NETINT APPS057-Codensity T408™ VQ and Latency Testing Application Note

V1.0

Table of Contents

1 Table of Abbreviations

Abbreviation:	Full Form:
AI	Artificial Intelligence
API	Application Programming Interface
AUD	Access Unit Delimiters
AV	Audio Video
AV1	Alliance Open Media Video 1 Codec
CABAC	Context Adaptive Binary Arithmetic Coding
CBR	Constant Bit Rate
CPU	Central Processing Unit
CRF	Constant Rate Factor
DSP	Digital Signal Processing
EOF	End Of File
FIXQB	Fixed Quantization Parameter
FF	Form Factor
FPS	Frames Per Second
GOP	Group of pictures
HDR	High Dynamic Range
HHHL	Half Height, Half Length
HHFL	Half Height, Full Length
HLG	Hybrid Log Gamma
HRD	Hypothetical Reference Decoder
IE	Inference Engine
Mbps	Mega Bit per second
MBps	Mega Byte per second
NLP	Natural Language Processing
NPU	Neural Processing Unit
NVMe	Non-Volatile Memory express
PCIe	Peripheral Component Interconnect express
PPS	Picture Parameter Set
QP	Quoted Printable
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RDO	Optimization
ROI	Region of interest
SDK	Software Development Kit
SEI	Supplemental Enhancement Information
SRIOV	Single Root I/O Virtualization
SPS	Sequence Parameter Set
VBR	Variable Bit Rat
VCL	Video Coding Layer
VPS	Video Parameter Set
VUI	Video Usability Information

NETINT © 2022

2 References

Technical Notes: APPS032 – Codensity T408 Latency Measurement Logger Application Note

APPS047 – Codensity T408 Decklink Interworking Configuration and Usage Application note

QuickStartGuideT408_T432_FW3_0

IntegrationProgrammingGuideT408_T432_FW3_0

3 Background

This document describes how to use the metrics measurement scripts provided. This document will go over how to run the scripts and how to use the data that is generated after.

3.1 Intended Audience

This document is intended for customers and engineers who are testing NETINT transcoders to get VQ and latency data for comparisons against other encoders/decoders.

3.2 Compatibility

Software Compatibility

This guide is intended to be used with NETINT Codensity T408 Video Transcoder software Release 3.0.0 and above.

Hardware Compatibility

Release 3.0.0 and above supports NETINT Codensity T408 Video Transcoder hardware.

Operating system

All OS supporting T408 Video Transcoder.

4 VQ Data Collection

4.1 Setting Up

Before running the scripts, make sure the system environment and the NETINT cards are setup correctly. Follow the steps on the Quick Start Guide provided in the release package to setup the cards and the required software including FFmpeg.

In addition to the T408 software and FFmpeg, VMAF will also need to be installed to run the scripts. You can install VMAF on your system by cloning the library from git (https://github.com/Netflix/vmaf.git) and following the instructions to install the VMAF CLI from here: https://github.com/Netflix/vmaf/blob/master/libvmaf/README.md

Once everything is setup get the VQ and latency measurement scripts form NETINT and extract them into a folder as shown below:

```
nvme@CLI137:~/scripts$ ls
codec_parameters.json  genLatencyList.py  run_latency_test.py
config.json            get_latency.sh     run_test.py
EncodingRecord.py      inputs             VmafCommand.py
FFmpegCommand.py       outputs            vmafplus_v0.5.0.json_720.json
FFProbeAnalysis.py     __pycache__
```

Figure 1: Scripts extracted into /home/[user]/scripts folder

4.2 Configuring the parameters

4.2.1 Source Files

Once all the files are extracted, place all the source files that are going to be tested into the *inputs* folder. Make sure the source files are raw video files of *".y4m"* file format (raw video with metadata). Shown below is an example on how to convert a file to *".y4m"*:

ffmpeg -i bbb_sunflower_1080p_30fps_normal.mp4 -pix_fmt yuv420p -vsync 0 source.y4m

Note: If you want to test a SDI input or a livestream input, please capture the source and convert to y4m file before running your tests.

4.2.2 Testing Parameters

The testing parameters can be modified using two JSON files:

1. config.json

Below is a snapshot of the config.json file:

```json
{
    "paths":
    {
        "ffmpeg": "ffmpeg",
        "ffprobe": "ffprobe",
        "vmaf": "vmaf",
        "input_folder": "./inputs/",
        "output_folder": "./outputs/",
        "data_folder": "./outputs/data",
        "bitstream_folder": "./outputs/bitstreams"
    },
    "sequence_list":
    [
        "EuroTruckSimulator_720p30.y4m",
        "Life_1080p30.y4m"
    ],
    "codec_list":
    [
        "h264",
        "h265",
        "x264",
        "x265"
    ],
    "bitrates":[5000, 3500],
    "instances": 2
}
```

Figure 2: config.json file

The paths section has the required paths for the input and output folders and the application bin paths for FFmpeg, FFprobe and VAMF. The example shown above uses default paths. If you wish to use a specific FFmpeg version installed on your system, you must set the ffmpeg path in the json file above.

The sequence list section should have the names of all the source files (which are in the *inputs* folder). These are the files that will be tested so make sure to include all the files that need to be tested.

The codec list section will have the list of codecs that will be tested. This section supports up to 4 values:

1. h264 – Netint H264 Encoder

2. h265 – Netint H265 Encoder
3. x264 – Libx264 Software Encoder
4. x265 – Libx265 Software Encoder

The bitrates section is a list of bitrates in kbps. Update this section with all the bitrates that you want to test. In the screenshot above you can see two bitrates have been listed: 5000kbps and 3500 kbps.

The instances section is used for latency testing. This value sets the number of FFmpeg instances that will run in parallel. It is only used for latency testing. This option is ignored for VQ testing.

2. codec_parameters.json

Figure 3 is a snapshot of codec_parameters.json file:

For every codec there are two sections: codec_name and parameters. The codec_name must not be changed as it sets the encoder for ffmpeg. The parameters section contains encoding parameters that will be tested.

For h264 and h265 sections, netint parameters can be used. Refer to the Integration Programming Guide provided with the release for the parameters you can use with T408/T432 card.

Parameters for x264 can be found here:

https://trac.ffmpeg.org/wiki/Encode/H.264

x264 parameters can be used for x265 as well. More x265 parameters can be found here:

https://x265.readthedocs.io/en/master/

```
{
    "h264":
    {
        "codec_name": "h264_ni_logan_enc",
        "parameters":
        {
            "RcEnable": 1,
            "profile": 2,
            "gopPresetIdx": 2,
            "lowDelay": 1
        }

    },
    "h265":
    {
        "codec_name": "h265_ni_logan_enc",
        "parameters":
        {
            "RcEnable": 1,
            "profile": 2,
            "gopPresetIdx": 2,
            "lowDelay": 1
        }
    },
    "x264":
    {
        "codec_name": "libx264",
        "parameters":
        {
            "preset": "fast"
        }
    },
    "x265":
    {
        "codec_name": "libx265",
        "parameters":
        {
            "preset": "fast"
        }
    }
}
```

Figure 3: codec_parameters.json file

4.3 Running the VQ Test

Once the configuration is done, the VQ test can be run. The following command is used to run the test:

python3 run_test.py

Once the test starts the console output will be similar to the figure shown below:

```
***********************************
ffmpeg -hide_banner -i ./inputs/Life_1080p30.y4m -c:v h264_ni_logan_enc -xcoder-params Rc
Enable=1:profile=2:gopPresetIdx=2:lowDelay=1 -b 5000k -y ./outputs/bitstreams/Life_1080p3
0.y4m_5000_h264.264
Input #0, yuv4mpegpipe, from './inputs/Life_1080p30.y4m':
  Duration: 00:00:04.33, start: 0.000000, bitrate: 746497 kb/s
    Stream #0:0: Video: rawvideo (I420 / 0x30323449), yuv420p(progressive), 1920x1080, SA
R 1:1 DAR 16:9, 30 fps, 30 tbr, 30 tbn, 30 tbc
Please use -b:a or -b:v, -b is ambiguous
Stream mapping:
  Stream #0:0 → #0:0 (rawvideo (native) → h264 (h264_ni_logan_enc))
Press [q] to stop, [?] for help
[h264_ni_logan_enc @ 0x55ba97de3cc0] Session state: 0 allocate frame fifo.
[h264_ni_logan_enc @ 0x55ba97de3cc0] dts offset: 0
Output #0, h264, to './outputs/bitstreams/Life_1080p30.y4m_5000_h264.264':
  Metadata:
    encoder         : Lavf58.45.100
    Stream #0:0: Video: h264 (h264_ni_logan_enc), yuv420p, 1920x1080 [SAR 1:1 DAR 16:9],
q=2-31, 5000 kb/s, 30 fps, 30 tbn, 30 tbc
    Metadata:
      encoder         : Lavc58.91.100 h264_ni_logan_enc
frame=   62 fps=0.0 q=-0.0 size=    1024kB time=00:00:02.06 bitrate=4059.0kbits/s speed=4
frame=  121 fps=119 q=-0.0 size=    2304kB time=00:00:04.03 bitrate=4679.6kbits/s speed=3
frame=  130 fps=118 q=-0.0 Lsize=    2636kB time=00:00:04.33 bitrate=4982.8kbits/s speed=
3.94x
video:2636kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead:
 0.000000%
Encoder_complete_info: session_id 0x8193, total frames input: 130  buffered: 0  completed
: 130  output: 130  dropped: 0 ,  inst_errors: 0
Encoder HW[0] INST[127]-average usage:40%
************************************
```

Figure 4: Console log of run_test.py

When the test is done the following message will be displayed on your console:

End of all encoding

The VQ data of the test will be stored in the *data* folder within the *outputs* folder (*outputs/data).* There will be two types of logs for this test:

1. VMAF logs: the vmaf and psnr scores of the encoded clip. This log will contain the overall score and the frame-by-frame score.
2. CSV files: the test will output 2 *.csv* files: *data_h264.csv* and *data_h265.cvs*. These csv files will contain all the VQ data in a condensed from that can be open in an excel sheet for ease of use. The data from the csv file can be used for comparing the VMAF and PSNR scores between the Netint encoder and software encoders.

input	output	codec	width	height	fps	encoder	target_bit	actual_bit	avg_vmaf	avg_psnr
./inputs/L	./outputs/	h264	1920	1080	30	h264_ni_l	5000	4982848	88.87	35.42
./inputs/L	./outputs/	h264	1920	1080	30	h264_ni_l	3500	3626894	84.38	34.31

Figure 5: Sample Output of VQ Test

All the encoded bitstream will be in the *bitstreams* folder within the *outputs* folder (*outputs/bitstreams*).

5 Latency Data Collection

5.1 Setting Up

To collect latency data, libxcoder much be configured to output the encoder latency when encoding. Please refer to APPS032 – Codensity T408 Latency Measurement Logger Application Note for steps on configuring libxcoder.

For the rest of the setup please follow steps from Section 5.1 of this document.

5.2 Configuring the parameters

For configuring the test parameters, please refer to Section 5.2.2.

Note: For latency testing only Netint encoders will be run, other encoders will be ignored for this test.

5.3 Running the Latency Test

Once everything is setup and configured the latency test can be run using the following command:

python3 run_latency_test.py

The console output when the test is running will look like the figure below:

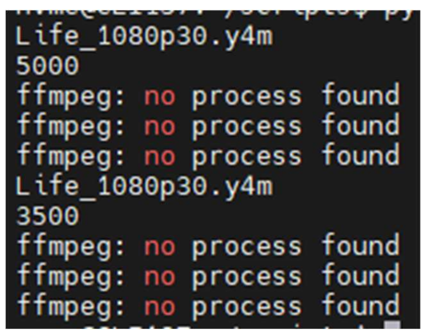

Figure 6: Console output of latency test

Once the test is run, it will generate 2 *.csv* files: *latency_data_h264s.cvs* and *latency_data_h265s.csv.* These files will have the data in the following format:

Input file, resolution, number of instances, target bitrate, latency in milliseconds

DOTA2	h264	720	1	3000	2.331023925
DOTA2	h264	720	1	3000	3.331659804
DOTA2	h264	720	64	3000	8.127247181
DOTA2	h264	720	64	3000	37.29617064
DOTA2	h264	1080	1	5000	4.390094592
DOTA2	h264	1080	1	5000	6.545032904
DOTA2	h264	1080	64	5000	49.17703491
DOTA2	h264	1080	64	5000	83.55976648
WITCHER3	h264	720	1	3000	2.347668079
WITCHER3	h264	720	1	3000	3.320694724
WITCHER3	h264	720	64	3000	7.573899716
WITCHER3	h264	720	64	3000	36.78599451
WITCHER3	h264	1080	1	5000	4.378300366
WITCHER3	h264	1080	1	5000	6.624842037
WITCHER3	h264	1080	64	5000	48.52457239
WITCHER3	h264	1080	64	5000	84.82690773

Figure 7: Sample Output of Latency Test

This data will help see the change in latency with either different encoding parameters or number of instances.

Note: Each run will append to the same *.csv* for latency runs to keep track of old data.

6 Revision History

Version	Change	Date	Author Name
0.1	Draft Document Created	October 20, 2022	Arav Bhatt
1.0	Issued version	October 20, 2022	Arav Bhatt

7 Legal Notice

Information in this document is provided in connection with NETINT products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights1 is granted by this document. Except as provided in NETINT's terms and conditions of sale for such products, NETINT assumes no liability whatsoever and NETINT disclaims any express or implied warranty, relating to sale and/or use of NETINT products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right.

A "Mission Critical Application" is any application in which failure of the NETINT Product could result, directly or indirectly, in personal injury or death. Should you purchase or use NETINT's products for any such mission critical application, you shall indemnify and hold NETINT and its subsidiaries, subcontractors and affiliates, and the directors, officers, and employees of each, harmless against all claims costs, damages, and expenses and reasonable attorney's fees arising out of, directly or indirectly, any claim of product liability, personal injury, or death arising in any way out of such mission critical application, whether or not NETINT or its subcontractor was negligent in the design, manufacture, or warning of the NETINT product or any of its parts.

NETINT may make changes to specifications, technical documentation, and product descriptions at any time, without notice. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications.

NETINT, Codensity, and NETINT Logo are trademarks of NETINT Technologies Inc. All other trademarks or registered trademarks are the property of their respective owners.

© 2022 NETINT Technologies Inc. All rights reserved.