# Nvidia Codec Decoder + TRT inference Case Study

Contents

[Nvidia Codec Decoder + TRT inference Case Study 1](#_Toc44320907)

[1. Overview 1](#_Toc44320908)

[2. Software Environment Configuring 2](#_Toc44320909)

[2.1. Configuring TRT 2](#_Toc44320910)

[2.2. Configuring Codec SDK 2](#_Toc44320911)

[2.3. Multi-stream decoding + TRT Inference 4](#_Toc44320912)

[2.3.1. Benchmarking 4](#_Toc44320913)

# Overview

This is a case study for end2end performance benchmarking, combining multi-stream video Decoding + TRT inference for a ResNet50 ONNX model, in a single process multiple threads environment. End2end performance benchmarking for codec + inference is the first step for nowadays CV tasks. This case study involves decoding multiple streams of videos, transform the format from NV12 to RGB, in range [-1.0,1.0], and then import to TRT for inference.

We modify two samples from Nvidia Codec SDK Video\_Codec\_SDK\_9.0.20 and NGC TensorRT container nvcr.io/nvidia/tensorrt:19.09-py3 respectively. We use a sample video from NGC DeepStream container, Deepstream4.0/samples/streams/sample\_720p.mp4. ResNet50 model is downloaded from here <https://s3.amazonaws.com/download.onnx/models/opset_9/resnet50.tar.gz>, and for highest throughput performance, only TRT INT8 mode is tested, with pre-calibrated dynamic layer ranges. All of the test cases are run on one V100 16GB GPU in a DGX1 server.

|  |
| --- |
| V100 16GB  Cuda 10.1  GPU driver 418.67  nvcr.io/nvidia/tensorrt:19.09-py3  Video\_Codec\_SDK\_9.0.20,  Deepstream4.0/samples/streams/sample\_720p.mp4  wget https://s3.amazonaws.com/download.onnx/models/opset\_9/resnet50.tar.gz, INT8 |

Table , Software configuration

# Software Environment Configuring

The simplest way to configure software environment is to download a docker container from NGC. First pull a TRT container from NGC; then download Video Codec SDK package to a /my/projects dir. Follow the instruction here before configuring, <https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#pre-installation-actions>, install CUDA if not installed.

# Configuring TRT

We use this command line to run the TRT container:

nvidia-docker run -it -v /home/username/:/username --net host nvcr.io/nvidia/tensorrt:19.09-py3 bash

Next run TRT sample sampleINT8API and verifying its validation, we merge it with decoding code for end2end benchmarking.

|  |
| --- |
| Run TRT workspace/tensorrt/samples/sampleINT8API |
| Make sure GPU driver、CUDA toolkit are ready:  >nvidia-smi  Download TRT from NGC and run container:  nvidia-docker run -it -v /home/username/:/username --net host nvcr.io/nvidia/tensorrt:19.09-py3 bash  Goto data dir:  cd ./tensorrt/data/  Download ResNet50 ONNX model:  >wget <https://s3.amazonaws.com/download.onnx/models/opset_9/resnet50.tar.gz>  >tar -xvzf resnet50.tar.gz, and copy model.onnx to /workspace/tensorrt/data/int8\_api/resnet50.onnx  >cd /workspace/tensorrt/samples/sampleINT8API  >make  >./sample\_int8\_api --data=../../data/ --model=../../data/int8\_api/model.onnx --image=../../data/int8\_api/airliner.ppm --ranges=../../data/int8\_api/resnet50\_per\_tensor\_dynamic\_range.txt --reference=../../data/int8\_api/reference\_labels.txt  Configuration is valid. |

# Configuring Codec SDK

|  |
| --- |
| >Run container as in described in 2.1.  Validify codec lib is installed, codec is part of CUDA toolkit. In host:  username@node2:~$ ldconfig -p | grep libnvcuvid.so  libnvcuvid.so.1 (libc6,x86-64) => /usr/lib/x86\_64-linux-gnu/libnvcuvid.so.1  libnvcuvid.so (libc6,x86-64) => /usr/lib/x86\_64-linux-gnu/libnvcuvid.so  In container:  root@781c711977c4:/workspace# ldconfig -p | grep libnvcuvid.so  libnvcuvid.so.1 (libc6,x86-64) => /usr/lib/x86\_64-linux-gnu/libnvcuvid.so.1  if no similar output printed, that means you need to reinstall the GPU driver, or copy codec lib from somewhere. |
| Before run Codec SDK samples, you need more installation:  >cd /my/projects/codec/  Install nv-codec-headers：  >git clone https://git.videolan.org/git/ffmpeg/nv-codec-headers.git  >cd nv-codec-headers && make install && cd –  More installation:  >apt-get update  apt-get install build-essential yasm cmake libtool libc6 libc6-dev unzip wget libnuma1 libnuma-dev  apt-get install pkgconf  download and install FFmpeg:  git clone [https://git.ffmpeg.org/ffmpeg.git](https://git.ffmpeg.org/ffmpeg.git%20ffmpeg/)  cd ./ffmpeg  ./configure --enable-nonfree --enable-cuda --enable-cuda-nvcc --enable-cuvid --enable-nvenc --enable-libnpp --extra-cflags=-I/usr/local/cuda/include --extra-cflags=-I/usr/local/include/ffnvcodec --extra-ldflags=-L/usr/local/cuda/lib64  make -j 8  make install |
| Download Codec SDK and put it here /my/projects/:  cd /my/projects/Video\_Codec\_SDK\_9.0.20/Samples/AppDecode/AppDecMultiInput  make  下载DeepStream container,  cp /my/projects/Deepstream4.0/deepstream\_x86\_public/deepstream\_x86\_public/samples/streams/ sample\_720p.mp4 ./  make  ./AppDecMultiInput -i ./sample\_720p.mp4 -o ./output -gpu 0 -thread 4  Output log is:  *GPU in use: Tesla V100-SXM2-16GB*  *[INFO ][07:40:44] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)*  *[INFO ][07:40:45] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)*  *[INFO ][07:40:45] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)*  *[INFO ][07:40:45] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)*  *[INFO ][07:40:45] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)*  *Session Initialization Time: 52 ms*  *Session Initialization Time: 64 ms*  *Session Initialization Time: 76 ms*  *Session Initialization Time: 90 ms*  *concurrency streams: 4, each decodes 1442 frames, fps: 130.845*  *Merged video saved in ./output. A total of 1442 frames were decoded.*  *Session Deinitialization Time: 8 ms*  *Session Deinitialization Time: 19 ms*  *Session Deinitialization Time: 7 ms*  *Session Deinitialization Time: 13 ms* |

# Multi-stream decoding + TRT Inference

Refactoring source code, copy TRT sample code from sampleINT8API to AppDecMultiInput, rename the project as codec2TRT, we put source code here for reference, <https://github.com/wujinzhong/CUDA-samples/tree/master/codec2TRT>. Copy source code to Codec SDK Samples dir, /Samples/AppDecode/codec2TRT.

# Benchmarking

Run command line:

>make

>./AppDecMultiInput -thread 4 -maxBatchSize 8

The log looks like this:

|  |
| --- |
| ./AppDecMultiInput -thread 4 -maxBatchSize 8 |
| root@0593980d1c0f:/my/project/codec2TRT# ./AppDecMultiInput -thread 4 -maxBatchSize 8  GPU in use: Tesla V100-SXM2-16GB  [02/11/2020-02:51:22] [I] Please follow README.md to generate missing input files.  [02/11/2020-02:51:22] [I] Validating input parameters. Using following input files for inference.  [02/11/2020-02:51:22] [I] Model File: ./data/model.onnx  [02/11/2020-02:51:22] [I] Image File: ./data/airliner.ppm  [02/11/2020-02:51:22] [I] Reference File: ./data/reference\_labels.txt  [02/11/2020-02:51:22] [I] Dynamic Range File: ./data/resnet50\_permy\_tensor\_dynamic\_range.txt  [02/11/2020-02:51:22] [I] Building and running a INT8 GPU inference engine for ./data/model.onnx  ./data/model.onnx\_bs8\_int8.plan  engine read from ./data/model.onnx\_bs8\_int8.plan!  [INFO ][02:51:24] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)  go in DecProc\_Codec\_Inf for thread0  go in DecProc\_Codec\_Inf for thread1  go in DecProc\_Codec\_Inf for thread2  go in DecProc\_Codec\_Inf for thread3  [INFO ][02:51:24] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)  [INFO ][02:51:24] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)  [INFO ][02:51:24] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)  [INFO ][02:51:24] Media format: QuickTime / MOV (mov,mp4,m4a,3gp,3g2,mj2)  Session Initialization Time: 31 ms  performing inference for frame0 at thread0  performing inference for frame1 at thread0  performing inference for frame2 at thread0  ……  ……  performing inference for frame1440 at thread3  performing inference for frame1441 at thread3  maxBatchSize 8  verifyOutput for batchSize(8)  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  [02/11/2020-02:51:29] [I] [1] table lamp  [02/11/2020-02:51:29] [I] [2] pole  [02/11/2020-02:51:29] [I] [3] medicine chest  [02/11/2020-02:51:29] [I] [4] chime  [02/11/2020-02:51:29] [I] [5] wardrobe  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  [02/11/2020-02:51:29] [I] SampleINT8API result: Detected:  performing inference for frame1440 at thread2  performing inference for frame1441 at thread2  maxBatchSize 8  concurrency streams: 4, each decodes mean 1442 frames, total fps: 1121.44  Merged video saved in ./output. A total of 5768 frames were decoded.  Session Deinitialization Time: 6 ms  Session Deinitialization Time: 5 ms  Session Deinitialization Time: 5 ms  Session Deinitialization Time: 9 ms |

Line “concurrency streams: 4, each decodes mean 1442 frames, total fps: 1121.44” shows what the performance is.