Name of Certifying Engineer(s): Peng Meng

Email of Certifying Engineer(s):pengmeng@tencent.com

Name of System Under Test: clx-8255c\_pytorch

Division (check one):

* Open
* Closed

Category (check one):

* Available
* Preview
* Research, Development, and Internal (RDI)

Benchmark (check one):

* MobileNet
* SSD-MobileNet
* ResNet
* SSD-1200
* NMT
* Other, please specify:

Scenario (check one):

* Single-Stream
* Multi-Stream
* Server
* Offline

90th Percentile Latency: N/A

97th Percentile Latency: N/A

99th Percentile Latency: N/A

Does the submission meet the latency target? (check one)

* Yes (Single-Stream and Offline no requirements)
* Yes (MobileNet x Multi-Stream 50 ms @ 99%)
* Yes (MobileNet x Server 10 ms @ 99%)
* Yes (SSD-MobileNet x Multi-Stream 50 ms @ 99%)
* Yes (SSD-MobileNet x Server 10 ms @ 99%)
* Yes (ResNet x Multi-Stream 50 ms @ 99%)
* Yes (ResNet x Server 15 ms @ 99%)
* Yes (SSD-1200 x Multi-Stream 66 ms @ 99%).
* Yes (SSD-1200 x Server 100 ms @ 99%)
* Yes (NMT x Multi-Stream 100 ms @ 97%)
* Yes (NMT x Server 250 ms @ 97%)
* No

Number of Queries: 1

Samples per Query: 415800

Does the submission meet the appropriate minimum number of queries or samples depending on the Scenario x Benchmark? (check one)

* Yes (Single-Stream 1,024 queries)
* Yes (Offline 24,576 samples)
* Yes (NMT Server and Multi-Stream 90,112 queries)
* Yes (Image Models Server and Multi-Stream 270,336 queries)
* No

Accuracy: 75.710%

Does the submission meet the accuracy target? (check one)

* Yes (MobileNet 71.68% x 98%)
* Yes (SSD-MobileNet 0.22 mAP x 99%)
* Yes (ResNet 76.46% x 99%)
* Yes (SSD-1200 0.20 mAP x 99%)
* Yes (NMT 23.9 BLEU x 99%)
* No

In accuracy mode, did the submission run on the whole accuracy set? (check one)

* Yes
* No

How many samples are loaded into the QSL in performance mode?

415800

Does the number of loaded samples in the QSL in performance mode meet the minimum requirement? (check one)

* Yes (ResNet and MobileNet 1,024 samples)
* Yes (SSD-MobileNet 256 samples)
* Yes (SSD-1200 64 samples)
* Yes (NMT 3,903,900 samples)
* No

Is the experimental duration greater than or equal to 60 seconds? (check one)

* Yes
* No

Does the submission use LoadGen? (check one)

* Yes
* No

Is your loadgen commit from one of these allowed commit hashes?

* 61220457dec221ed1984c62bd9d382698bd71bc6
* 5684c11e3987b614aae830390fa0e92f56b7e800
* 55c0ea4e772634107f3e67a6d0da61e6a2ca390d
* d31c18fbd9854a4f1c489ca1bc4cd818e48f2bc5
* 1d0e06e54a7d763cf228bdfd8b1e987976e4222f
* Other, please specify:

Do you have any additional change to Loadgen?

* Yes, please specify: (random seed and set no trace patch), please refer to our README
* No

Does the submission run the same code in accuracy and performance modes? (check one)

* Yes
* No

Where is the LoadGen trace stored? (check one)

* Host DRAM
* Other, please specify: There is no trace

Is the submission run the correct number of times for the relevant scenario? (check one)

* Yes (Single-Stream, Multi-Stream, Offline 1x)
* Yes (Server 5x)
* No

Are the weights calibrated using data outside of the calibration set? (check one)

* Yes
* No

What untimed pre-processing does the submission use? (check all that apply)

* Resize
* Reorder channels or transpose
* Pad
* A single crop
* Mean subtraction and normalization
* Convert to whitelisted format
* No pre-processing
* Other, please specify:

What numerics does the submission use? (check all that apply)

* INT4
* INT8
* INT16
* UINT8
* UINT16
* FP11
* FP16
* BF16
* FP32
* Other, please specify:

Which of the following techniques does the submission use? (check all that apply)

* Wholesale weight replacement
* Weight supplements
* Discarding non-zero weight elements
* Pruning
* Caching queries
* Caching responses
* Caching intermediate computations
* Modifying weights during the timed portion of an inference run
* Weight quantization algorithms that are similar in size to the non-zero weights they produce
* Hard coding the total number of queries
* Techniques that boost performance for fixed length experiments but are inapplicable to long-running services except in the offline scenario
* Using knowledge of the LoadGen implementation to predict upcoming lulls or spikes in the server scenario
* Treating beams in a beam search differently. For example, employing different precision for different beams
* Changing the number of beams per beam search relative to the reference
* None of the above

Is the submission congruent with all relevant MLPerf rules?

* Yes
* No

Does the submission accurately reflect the real-world performance of the SUT?

* Yes
* No