PyNvVideoCodec

Read Me

Table of Contents

Chapter 1. Read Me

1.1. Release Notes

Key Features and Enhancements

This release of PyNvVideoCodec includes support for the following features:

- Codec
  - H.264
  - HEVC
  - AV1
- Surface format
  - NV12 (8 bit)
  - YUV 4:2:0 (10 bit)
  - YUV 4:4:4 (8 and 10 bit)
- Interoperability
  - Supports DLPack to facilitate data exchange with popular DL frameworks like PyTorch and TensorRT.
  - Supports CUDA Array Interface to facilitate data exchange with NVIDIA's CV-CUDA library.
- CUDA stream support for optimizing throughput.
- Contains a collection of Python sample applications that demonstrate the usage of APIs.

Limitations and Known Issues

- DLPack interoperability is supported only for NV12.
- Currently, only pageable allocations are supported.
- PyNvVideoCodec uses the FFmpeg binaries for demuxing of audio and video content.

NVIDIA will not update the FFmpeg binaries included in our release package as these binaries are available, maintained and updated by the FFmpeg open-source community.

> **ATTENTION:** NVIDIA does not provide support for FFMPEG; therefore, it is the responsibility of end users and developers, to stay informed about any vulnerabilities or quality bugs reported against FFMPEG. Users are encouraged to refer to the official FFmpeg website and community forums for the latest updates, patches, and support related to FFmpeg binaries and act as they deem necessary.

Package Contents

This package contains the following:

1. Sample applications demonstrating usage of PyNvVideoCodec APIs for encoding, decoding and transcoding use cases.
   - [.\samples\]
2. Python Bindings
   - [.src\PyNvVideoCodec]
3. Video codec helper classes and utilities
   - [.src\VideoCodecSDKUtils]
4. FFmpeg libraries and source code
   - [.external\ffmpeg]
5. Documents
   - [.docs]

The sample applications provided in the package are for demonstration purposes only and may not be fully tuned for quality and performance. Hence the users are advised to do their independent evaluation for quality and/or performance.

1.2. System Requirements

Table 1. System Requirements

Operating System	▶ Windows 10 or higher ▶ Ubuntu 18.04 or higher
GPU	▶ Turing ▶ Ampere

	▶ Ada ▶ Hopper
Drivers	▶ NVIDIA Windows display driver 531.61 or newer ▶ NVIDIA Linux display driver 530.41.03 or newer Get most recent NVIDIA Display Driver
Python	▶ Python 3.10 ▶ Python 3.10 Dev (required in Ubuntu only)
CMake	▶ (3.21 and onwards) ▶ build-essential (required in Ubuntu only)
Visual Studio(Windows only)	▶ Visual Studio
CUDA Toolkit	Latest CUDA Toolkit
Python modules to run Sample applications	PyCUDA and PyTorch

Windows Subsystem for Linux (WSL) Configuration Requirements

- Add the directory /usr/lib/wsl/lib to PATH environment variable, in case it is not added by default. This is required to include path for the WSL libraries.
- Plus all the requirements under System Requirements

1.3. Installing PyNvVideoCodec Python Module

ATTENTION: This project will download and install additional third-party open source software projects - DLPack. Review the license terms of these open source projects before use.

The Python module can be installed using following ways.

Installing from PyPI

1. The ready-to-use Python WHL's (Wheel) of the PyNvVideoCodec for Windows and Linux OSes are hosted on PyPI.
2. Open the bash/shell prompt and run:

```
$>pip install "PyNvVideCodec"
```

3. This is the recommended way.

Building and Installing from Source on NVIDIA NGC

The package containing PyNvVideCodec Python module's source code, all dependencies, Python sample applications, and documents is hosted on NVIDIA NGC.

Follow these steps:

1. Download the zip file of the latest package from NVIDIA NGC .
2. Open the bash/shell prompt on the same directory where zip was downloaded and run the following command, replacing "PyNvVideCodec.zip" with the actual name of the downloaded zip file:

```
$>pip install "PyNvVideCodec.zip"
```

3. You can access documents and Python sample applications from the package.

Use this method if you need any customization on PyNvVideoCodec Python module e.g. enabling NVTX markers for profiling

Follow these steps to build customized version:

1. Unzip the source package to a directory.
2. Do the necessary modifications to the source.
3. On the same directory where `setup.py` is located, run the following commands:

```
$>pip install .
```

1.4. Running Samples

PyNvVideoCodec package contains the following Python samples in the `PyNvVideoCodec/samples` folder. For each of these samples, you can use the `-h` option to see the available command line options.

Table 2. Command Line Options per Sample Application

Sample Application	Functionality	Example Command Line
`Decode.py`	Illustrates the demuxing and decoding of a media file.	`Decode.py -g 0 -i ip_media_file_path -o op_yuv_file_path -d 1`
`DecodeAsync.py`	Demonstrates how to decode media file into output surfaces allocated on non default cuda	`DecodeAsync.py -g 0 -i ip_media_file_path -o op_yuv_file_path -s`

Sample Application	Functionality	Example Command Line
	stream.Refer Stream Aware Allocations for more details.	
DecodePerf.py	Measures decoding performance in FPS per process	`DecodePerf.py -g 0 -i ip_media_file_path -d 1 -n 1`
Encode.py	Illustrates encoding of frames using CUDA device buffers as input.	`Encode.py -i ip_yuv444_file_path -o op_bistream_path -s 1280x720 -if yuv444 -c hevc -json encode_config.json`
EncodeFromCPUBuffer.py	Illustrates encoding of frames using host memory buffers as input.	`Encode.py -i ip_yuv444_file_path -o op_bistream_path -s 1280x720 -if yuv444 -c hevc -cb 1 -json encode_config.json`
EncodeReconfigure.py	Demonstrates bitrate change at runtime without the need to reset the encoder session.The application reduces the bitrate by half and then restores it to the original value after 100 frames.	`Encode.py -i ip_yuv444_file_path -o op_bistream_path -s 1280x720 -if yuv444 -c hevc -json encode_config_lowlatency.json`
EncodePerf.py	Measures encoding performance in FPS per process.	`Encode.py -i ip_yuv444_file_path -o op_bistream_path -s 1280x720 -if yuv444 -c hevc -json encode_config_perf.json -n 3`
Transcode.py	Demonstrates transcoding of an input video stream.	`Transcode.py -g 0 -i ip_media_file_path -o op_media_file_path -c h264`
TranscodeWithPostProc.py	Demonstrates zero copy data exchange with PyTorch, does the transcoding of an input video stream, runs a clamping kernel on decoded output and encodes it back.	`TranscodeWithPostProc.py -g 0 -i ip_media_file_path -o op_media_file_path`

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgment, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, CUDA Toolkit, cuDNN, DALI, DIGITS, DGX, DGX-1, DGX-2, DGX Station, DLProf, GPU, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NVCaffe, NVIDIA Deep Learning SDK, NVIDIA Developer Program, NVIDIA GPU Cloud, NVLink, NVSHMEM, PerfWorks, Pascal, SDK Manager, Tegra, TensorRT, TensorRT Inference Server, Tesla, TF-TRT, Triton Inference Server, Turing, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com