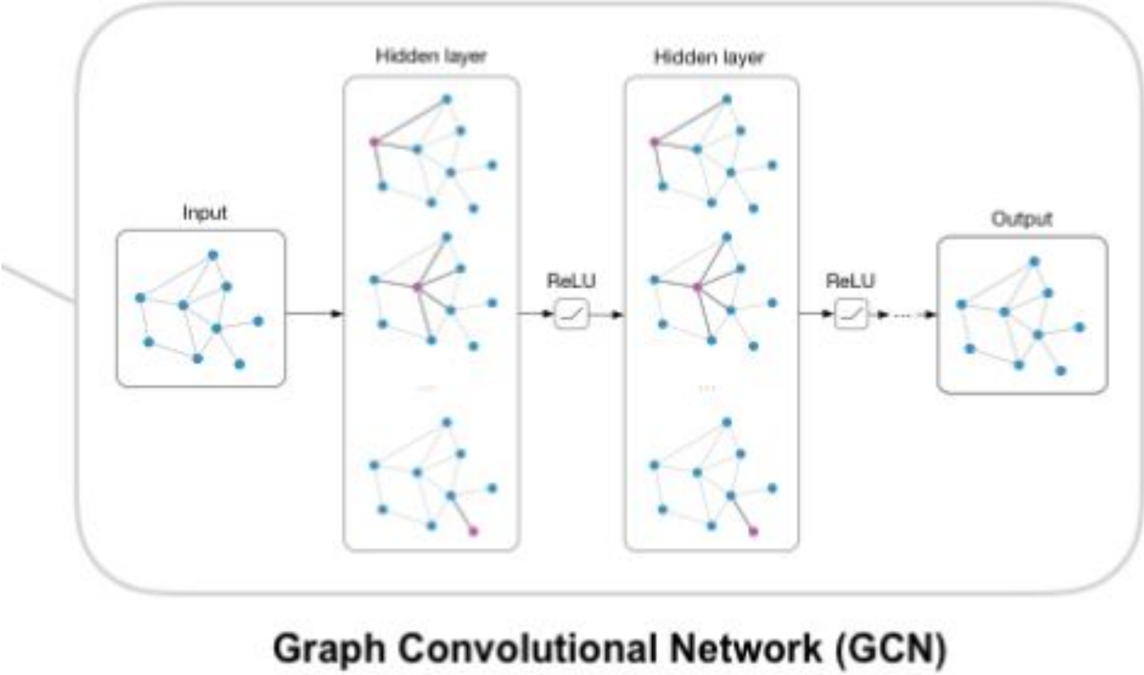
Total params: 160,796
Trainable params: 160,796
Non-trainable params: 0

Model: "decoder"

Layer (type)	Output Shape	Param #
z (InputLayer)	[(None, 8)]	0
dense_2 (Dense)	(None, 32)	288
dense_3 (Dense)	(None, 136)	4488
dense_4 (Dense)	(None, 1128)	154536
reshape (Reshape)	(None, 47, 24)	0
up_sampling1d (UpSampling1D)	(None, 94, 24)	0
conv1d_transpose (Conv1DTran	(None, 96, 20)	1460
conv1d_transpose_1 (Conv1DTr	(None, 98, 16)	976
lambda_6 (Lambda)	(None, 98, 1, 16)	0
conv_2d_transpose (Conv2DTra	(None, 100, 3, 1)	145
lambda_7 (Lambda)	(None, 100, 3)	0
Un_Normalize (StdUnnormaliza	(None, 100, 3)	0

Total params: 161,893
Trainable params: 161,893
Non-trainable params: 0

Convolutional variational Graph Autoencoder (Graph Conv-VAE)



Pictorial representation of a Multi-layer Graph Convolutional Network (GCN-VAE) with first-order filters

Convolutional variational Graph Autoencoder (Graph Conv-VAE)

Definitions:

- X : N x D feature matrix
  - N: number of jet constituents, D: number of input features (pT, eta phi)
- A: adjacency matrix
- Z: node-level output, N×F feature matrix, where F is the number of output features per node

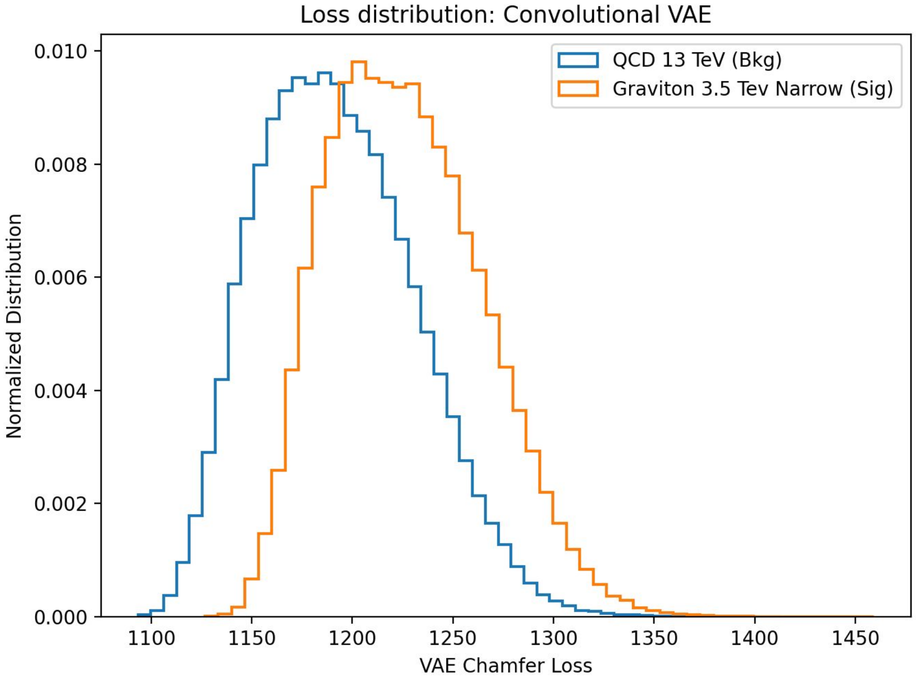
→ *Graph Conv-VAE tries to reconstruct nodes features at the output*

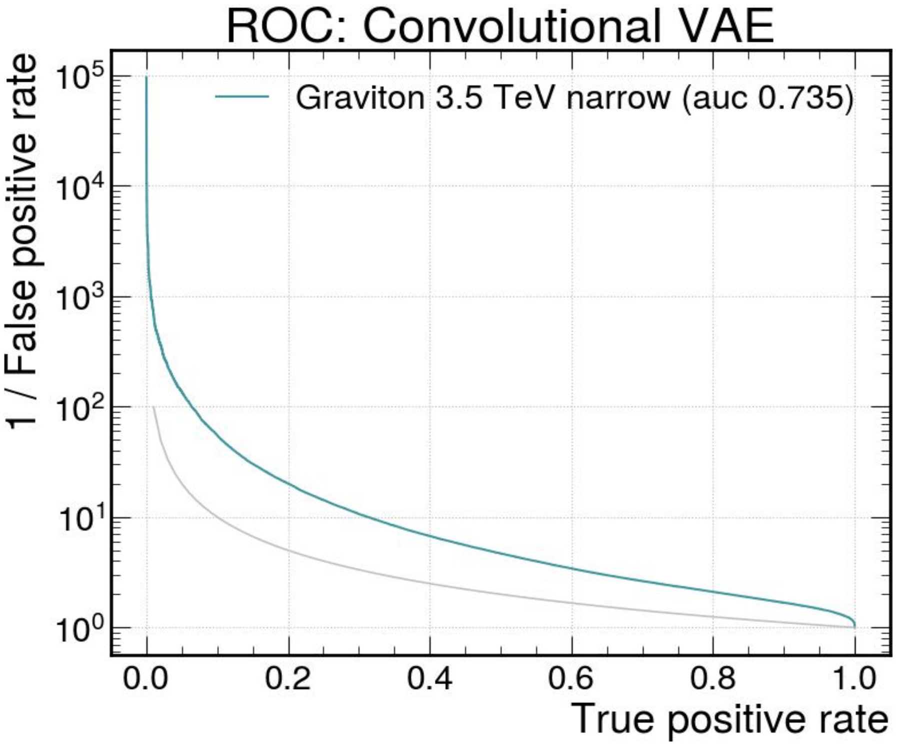
Training configuration:

- Machine learning library: Tensorflow and Keras
- Activation function: tf.nn.tanh; latent dimension:8; beta_kl: 10; kl_warmup_time: 5
- Optimizer: Adam; learning rate: 0.0001
- Loss function:

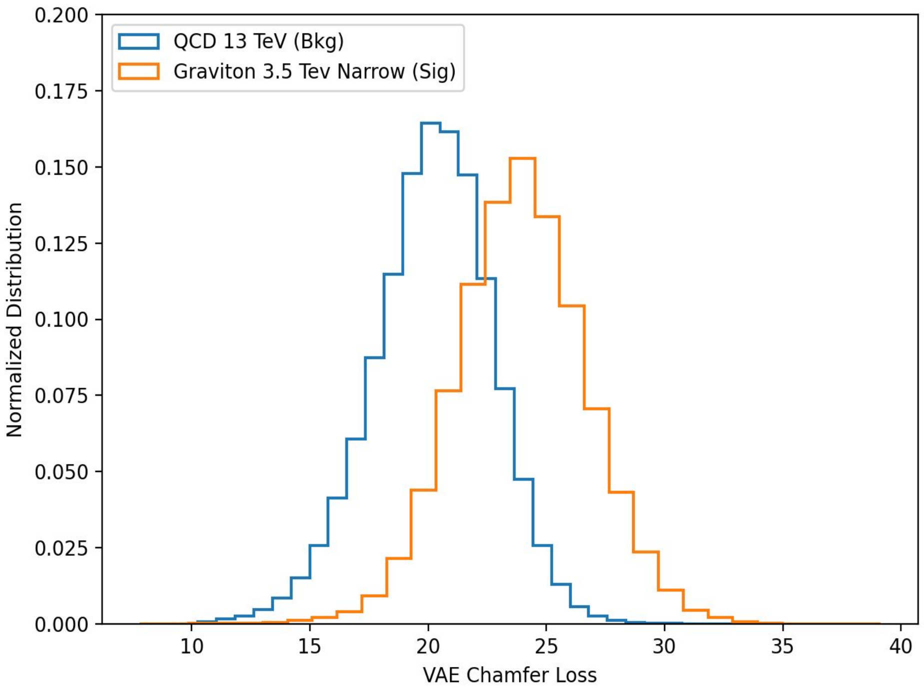
$$\text{Loss} = L_{\text{RECO}} + \beta \cdot D_{KL}$$

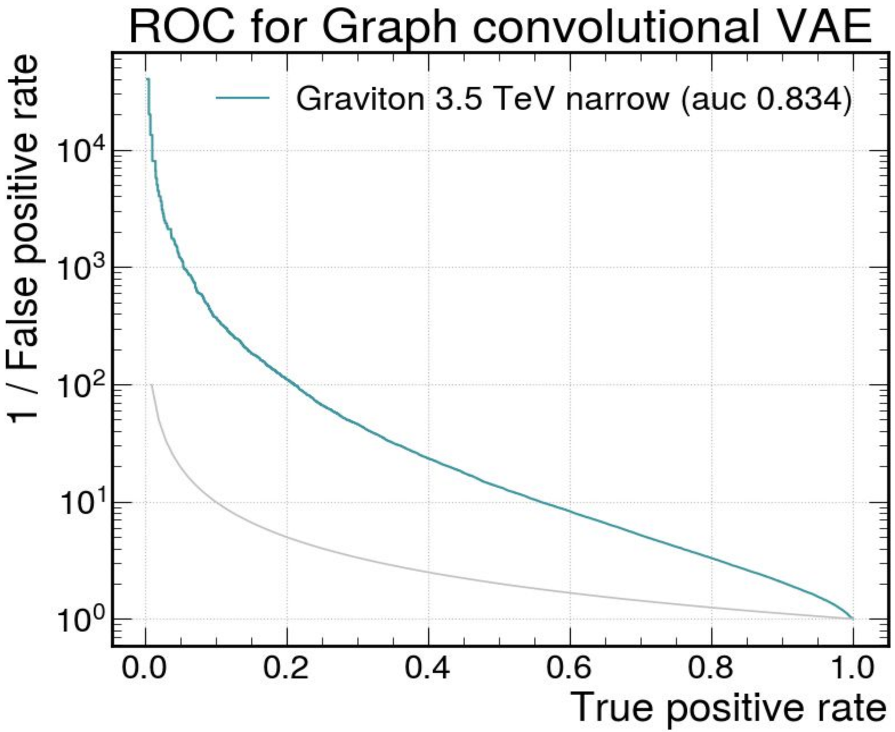
Conv-VAE (Jet-level) Performance: Chamfer loss





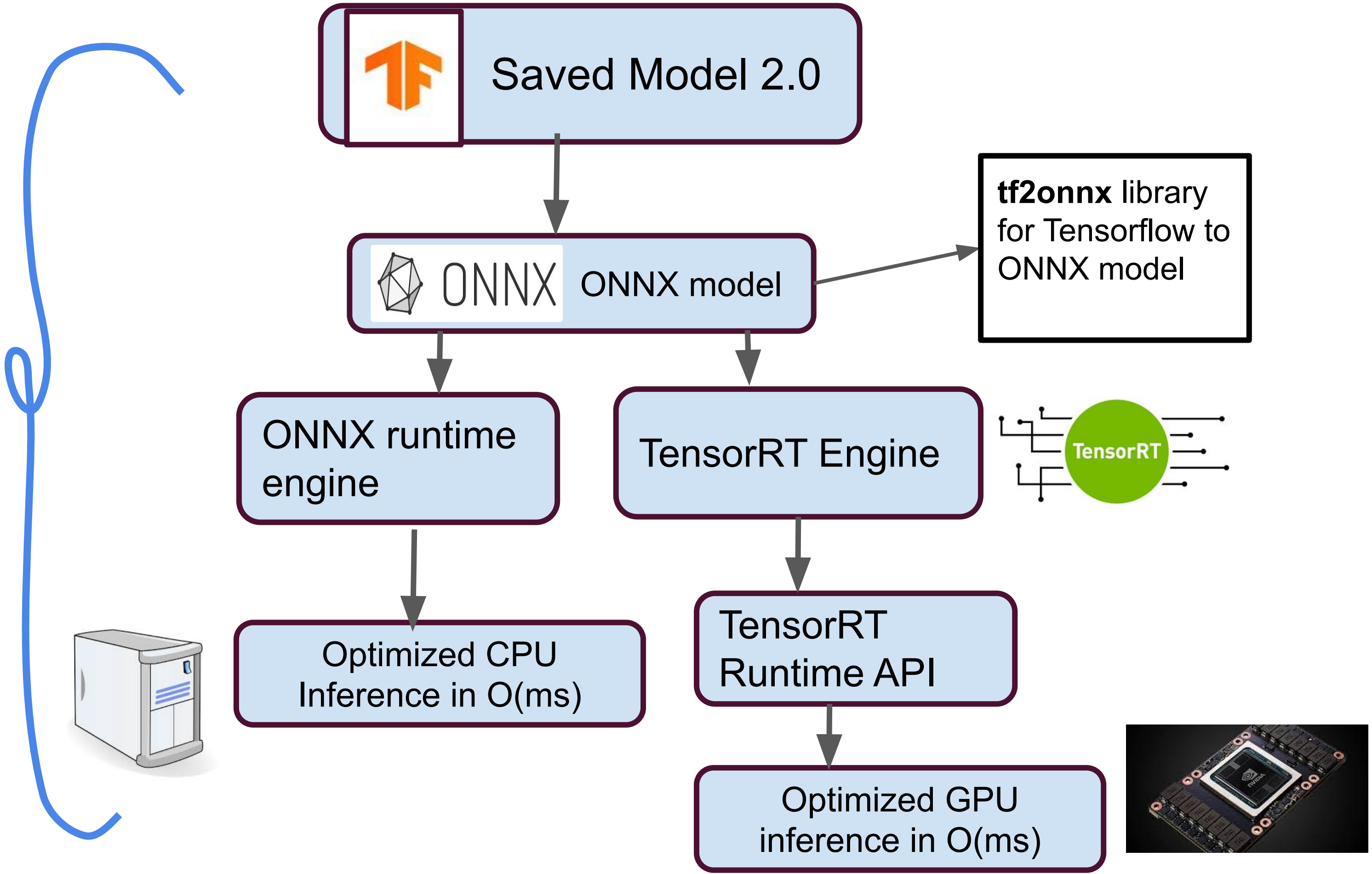
Graph Conv-VAE (Jet-level) Performance: Chamfer loss



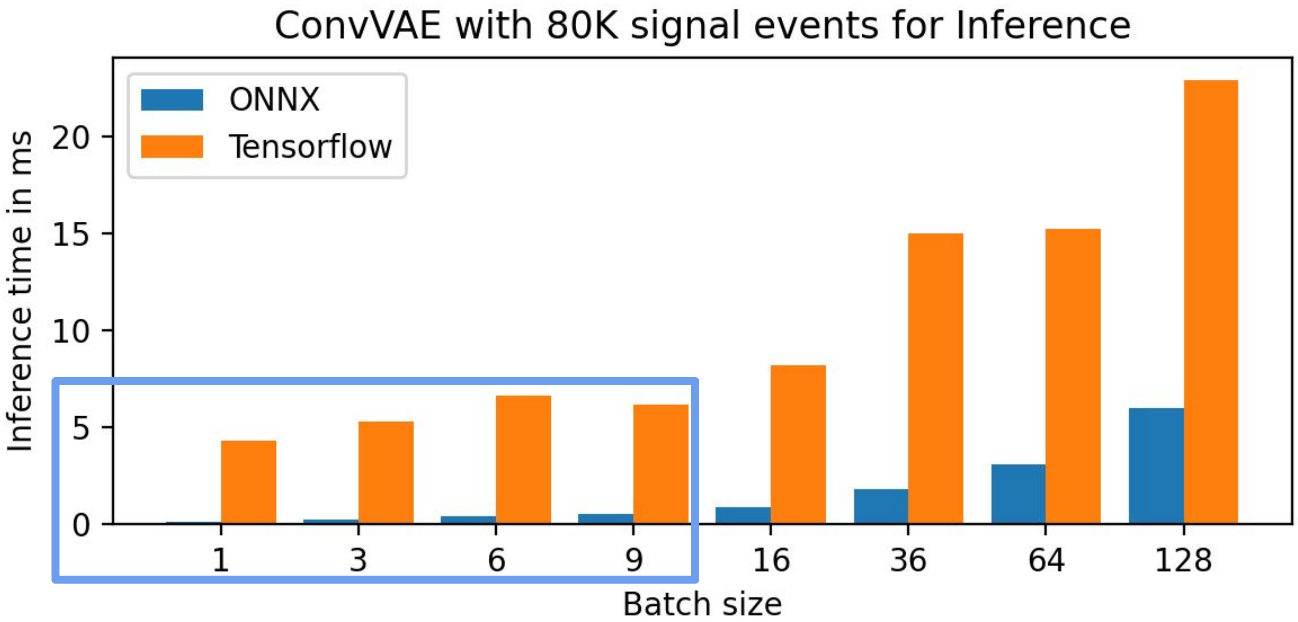


Results: Optimized CPU based model inference - Inference time (latency) and resource (memory usage)

Optimized CPU and GPU based model inference - Workflow

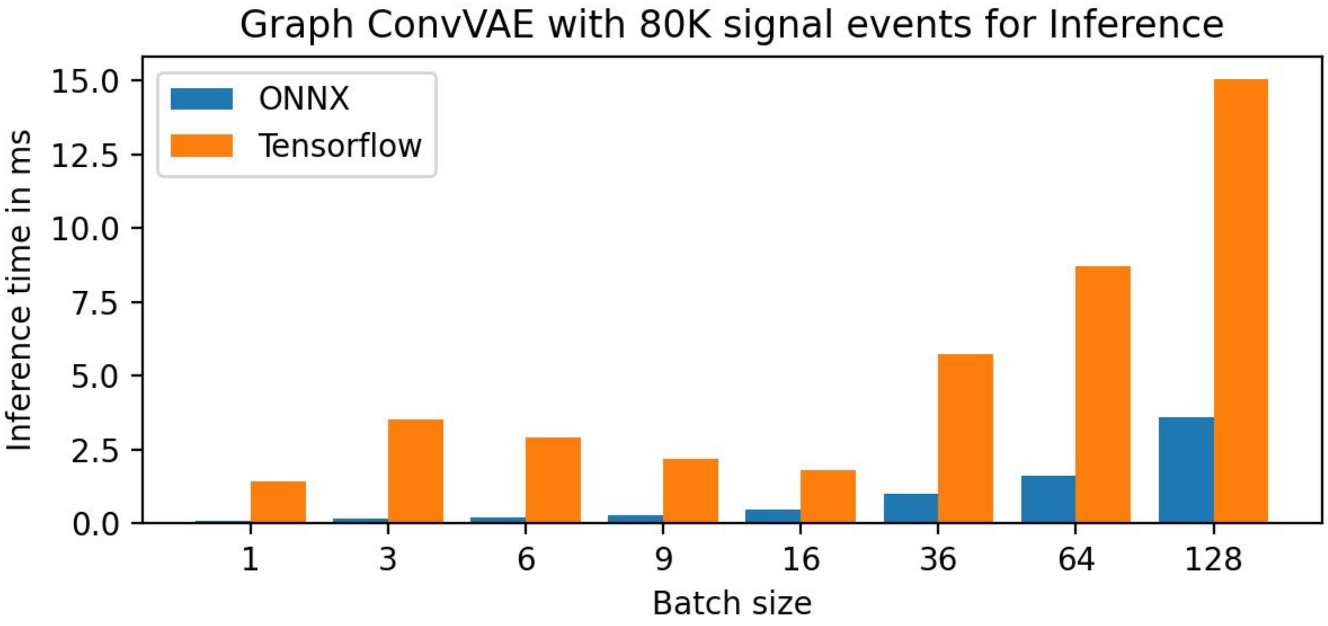


Results: Conv-VAE model inference latency on CPU



- At LHC HLT trigger, **typically 9 jets** are present in a single event for the inference process to detect anomalies
- Maximal gain at batch size: 1; with ONNX Runtime, we get the inference time within 5 ms for as many as 64 inferences

Results: Graph Conv-VAE model inference latency on CPU



With ONNX Runtime, we get the inference time within 2.5 ms for as many as 64 inferences

CPU memory profiling for inference resource consumption: Conv-VAE

- Tool used: Bloomberg's memray (memory profiler for python)

- We observe the memory footprint for the inference execution run vs. varying batch size
  - Results show exact memory utilization (very little change) for both TF and ONNX environments

- Measure the memory footprint of the inference execution call via both native TF and ONNX
  - We consider 150K signal events as this is the minimum number of events required in order to measure the memory footprint
  - **Result:** Memory consumption with TF run: 282.35 MB; ONNX Runtime: 264.024 MB
  - Memory footprint of TF and ONNX is almost same

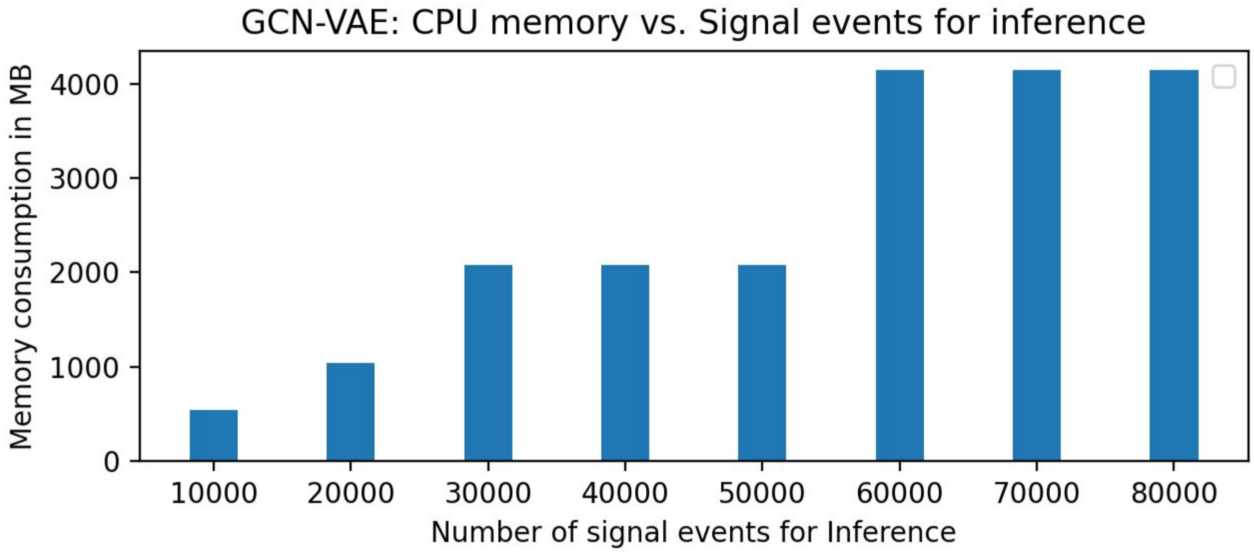
CPU memory profiling for inference resource consumption: Graph Conv-VAE

- We observe the memory footprint to stay same when we vary batch sizes as with Conv-VAE

- Measure the memory footprint of the inference execution call via both native TF and ONNX.
  - We consider 80K signal events for inference
  - **Result:** Memory consumption with TF run: 4151 MB; ONNX Runtime: 4141 MB
  - Again, we observe memory footprint of TF and ONNX is almost same

CPU memory profiling for inference resource consumption: Graph Conv-VAE

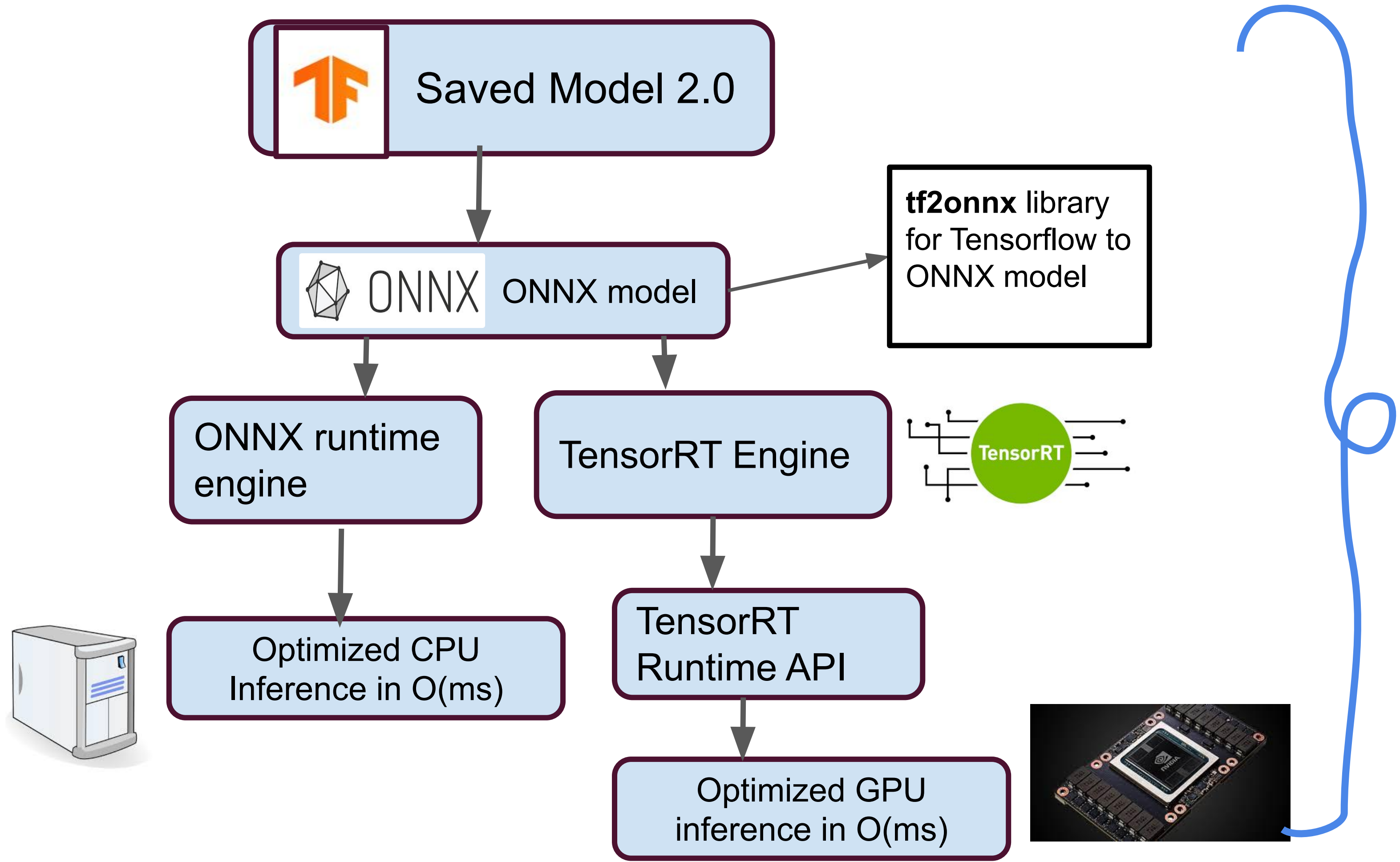
- Memory footprint vs. number of signal events for inference



The memory footprint gradually increases and then stays flat when we vary signal events

Optimized GPU model inference with NVIDIA TensorRT

Optimized GPU based model inference - Workflow



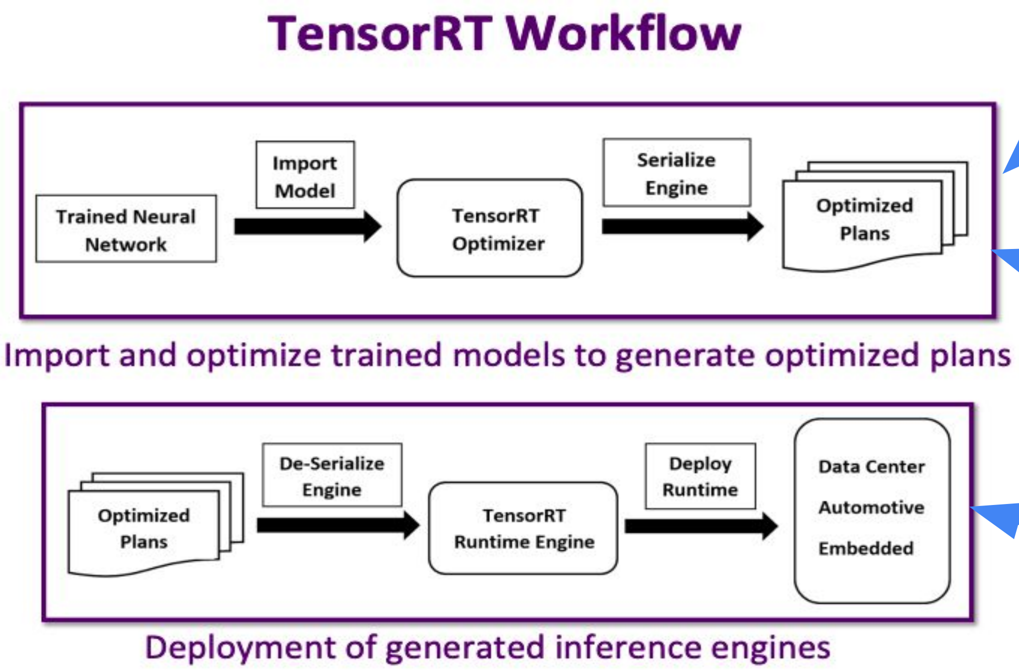
Software libraries used for TensorRT GPU model inference

software	version
NVIDIA TensorRT	7.2.3.4
CUDA	11.2
CUDNN	8.1.1.33
Tensorflow	2.5.0
Onnxruntime	1.8.0
Pycuda	2021.1
Keras	2.4.3
numpy	1.21.1
Pandas	1.2.2
Protobuf	2.5.0
Nsight Systems	2021.2

Libraries for TensorRT GPU inference

*Used for GPU CUDA **memory profiling** for TensorRT inference*

Optimized GPU model inference: ONNX to TensorRT (Steps)



Tensorflow SavedModel File to ONNX model conversion and ONNX model check (Step 1)

ONNX model to TensorRT (TRT) engine creation (GPU-specific) (Step 2)

Computing the inference using TRT runtime (GPU-specific context) using TRT engine plan file (Step 3)

Optimized ONNX-TensorRT GPU inference - Memory consumption of Tesla V100 and Tesla T4 GPUs

Tool: NVIDIA Nsight systems for memory profiling and get memory statistics

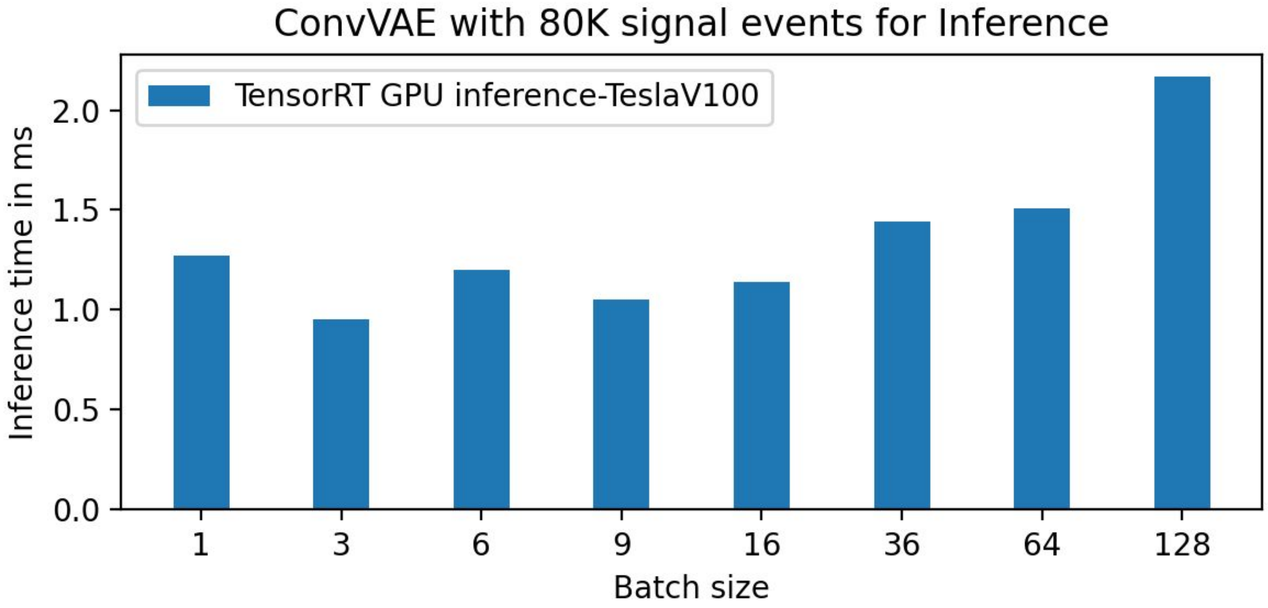
```
nsys profile --stats=true -t cuda python3 inference_script args
```

```
CUDA Memory Operation Statistics (by time):

 Time(%)  Total Time (ns)  Operations  Average   Minimum  Maximum  StdDev       Operation
 -------  ---------------  ----------  --------  -------  -------  -------  ------------------
    54.7        9,522,365         698  13,642.4    1,215  182,847  8,476.2  [CUDA memcpy HtoD]
    44.8        7,789,035         626  12,442.5    2,560   24,032    709.5  [CUDA memcpy DtoH]
     0.3           49,664           8   6,208.0    4,096   10,912  2,862.7  [CUDA memcpy DtoD]
     0.2           38,975           9   4,330.6    1,920    5,184    950.1  [CUDA memset]


CUDA Memory Operation Statistics (by size in KiB):

   Total     Operations  Average  Minimum   Maximum   StdDev       Operation
 ----------  ----------  -------  -------  ---------  -------  ------------------
 98,647.859         698  141.329    0.004  1,634.480   85.867  [CUDA memcpy HtoD]
 93,750.008         626  149.760    0.008    150.000    5.995  [CUDA memcpy DtoH]
     61.004           9    6.778    1.004      7.500    2.165  [CUDA memset]
  1,555.000           8  194.375    1.000    769.000  354.669  [CUDA memcpy DtoD]

Report file moved to "/home/sahasan/dijetanomaly/vande/TRT_engines/report1.qdrep"
Report file moved to "/home/sahasan/dijetanomaly/vande/TRT_engines/report1.sqlite"
```

Results: Optimized GPU model inference with TensorRT - Inference time (latency) and resource (memory usage)

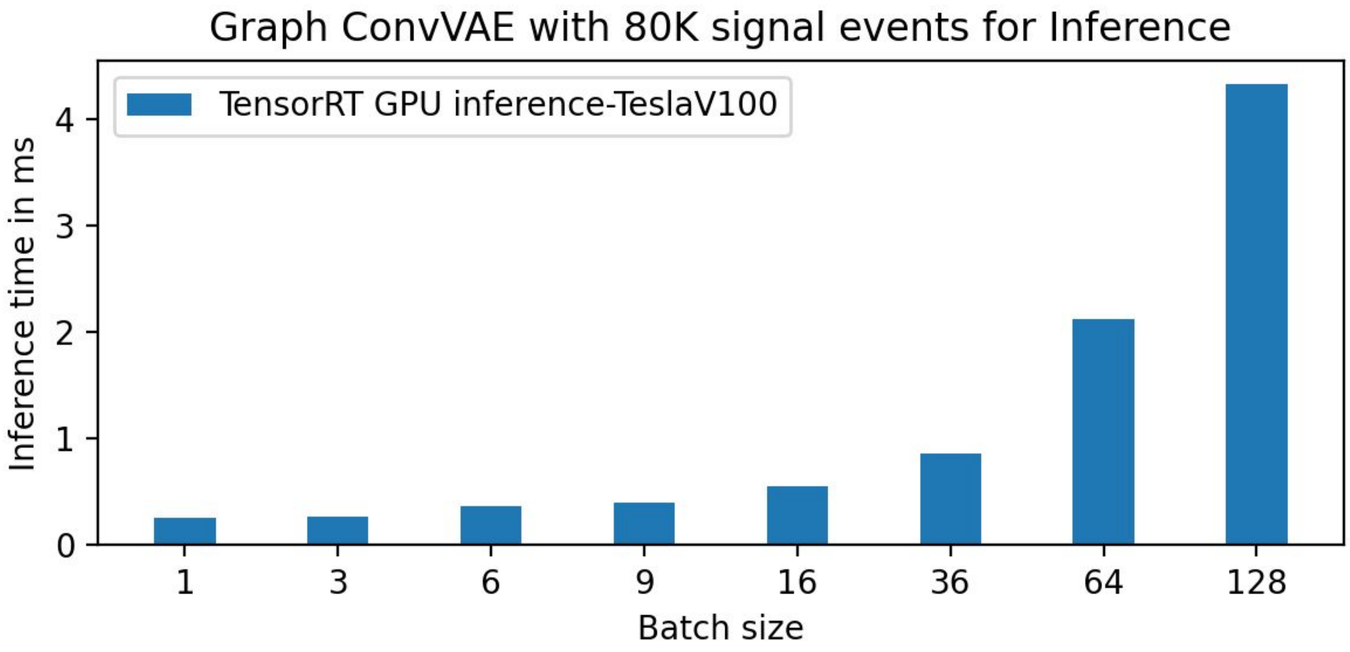
TensorRT GPU model inference latency (Conv-VAE) - Tesla V100

- **Precision: FP32**



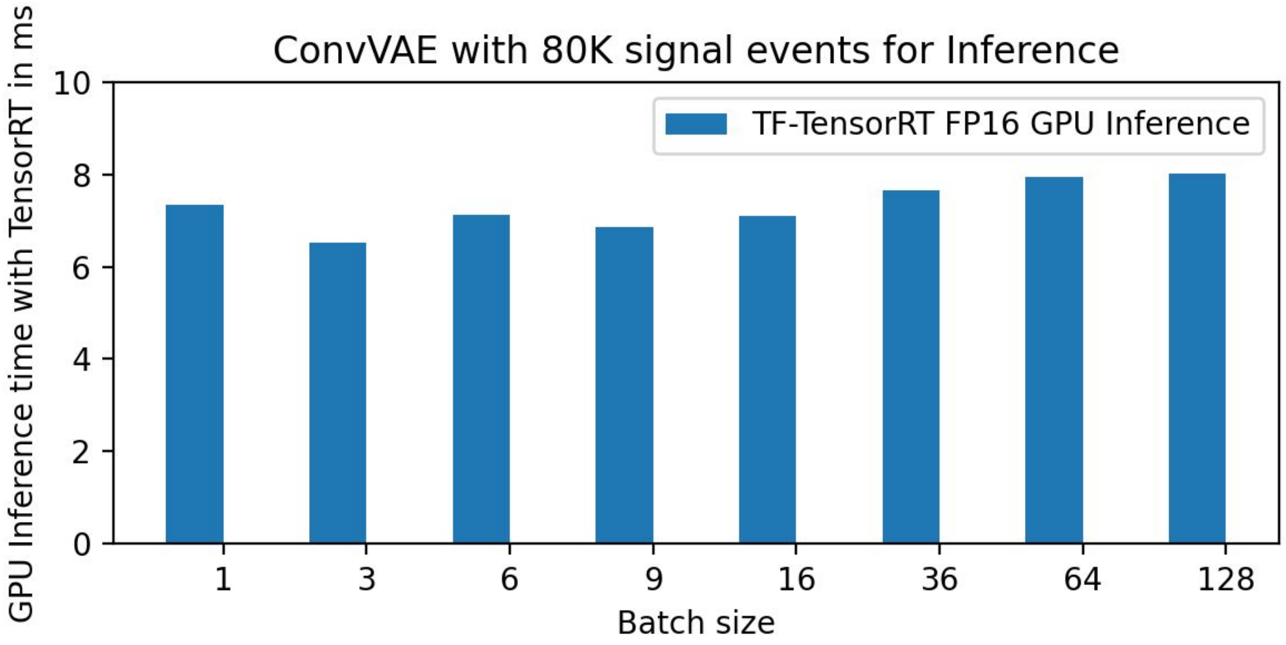
The inference time stays relatively flat with varying batch size and almost doubles at the highest batch size

TensorRT GPU model inference latency (GCN-VAE) - Tesla V100



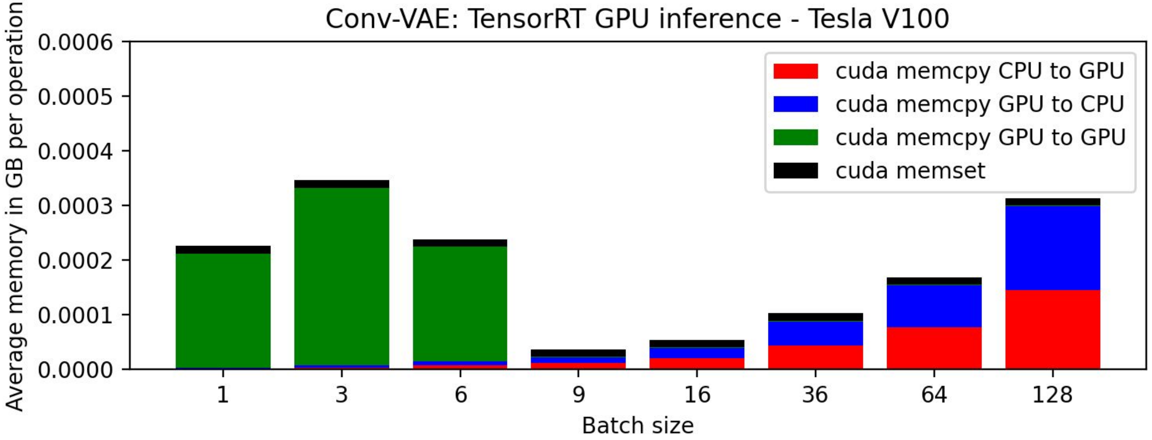
The inference time gradually increases and then linearly at higher batch sizes

Tensorflow to TensorRT model inference (without ONNX)

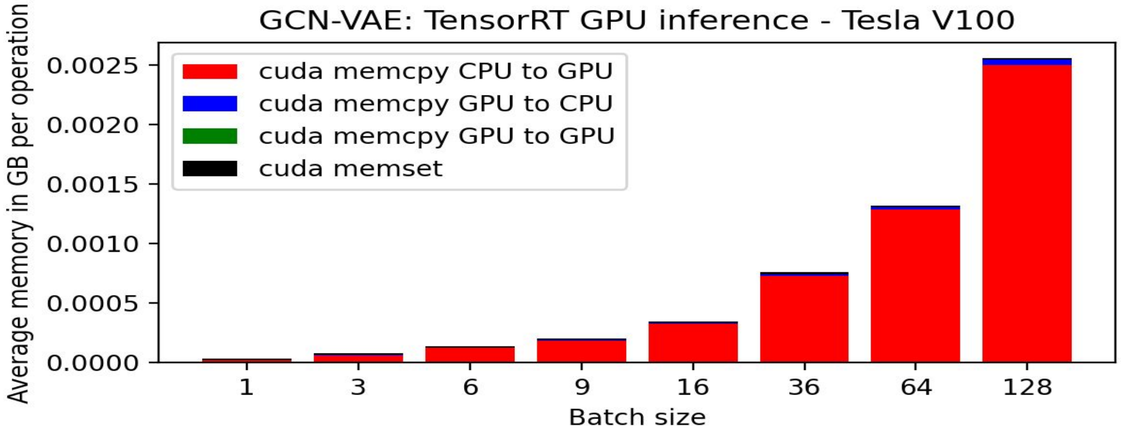


*The trend is the inference time stays relatively flat for different batch sizes **but much higher inference time values than the optimized ONNX + TensorRT based GPU inference***

Optimized ONNX-TensorRT GPU inference for Conv-VAE vs Graph Conv-VAE: Average memory (Tesla V100)

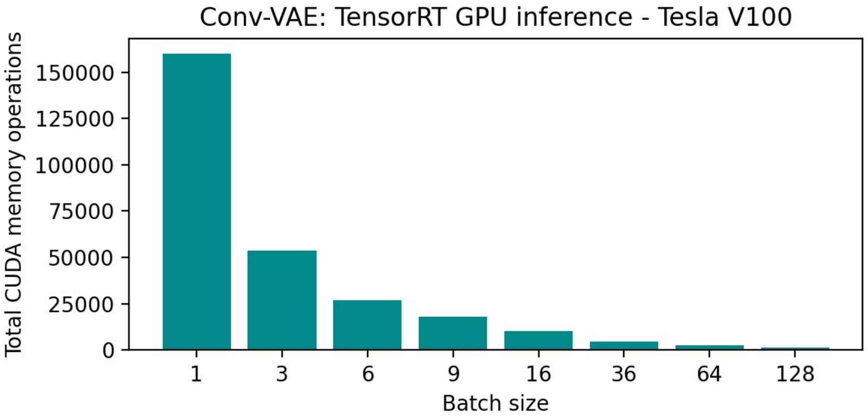


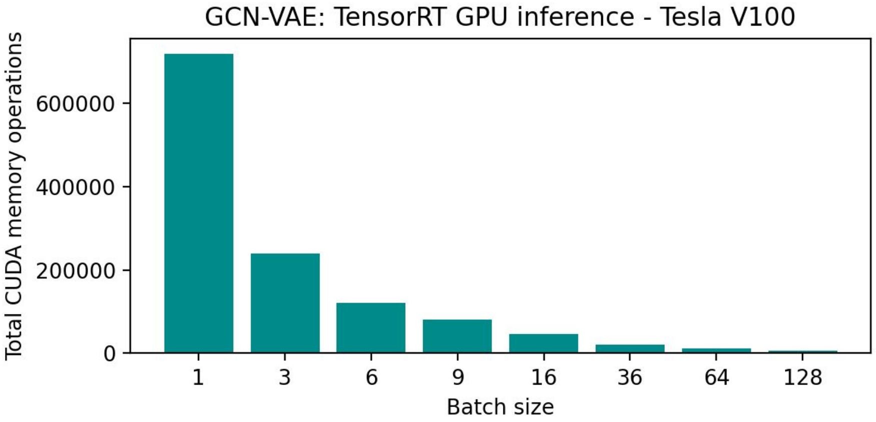
The memory copy operations (cpu to gpu) and (gpu to cpu) dominates at higher batches



The memory copy operations (cpu to gpu) dominate at all batch sizes

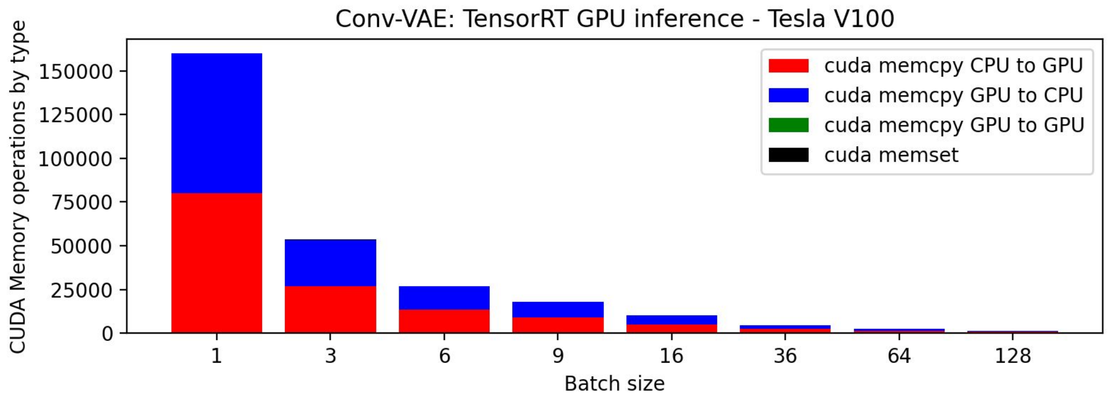
Optimized ONNX-TensorRT GPU inference for Conv-VAE vs. Graph Conv-VAE : Total CUDA memory operations (Tesla V100)

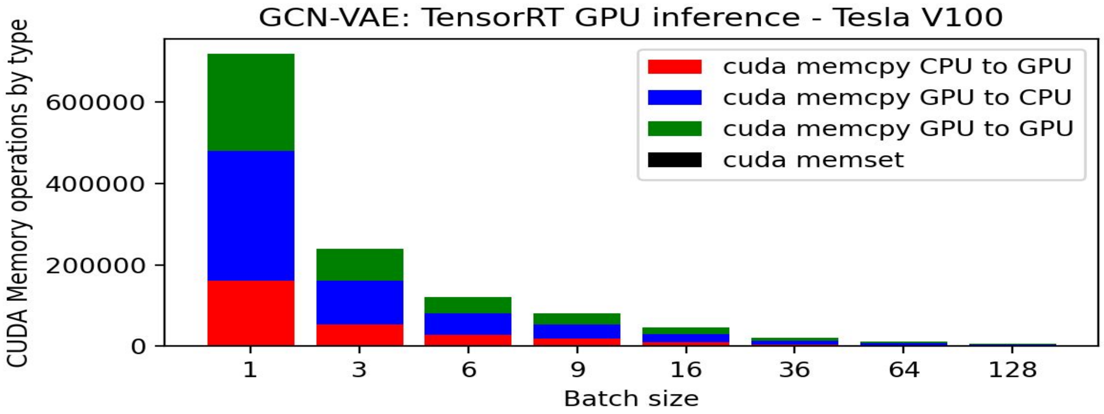




CUDA memory operations decline drastically according to the increase in batch size

Optimized ONNX-TensorRT GPU inference for Conv-VAE and Graph Conv-VAE: Memory operations by type (Tesla V100)





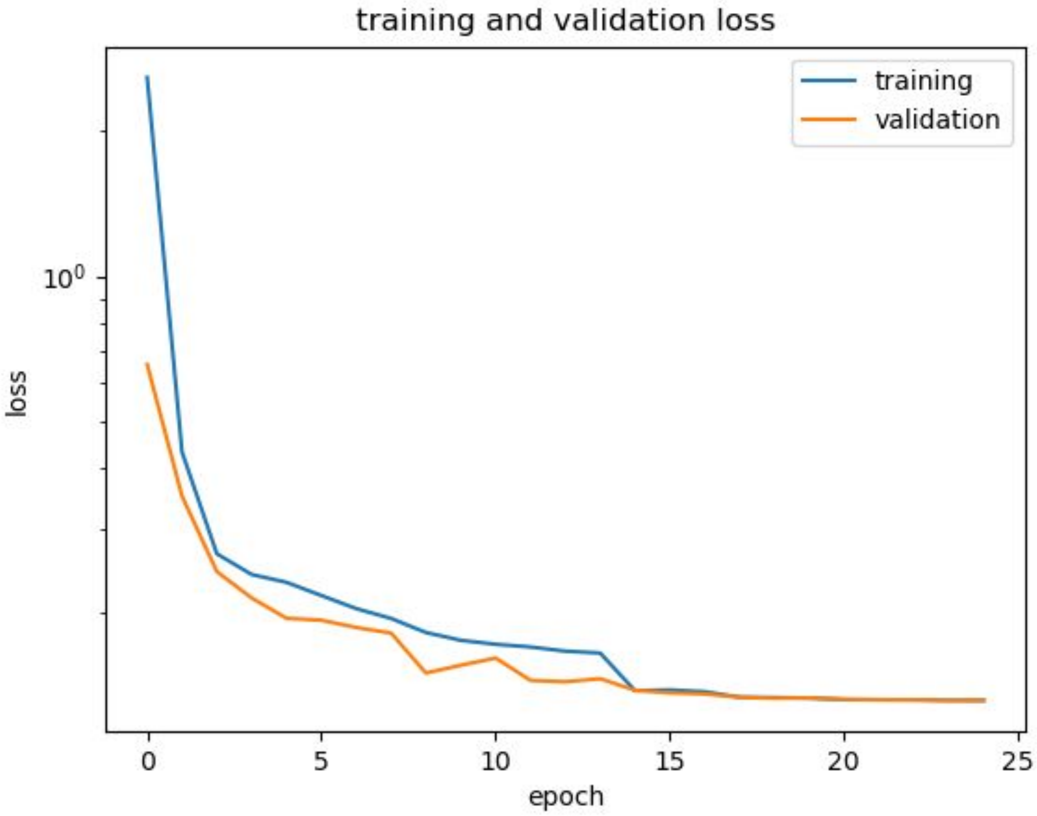
The cuda memory copy (cpu to gpu) and (gpu to cpu) operations dominate over different batch sizes

Conclusion

- We demonstrate significant inference time savings with ONNX and TensorRT over native Tensorflow 2 (keras) based inference for both CPU and GPU
- We observe the O(msec) latency with different batch sizes for CPU- and GPU-based model inference, well within the O(100 msec) allocated to the processing of one event
- We also perform CPU and GPU memory profiling of model inference to assess resource consumption (memory usage) of our DL algorithms
- Our conclusions demonstrate that DL algorithms optimized with available libraries are perfectly compatible with the operation constraints of a typical HLT environment
- This study confirms that there is no technical challenge in deploying DL algorithms in the ATLAS and CMS HLT farms in the near future

BACKUPS

Conv-VAE model (Jet-level): Loss curve



TESLA V100 during the ONNX-TensorRT GPU model Inference Run

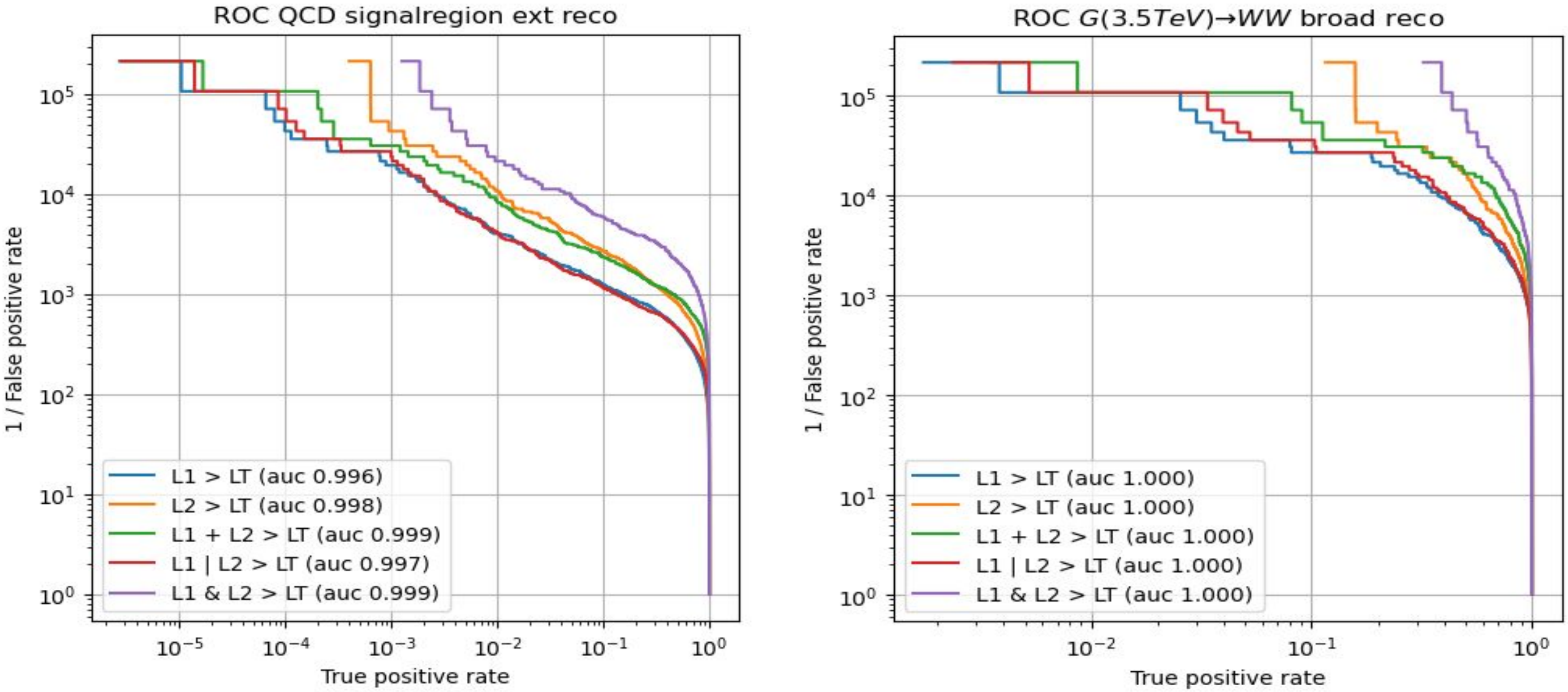
```
-bash-4.2$ nvidia-smi
Wed Dec  8 15:02:00 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        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  | 00000000:18:00.0 Off |                    0 |
| N/A   43C    P0    27W /  70W |    256MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla T4            Off  | 00000000:3B:00.0 Off |                    0 |
| N/A   44C    P0    27W /  70W |    256MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-PCIE...  Off  | 00000000:86:00.0 Off |                    0 |
| N/A   41C    P0    36W / 250W |  31461MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    163687      C   python3                           253MiB |
|    1   N/A  N/A    163687      C   python3                           253MiB |
|    2   N/A  N/A    163687      C   python3                         31457MiB |
+-----------------------------------------------------------------------------+
```

Only TeslaV100 in use for inference

TESLA T4 during the ONNX-TensorRT GPU model Inference Run

```
-bash-4.2$ nvidia-smi
Wed Dec  8 17:01:04 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| 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  | 00000000:18:00.0 Off |                    0 |
| N/A   44C    P0    32W /  70W |   1215MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla T4            Off  | 00000000:3B:00.0 Off |                    0 |
| N/A   45C    P0    27W /  70W |    256MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-PCIE...  Off  | 00000000:86:00.0 Off |                    0 |
| N/A   41C    P0    36W / 250W |  31157MiB / 32510MiB |      6%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    186509      C   python3 TeslaT4 in use during inference 1212MiB |
|    1   N/A  N/A    186509      C   python3                          253MiB |
|    2   N/A  N/A    186509      C   python3                        31153MiB |
+-----------------------------------------------------------------------------+
```

C-VAE architecture (Jet-level): Signal to Background classification performance

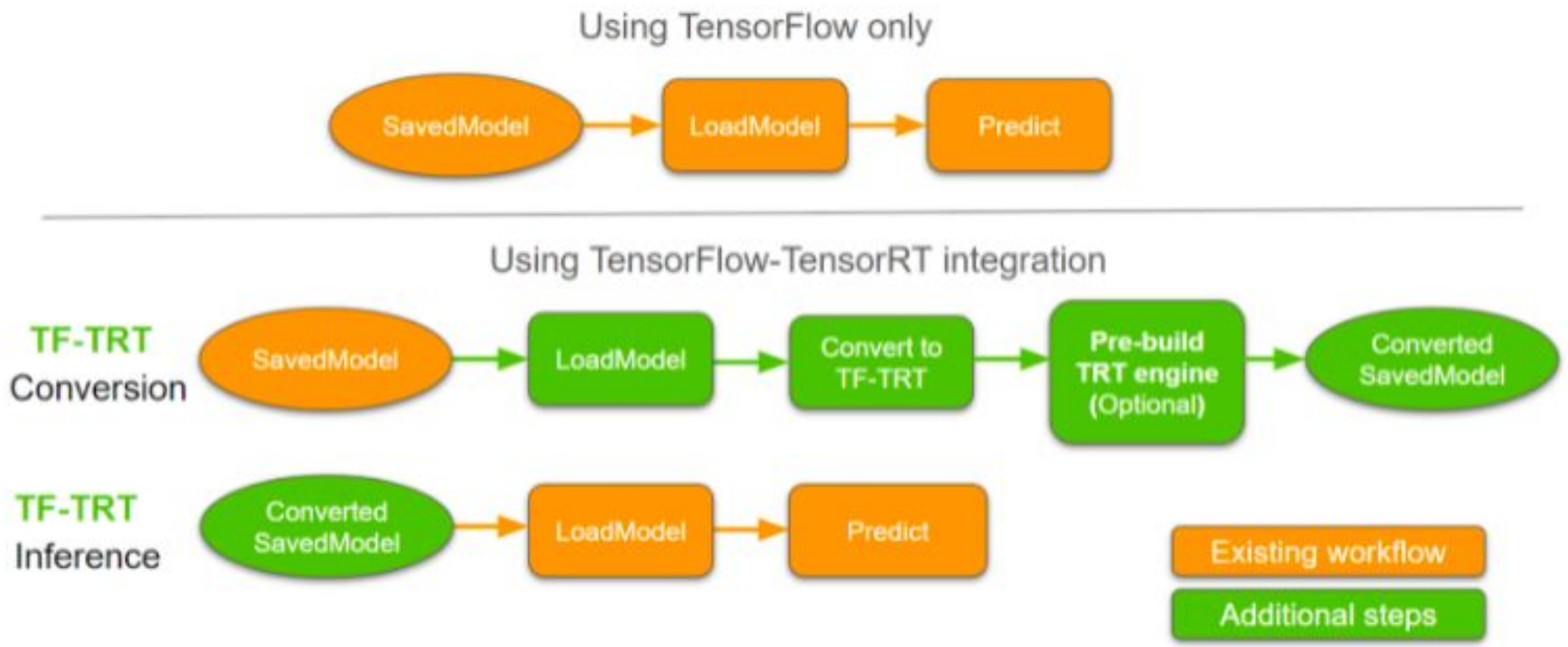


ONNX model to TensorRT (TRT) TRTExec tool for TRT engine creation

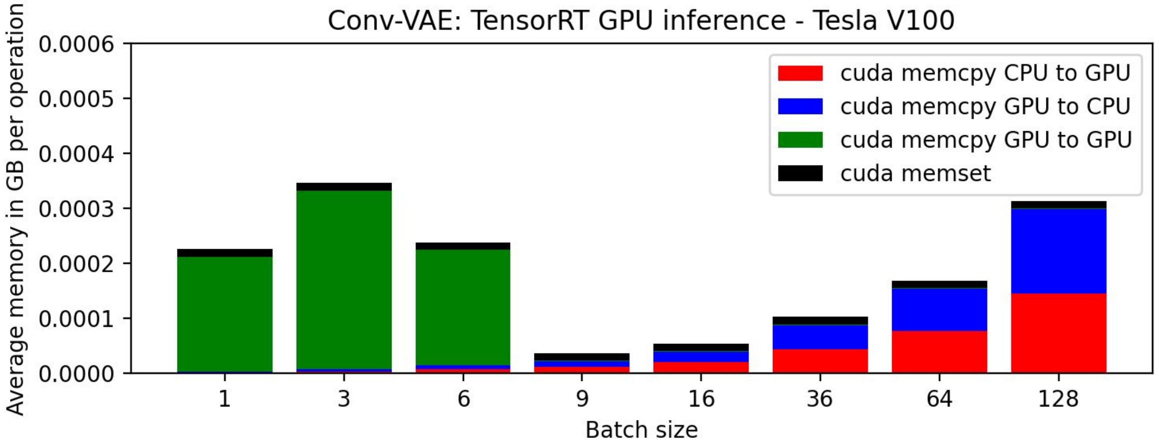
- TRTexec is successful (shows the PASSED message at the end of the run) and generates the TRT engine file from the VAE onnx model

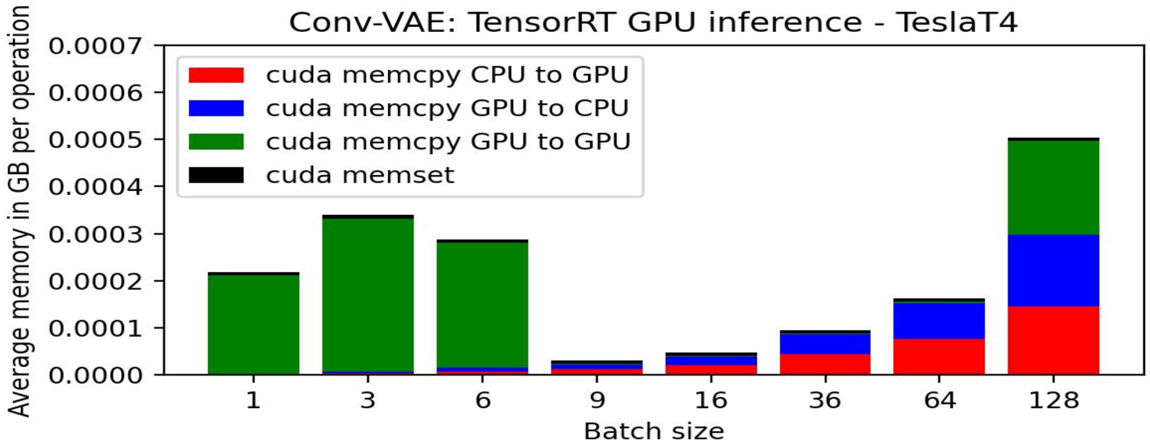
```
[11/29/2021-17:04:38] [I] Average on 10 runs - GPU latency: 1.05776 ms - Host latency: 1.06914 ms (end to end 1.07378 ms, enq
ueue 1.05381 ms)
[11/29/2021-17:04:38] [I] Average on 10 runs - GPU latency: 0.941919 ms - Host latency: 0.952978 ms (end to end 0.959644 ms,
enqueue 0.93938 ms)
[11/29/2021-17:04:38] [I] Average on 10 runs - GPU latency: 0.948535 ms - Host latency: 0.959863 ms (end to end 0.966528 ms,
enqueue 0.945215 ms)
[11/29/2021-17:04:38] [I] Host Latency
[11/29/2021-17:04:38] [I] min: 0.88623 ms (end to end 0.896118 ms)
[11/29/2021-17:04:38] [I] max: 6.20422 ms (end to end 6.22974 ms)
[11/29/2021-17:04:38] [I] mean: 1.05717 ms (end to end 1.06465 ms)
[11/29/2021-17:04:38] [I] median: 0.953857 ms (end to end 0.960449 ms)
[11/29/2021-17:04:38] [I] percentile: 1.98767 ms at 99% (end to end 2.00037 ms at 99%)
[11/29/2021-17:04:38] [I] throughput: 0 qps
[11/29/2021-17:04:38] [I] walltime: 3.00198 s
[11/29/2021-17:04:38] [I] Enqueue Time
[11/29/2021-17:04:38] [I] min: 0.875366 ms
[11/29/2021-17:04:38] [I] max: 6.17004 ms
[11/29/2021-17:04:38] [I] median: 0.939758 ms
[11/29/2021-17:04:38] [I] GPU Compute
[11/29/2021-17:04:38] [I] min: 0.874512 ms
[11/29/2021-17:04:38] [I] max: 6.18054 ms
[11/29/2021-17:04:38] [I] mean: 1.04411 ms
[11/29/2021-17:04:38] [I] median: 0.942139 ms
[11/29/2021-17:04:38] [I] percentile: 1.96106 ms at 99%
[11/29/2021-17:04:38] [I] total compute time: 2.89219 s
&&&& PASSED TensorRT.trtexec # trtexec --onnx=VAE_test_nov28_v4.onnx --verbose --saveEngine=vae_onnx.trt
```

Native Tensorflow to TensorRT model inference with GPUs



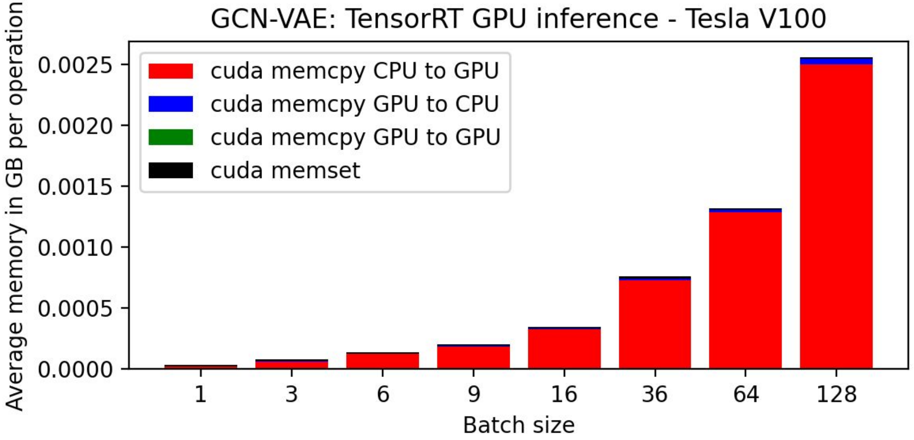
Optimized ONNX-TensorRT GPU inference for C-VAE: Average memory (Tesla V100 vs. Tesla T4)

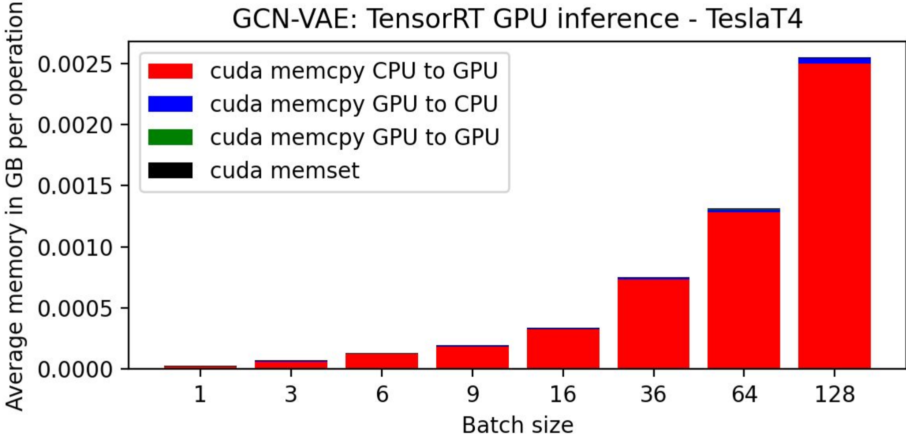




The memory copy operations (cpu to gpu) and (gpu to cpu) dominates at higher batches

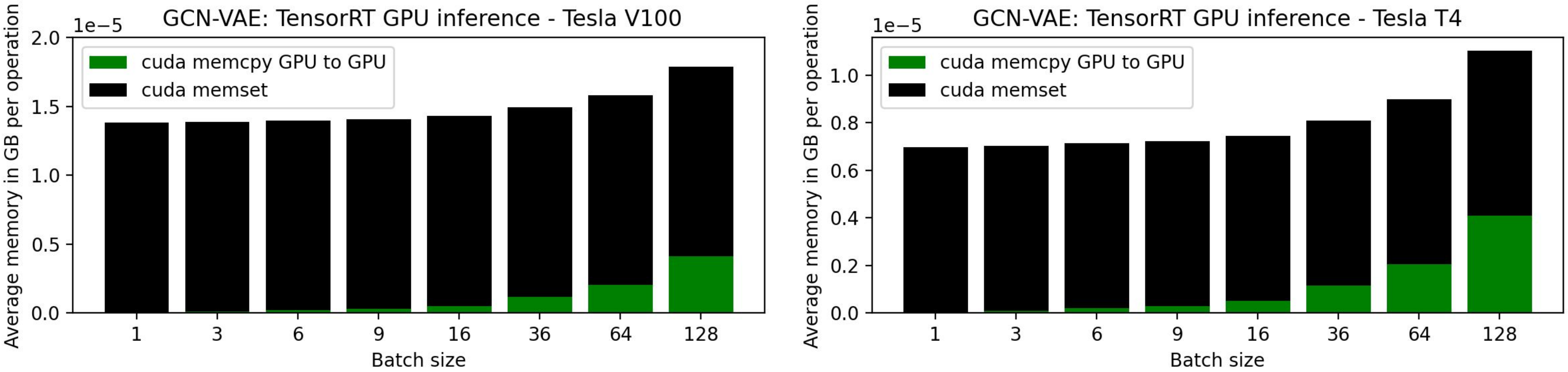
Optimized ONNX-TensorRT GPU inference for GCN-VAE: Average memory (Tesla V100 vs. Tesla T4)





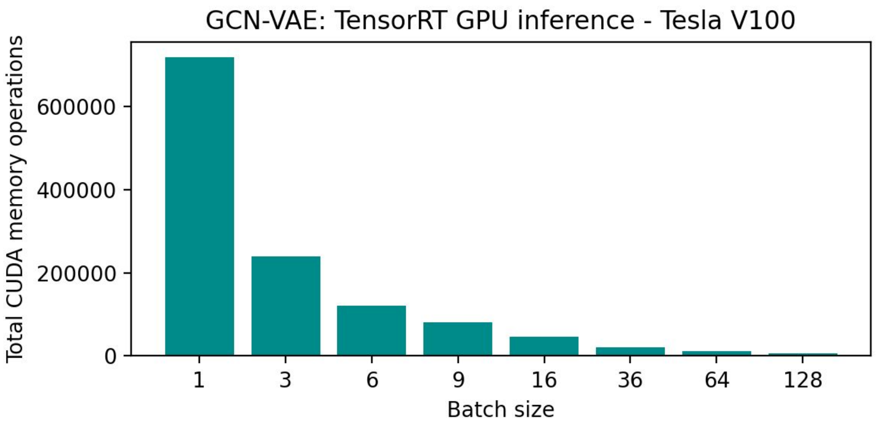
The CUDA memory copy (cpu to gpu) dominates significantly over all the batch sizes

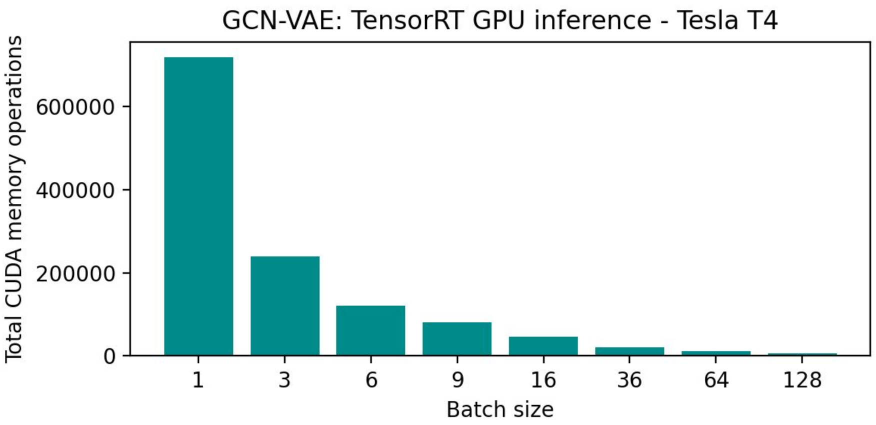
Optimized ONNX-TensorRT GPU inference for GCN-VAE: Average memory



Zooming into the average memory contribution of cuda memory copy (gpu to gpu)

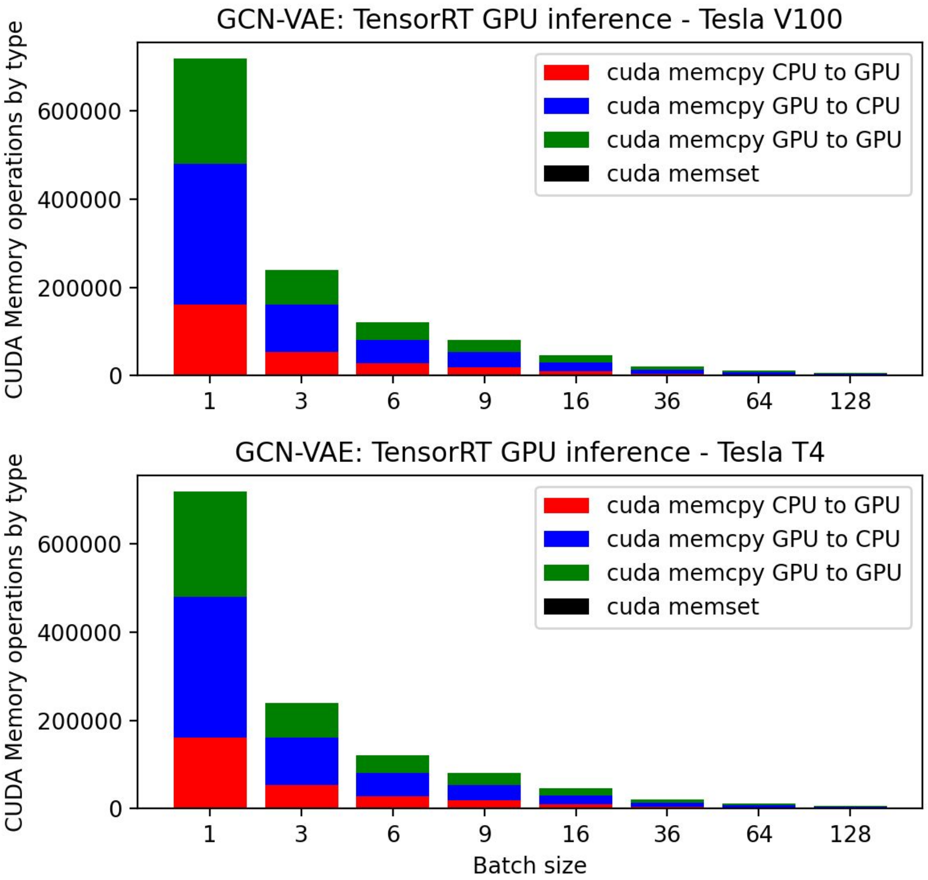
Optimized ONNX-TensorRT GPU inference for GCN-VAE: Total CUDA memory operations (Tesla V100 vs. Tesla T4)





CUDA memory operations decline drastically according to the increase in batch size

Optimized ONNX-TensorRT GPU inference for GCN-VAE: Memory operations by type (Tesla V100 vs. Tesla T4)



The cuda memory copy (cpu to gpu) and (gpu to cpu) operations dominate over different batch sizes