habana
GOYA

Goya Inference Platform White Paper

Nov 2020

Table of Contents

List of Figures and Tables

The information contained in this document is subject to change without notice.

1. Introduction

Demand for high-performance AI compute has doubled in size rapidly and is accelerating with the multiple domains, growing number of applications and services (e.g. image and gesture recognition in videos, speech recognition, natural language processing, recommendation systems and more), such problems which several years ago were considered difficult for machines to solve, are now solved as accurately as by human beings and more, using deep learning models. As such, deep learning is a transformational technology.

A typical deep learning algorithm comprises of multiple operators, such as matrix multiplication, convolutions, and other tensor operators, which add up to billions of compute-intensive operations. The execution of this massive amount of operations can be accelerated by using the inherent parallel processing that advanced GPUs offer. However, GPUs, which are primarily designed to render graphics in a super-fast way, are not optimized for deep learning workloads. The existing solution's inefficiency for deep learning workloads has a severe impact on the operational costs of cloud providers and data centers. To address this issue, a new class of software programmable AI processors are emerging, designed from the bottom-up for DNN workloads.

Habana's Goya is an AI accelerator card dedicated for inference workloads. The Goya processor is an AI inference processor, designed specifically to deliver superior performance with low latency, power efficiency and cost savings for cloud, data centers and other emerging applications.

2. Deep Learning Workflows – Training and Inference

A deep learning workflow consists of two conceptual steps:

- Training - adjusts neural network model parameters to perform well on given data
- Inference - executes a trained neural network model on new data to obtain the output

For a model to address a specific use case, one first needs to train the model. Once the model is trained, it can be used (for inference). Both training and inference have similar characteristics, but different hardware resource requirements.

During training, a large dataset is processed to train a neural network model so that the model will distinguish between different statistical properties of the samples within the dataset. After the model is ready for use, i.e., the model meets the accuracy goals, the model is ready for deployment. In a production environment, the model is used to efficiently process a new set of inputs to which it was not exposed during training. This operation is called inference and the goal of this phase is to infer attributes in the new data using the trained model.

Although Habana's training and Inference processors uses the same building blocks, there are some fundamental differences which required different architecture solutions - Training workloads require high-bandwidth memories with large capacity, in addition to the memory requirements for chip-to-chip communication. These requirements greatly increase solution BOM and its power consumption.

Inference is critical to support real-time applications such as natural language processing, recommendation systems, speech recognition and many others. Therefore, inference is required to complete with low latency. In addition, providing high throughput with low batch sizes is also critical for inference of many applications. To provide comprehensive inference capabilities, an inference solution should provide high throughput, low latency, low power and be cost effective.

3. GOYA Processor High-level Architecture

The Goya Inference Processor is based on the scalable architecture of Habana's Tensor-Processing Core (TPC) and includes a cluster of eight programmable cores. TPC is Habana's proprietary core designed to support deep learning workloads. It is a VLIW SIMD vector processor with Instruction-Set-Architecture and hardware tailored to serve deep learning workloads efficiently.

The TPC is C/C++ programmable, providing the user with maximum flexibility to innovate, coupled with many workload-oriented features such as: General Matrix Multiply (GEMM) operation acceleration, special-functions dedicated hardware, tensor addressing and latency hiding capabilities. The TPC natively supports these mixed-precision data types: FP32, INT32/16/8, UINT32/16/8. To achieve maximum hardware efficiency, Habana Labs SynapseAI® quantizer tool selects the appropriate data type by balancing throughput and performance versus accuracy. For predictability and low latency, Goya™ is based on software-managed, on-die memory along with programmable DMAs. For robustness, all memories are ECC-protected.

All Goya engines (TPCs, GEMM and DMA) can operate concurrently and communicate via shared memory. For external interface, the processor uses PCIe Gen4x16 enabling communication to any host of choice. The processor includes two 64-bit channels of DDR4 memory interface with max capacity of 16 GB.

The Goya architecture supports mixed precision of both integer and floating points, which allows it to flexibly support different workloads and applications, under quantization controls that the user can specify.

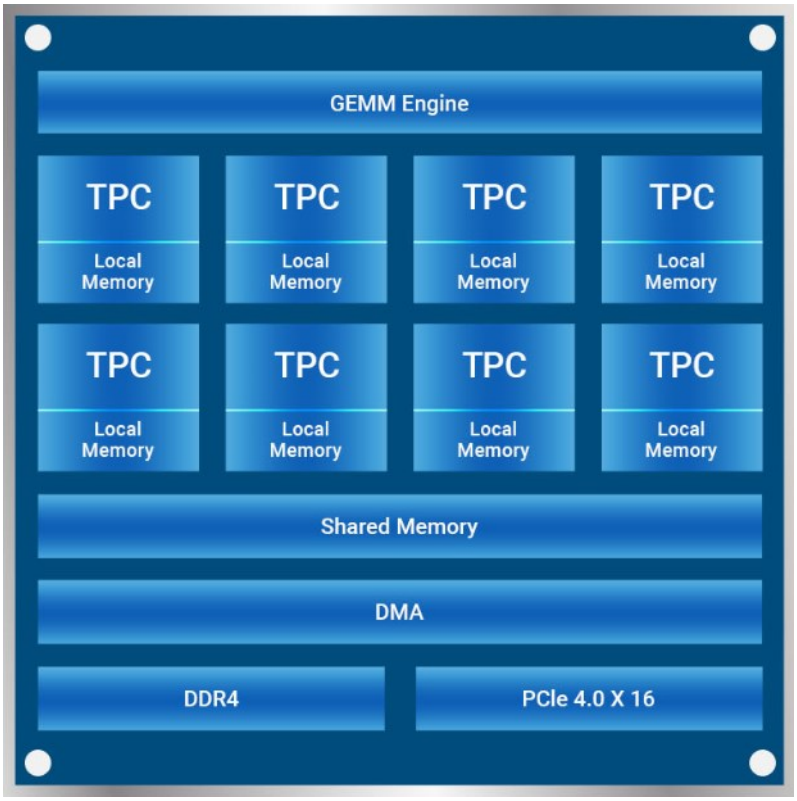

Figure 1 - GOYA High-level Architecture

4. Software Stack and development tools

Habana's software platform is designed to provide a full software stack including flexible development capabilities of the programmable Tensor Processor Cores, the SynapseAI® - Habana's home-grown compiler and runtime, Habana's extensive kernel library and development tools.

Habana provides, as part of its SW package an extensive set of TPC kernel libraries (1400+) and opens its TPC for the user programming, providing a complete TPC tool suite (debugger, simulator, compiler). These tools facilitate the development of customized TPC kernels that can augment the kernels provided by Habana Labs. Thus, users can quickly and easily deploy a variety of network models and algorithms on Goya to innovate and optimize to any unique requirements.

The SynapseAI is built for seamless integration with existing frameworks, that both define a Neural Network for execution and manage the execution Runtime. SynapseAI can be interfaced directly using either C or Python API, It also natively supports ONNX and TensorFlow 2.2 today and will be followed by native PyTorch and ONNX RT support. Integrating natively into DNN frameworks like TensorFlow, SynapseAI enables users to unleash the power of Deep Learning by executing the algorithms efficiently using its high-level software abstraction.

Habana Lab's software stack seamlessly interfaces with all deep learning frameworks. A trained DNN model is first converted into an internal representation. Following this step, ahead-of-time (AOT) compilation is used to optimize the model and create a working plan for the network model execution on the Goya hardware.

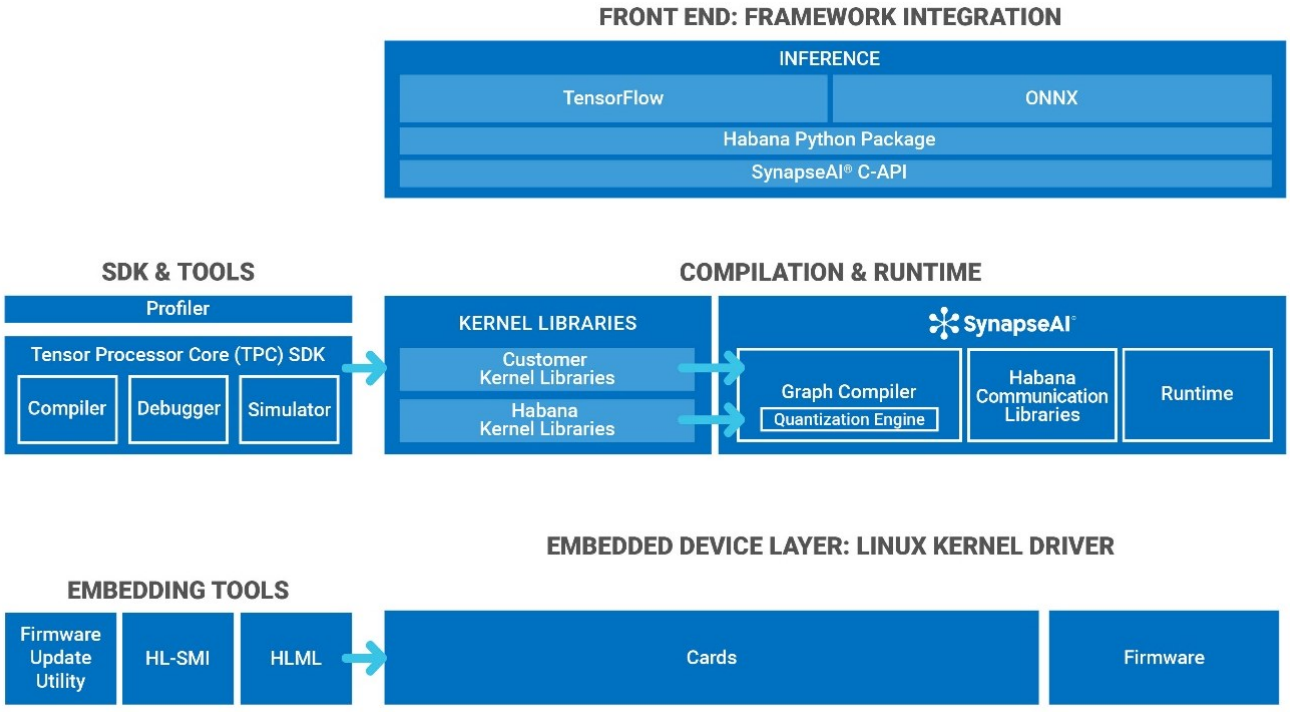

Figure 2 - GOYA Inference Platform – Software Stack

Given TPC's programmability, Goya is a very flexible platform. It enables quick adoption of different deep learning models and is not limited to supporting specific workloads or workloads from a specific domain.

4.1. SynapseAI® - Optimizer and Runtime

Habana Lab's SynapseAI® is a comprehensive software toolkit that simplifies the development and deployment of deep learning models for mass-market use. The SynapseAI® software provides inference network model compilation (Graph Complier) and runtime.

The Goya platform is training platform-agnostic. A DNN can be trained on any hardware platform (GPU, TPU, CPU or any other platform) to obtain a model. SynapseAI® imports the trained model and compiles it for use on the Goya platform. The result is an optimized execution code in terms of accuracy, latency, throughput, and efficiency.

SynapseAI® supports automatic quantization of models trained in floating-point format with near-zero accuracy loss. It receives a model description and representative inputs, and automatically quantizes the model to fixed-point data types, thus greatly reducing execution time and increasing power efficiency.

The user can specify the required level of performance gain and whether some accuracy may be sacrificed to improve performance.

5. GOYA Inference Performance

The key factors used in assessing the performance of the Goya inference platform compared to other solutions are throughput (speed), power efficiency, latency and the ability to support small batch sizes.
Below are performance results for various topologies from Tensorflow, ONNX public repositories and in-house topologies based on public sources.

Software Configuration: Ubuntu v-18.04, SynapseAI v-0.11.447
Hardware Configuration: Goya HL-100 PCIe card, Host: Xeon Gold 6152@2.10Ghz

Model	Imported from	Batch Size	Latency (msec)	Throughput (samples per sec)	Mix Data Types - lowest Precision	Model source
inception V1	TF	1	0.152	12091.4	INT8	http://download.tensorflow.org/models/inception_v1_2016_08_28.tar.gz
		4	0.391	15604.4	INT8	
		8	0.692	16286.9	INT8	
		10	0.839	16470.3	INT8	
inception V3	TF	1	0.396	3374.2	INT8	https://storage.googleapis.com/download.tensorflow.org/models/inception_v3_2016_08_28_frozen.pb.tar.gz
		4	1.225	4324.6	INT8	
		8	2.338	4345.8	INT8	
		10	2.82	4354.7	INT8	
bninception	ONNX	1	0.179	8438.5	INT8	open source model
		4	0.432	12680.4	INT8	
		8	0.763	13420.1	INT8	
		10	0.887	13861.0	INT8	
		20	1.72	14416.4	INT8	
resnet18 V1 224,224	ONNX	1	0.135	14544.6	INT8	https://s3.amazonaws.com/onnx-model-zoo/resnet/resnet18v1/resnet18v1.onnx
		4	0.249	27628.1	INT8	
		8	0.411	30676.6	INT8	
		10	0.493	31566.8	INT8	
resnet18 V2 224,224	ONNX	1	0.184	8836.1	INT8	https://github.com/onnx/models/blob/master/vision/classification/resnet/model/resnet18-v2-7.onnx
		4	0.426	13544.8	INT8	
		8	0.736	14630.4	INT8	
		10	0.892	14845.9	INT8	
resnet34 V1 224,224	ONNX	1	0.175	8569.6	INT8	https://s3.amazonaws.com/onnx-model-zoo/resnet/resnet34v1/resnet34v1.onnx
		4	0.366	15778.3	INT8	
		8	0.617	17406.8	INT8	
		10	0.725	18215.7	INT8	
resnet34 V2 224,224	ONNX	1	0.262	5563.5	INT8	https://github.com/onnx/models/blob/master/vision/classification/resnet/model/resnet34-v2-7.onnx
		4	0.602	9106.8	INT8	
		8	0.993	9985.1	INT8	
		10	1.263	10217.1	INT8	

resnet50 V1 224,224	TF	1	0.202	7142.8	INT8	https://s3.amazonaws.com/onnx-model-zoo/resnet/resnet50v1/resnet50v1.onnx
		4	0.427	12791.0	INT8	
		8	0.74	13964.7	INT8	
		10	0.872	14620.1	INT8	
	ONNX	1	0.199	7491.9	INT8	
		4	0.406	13573.3	INT8	
		8	0.693	14766.6	INT8	
		10	0.819	15487.7	INT8	
resnet50 V1 160x160	ONNX	1	0.176	8759.9	INT8	https://s3.amazonaws.com/onnx-model-zoo/resnet/resnet50v1/resnet50v1.onnx
		4	0.274	20836.9	INT8	
		8	0.417	25095.5	INT8	
		10	0.48	27408.4	INT8	
resnet50 V1 slim 224,224	TF	1	0.202	7433.6	INT8	http://download.tensorflow.org/models/resnet_v1_50_2016_08_28.tar.gz
		4	0.391	14458.6	INT8	
		8	0.655	15977.0	INT8	
		10	0.777	16798.2	INT8	
resnet50 V2 224,224	TF	1	0.293	4686.4	INT8	http://download.tensorflow.org/models/official/20181001_resnet/savedmodels/resnet_v2_fp32_savedmodel_NHWC.tar.gz
		4	0.652	7472.4	INT8	
		8	1.234	8154.3	INT8	
	ONNX	1	0.297	4672.3	INT8	https://github.com/onnx/models/blob/master/vision/classification/resnet/model/resnet50-v2-7.onnx
		4	0.662	7398.5	INT8	
		8	1.259	8058.1	INT8	
resnet101 V1 224,224	ONNX	1	0.301	4579.0	INT8	https://s3.amazonaws.com/onnx-model-zoo/resnet/resnet101v1/resnet101v1.onnx
		4	0.604	8213.1	INT8	
		8	1.098	9169.6	INT8	
		10	1.286	9799.9	INT8	
resnet101 V2 224,224	ONNX	1	0.445	2766.0	INT8	https://github.com/onnx/models/blob/master/vision/classification/resnet/model/resnet101-v2-7.onnx
		4	1.119	4154.5	INT8	
		8	2.059	4513.1	INT8	
		10	2.475	4596.7	INT8	
resnet152 v1 224,224	TF	1	0.659	1759.4	INT8	open source model
		4	0.901	4949.3	INT8	
		8	1.495	6614.2	INT8	
		10	1.677	7095.3	INT8	
	ONNX	1	0.458	2644.6	INT8	https://s3.amazonaws.com/onnx-model-zoo/resnet/resnet152v1/resnet152v1.onnx
		4	0.823	5742.3	INT8	
		8	1.519	6419.7	INT8	
		10	1.717	6856.3	INT8	
resnet152 V1 slim 224,224	TF	1	0.663	1757.4	INT8	http://download.tensorflow.org/models/resnet_v1_152_2016_08_28.tar.gz
		4	0.903	4935.4	INT8	
		8	1.502	6625.5	INT8	
		10	1.712	7096.0	INT8	

resnet152 V2 224,224		ONNX	1	0.63	1939.2	INT8	https://github.com/onnx/models/blob/master/vision/classification/resnet/model/resnet152-v2-7.onnx
			4	1.634	2948.8	INT8	
			8	2.796	3208.2	INT8	
			10	3.346	3258.4	INT8	
ResNext50-32_4d 224,224		ONNX	1	0.349	3827.4	INT8	open source model
			4	0.783	6004.2	INT8	
			8	1.475	6522.6	INT8	
			10	1.766	6655.6	INT8	
resnext101_32_4d 224,224		ONNX	1	0.409	3065.6	INT8	open source model
			4	1.001	4614.8	INT8	
			8	1.874	5092.8	INT8	
			10	2.116	5433.4	INT8	
tiny yolo v2	1088, 1920	Pytorch	1	2.073	533.6	INT8	https://github.com/marvis/pytorch-yolo2
	320, 320		1	0.169	10971.3	INT8	https://github.com/marvis/pytorch-yolo2
	416, 416		1	0.227	8117.5	INT8	https://github.com/marvis/pytorch-yolo2
	608, 608		1	0.414	3968.6	INT8	https://github.com/marvis/pytorch-yolo2
	960, 960		1	0.849	1796.7	INT8	https://github.com/marvis/pytorch-yolo2
Yolo V2	1088, 1920	Pytorch	1	6.359	162.9	INT8	https://github.com/marvis/pytorch-yolo2
	320, 320		1	0.561	2226.6	INT8	https://github.com/marvis/pytorch-yolo2
	416, 416		1	0.713	1688.5	INT8	https://github.com/marvis/pytorch-yolo2
	544, 736		1	1.411	816.7	INT8	https://github.com/marvis/pytorch-yolo2
	608, 608		1	1.391	880.2	INT8	https://github.com/marvis/pytorch-yolo2
	960, 960		1	3.733	291.7	INT8	https://github.com/marvis/pytorch-yolo2
yolo v3	416, 416	Pytorch	1	1.111	1102.2	INT8	https://github.com/marvis/pytorch-yolo3
	960, 960	Pytorch	4	11.375	360.9	INT8	https://github.com/marvis/pytorch-yolo3

BERT squad BASE max sequnce length = 128	MX	1	2.809	404.0	INT8	https://gluon-nlp.mxnet.io/model_zoo/bert/index.htm
		4	2.798	1611.4	INT8	
		8	2.816	2226.2	INT8	
		10	3.191	2477.7	INT8	
		10	5.762	1726.4	INT16	
		12	3.59	2773.8	INT8	
bert mrpc BASE max sequnce length = 128	MX	1	2.822	402.8	INT8	
		4	2.762	1605.9	INT8	
		8	2.83	2211.6	INT8	
		10	3.196	2455.4	INT8	
		10	5.777	1722.6	INT16	
		12	6.877	1762.8	INT16	
bvlc_googlenet 224,224	ONNX	1	0.278	5113.7	INT8	https://github.com/onnx/models/blob/master/vision/classification/inception_and_googlenet/googlenet/model/googlenet-7.onnx
googlenet_bn_no_lrn 224,224	ONNX	1	0.141	12787.4	INT8	Developed inhouse based on Googlenet with batch norm and w/o LRN
		4	0.345	17558.0	INT8	
		8	0.592	18424.7	INT8	
		10	0.715	18530.5	INT8	
Squeezenet 1.1 224,224	ONNX	1	0.107	23281.0	INT8	https://github.com/onnx/models/blob/master/vision/classification/squeezenet/model/squeezenet1.1-7.onnx
		4	0.209	36321.4	INT8	
		8	0.373	37740.1	INT8	
		10	0.46	37843.9	INT8	
ssd-vgg16 300,300	MX	1	0.833	1466.1	INT8	http://github.com/zhreshold/mxnet-ssd/blob/master/symbol/legacy_vgg16_ssd_300.py

Table 1: GOYA Inference Performance Benchmarks

5.1 Resnet-50

Resnet-50 is a convolutional neural network that is trained on the ImageNet database and classify images into 1000 object categories, it is considered as one of the most popular application benchmarks for image classification.

The Goya PCIe card provides measured throughput of 15,488 images/second for the ResNet-50 workload at an extreme low latency of 0.82 msec, well below the industry requirement of 7 msec. Moreover, as can be shown in the chart below, the Goya's performance is sustainable at a small batch size, allowing it to support real-time applications with high throughput, the scaling in batch size enable Goya's customers to utilize the same processor for multiple use cases, switching from latency to throughput driven tasks. This economy enables Goya's datacenter customers reducing their TCO and expanses.

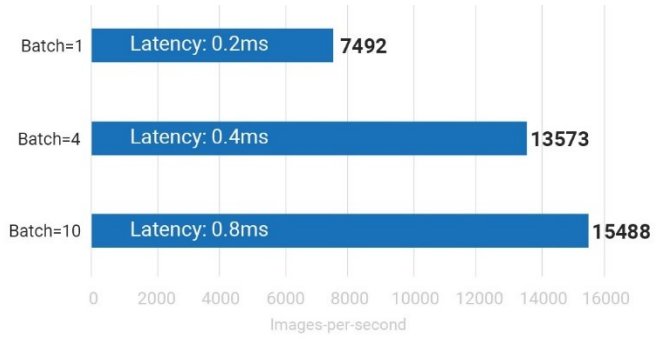

Figure 3 - GOYA ResNet50 Throughput and Latency

Software Configuration: Ubuntu v-18.04, SynapseAI v-0.11.447
Hardware Configuration: Goya HL-100 PCIe card, Host: Xeon Gold 6152@2.10Ghz

5.2 Googlenet

The Googlenet topology was developed in 2014, opting to use Local Response Normalization (LRN) over Batch Normalization. LRN is more computationally expensive. We have improved the throughput when replacing LRN with BN, which also results in improved accuracy, 72% top1 versus 68% originally. As this replacement reduces the amount of computation while improving accuracy, we expect it to perform better on any inference processor. Therefore, we are offering this improved topology (Googlenet_bn_no_lrn) on demand. Other than replacing LRN with BN, the topology has exactly the same structure.

5.3 Goya Performance in the ML-Perf v0.5 benchmark

The MLPerf was founded in 2018 by researchers from Baidu, Google, Harvard University, Stanford University, and the University of California Berkeley and becomes today the de-facto benchmarking tool for AI training and inference performance of ML hardware and software.

MLPerf launched the Inference benchmark suite on June 2019 and published the first Inference MLPerf benchmark results in November 2019 (https://mlperf.org/inference-results/).

MLPerf splits results into two divisions, closed and open. The Closed division allows for solution comparisons, adhering to an explicit set of rules; the open division allows vendors to more favorably showcase their solution(s) without the restrictive rules of the closed. Customers evaluating these results are also provided additional categories to help them discern which solutions are mature and commercially available (“available” category) vs ”preview” or ”research,” categories, which include hardware/software either not yet publicly available or experimental/for research purposes. Additional data is provided for the specific hardware (processors and systems) used in running the benchmarks, including details of the number of accelerators per solution evaluated, and the complexity of the host system used to measure the inferencing results (CPU generation, air-cooling vs. liquid-cooling) and more.

This is an essential industry initiative that will truly help customers determine which solutions are actually available for them to deploy (as opposed to just “preview”), and while the report is incomplete in scope - not yet measuring power, for instance - it is a critical first step toward the industry providing standardized, valid measures.

Following are the Goya MLpPerf Inference v0.5 results using a single Goya card, passive air cooling on an affordable host, an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, delivers the following results, in the Available category.

MLPerf Inference 0.5 Results

Benchmark	Dataset	Division	Status	Single Stream [msec]	Multi Stream [SPQ]	Server [QPS]	Offline [Throughput]
ResNet-50 v1.5	ImageNet	Closed	Available	0.24	700	13090*	14451
SSD-Large	COCO-1200X1200	Closed	Available	3.85	18	296.7	326.3

SPQ = Samples per Query @ 99% tail latency
QPS = Queries per Second @ 99% tail latency
(*) ResNet-50 v1.5 Server scenario result not verified by MLPerf

Figure 4 - MLPerf Inference v0.5 Results

In addition to the above results in the closed division, Habana has also contributed results that show Goya's superior throughput under latency constraints, which benefits real-time applications. Such tests are available in the Open division:

contribution follows the closed submission rules with only one change – more strict latency constraints for Multi-Stream scenario.

- ResNet-50: Goya delivers 20 Samples-Per-Query (SPQ) under latency constraint of 2ms and 40 SPQ under 3.3ms latency constraint. Thus, Goya is up to 25 times faster than the required latency for the closed division (50ms).
- SSD-large: Goya delivers 4 SPQ under latency constraint of 16.8ms and 8 SPQ under 30.8ms latency constraint, up to 4 times faster than the required latency for closed division (66ms)

MLPERF v0.5 includes the well-established vision benchmarks such as ResNet-50 and SSD-large, but it has not yet included BERT, which has achieved state-of-the-art results on many language tasks, and as such, is very popular among cloud service providers. Please refer to our BERT results in the next section.

5.4 BERT

One of challenges implementing natural language processing (NLP) applications is the cost of training resources. Many NLP (natural language processing) use cases require large amounts of data in ordered to be trained. In order to reuse and save training compute, time and costs, researchers have developed a pre-training method, training general purpose language representation models, based on enormous amount of unannotated text. This pre-trained model can then be fine-tuned using small amounts of data and with an additional, lean, output layer create state-of-the-art models for a wide range of tasks like question answering, text summation, text classification (e.g. spam or not) and sentiment analysis, resulting also in substantial accuracy improvement, compared to training on these datasets from scratch.

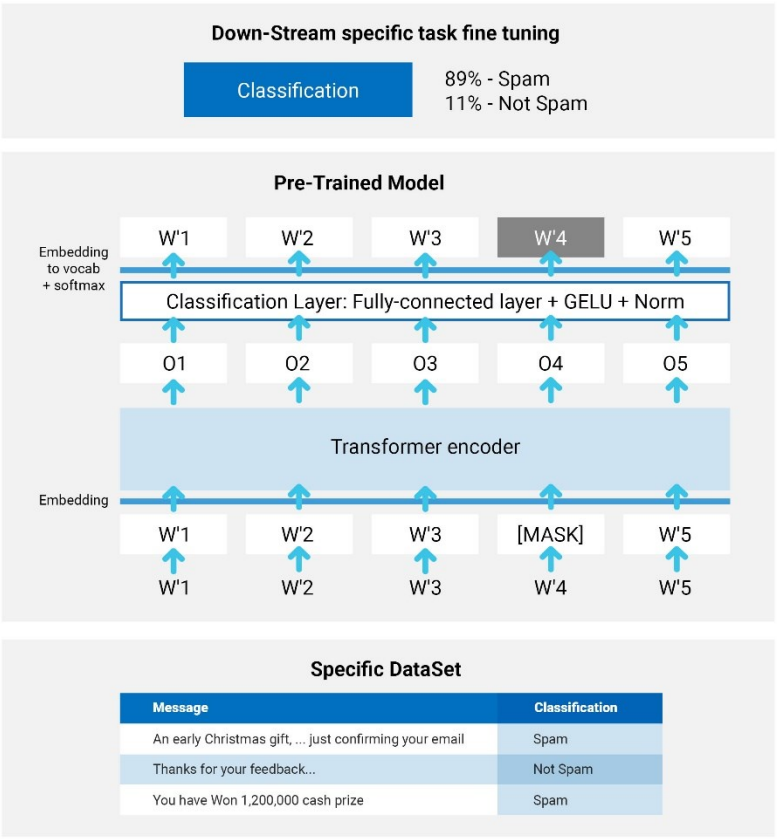

Figure 5 - BERT Two Phases: Pre-Trained Model and Specific Task Fine-Tuning

The approach pioneered by BERT constitutes a paradigm shift in language modelling using deep learning models, by offering a strong representation that can be used in transfer-learning new tasks. New models and techniques based on the BERT architecture have quickly taken their place as the de-facto standard in text understanding. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. It consists of multiple, multi-head self-attention layers, that leverage a bidirectional context conditioning on the input text, allowing the model to learn the context of a word based on all its surroundings (beginning and end of each sentence).

5.4.1. BERT Inference Using GOYA

The GOYA inference architecture delivers exceptional inference performance for the BERT workloads. All of BERT operators are natively supported by the Goya Inference processor and can be executed on Goya without any host intervention. A mixed precision implementation is deployed, using Habana's quantization tools which set the required precision per operator to maximize performance while maintaining accuracy.
BERT is amenable to quantization, with either zero or negligible accuracy loss, while GEMM operations are execute at INT8, some other operations, like Layer Normalization, are done in FP32.

Goya's heterogenous architecture is an ideal match to the BERT workload, as both the GEMM engine and the TPCs are fully utilized, supporting low batch sizes at high throughput. Goya's TPC provides significant speedup when calculating BERT's nonlinear functions resulting in leading benchmark results. In addition, the Goya's software-managed SRAM allows increased efficiency between different memory hierarchies, while executing.
Below are performance results measured on Goya, for a question answering task, identifying the answer to the input question within the paragraph, based on Stanford Question Answering Database (SQuAD). The tested topologies are BERTBase, Layers=12 Hidden Size=768 Heads=12 Intermediate Size=3,072 Max Seq Len = 128

To quantify and benchmark this task, we used an off the shelf available model - https://gluon-nlp.mxnet.io/model_zoo/bert/index.html

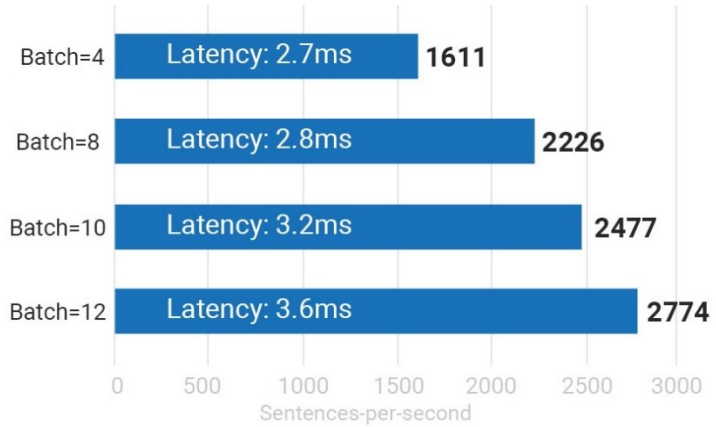

Figure 6 - BERT- Base SQuAD Inference Benchmark

Hardware: 1x Goya HL-100; CPU Xeon Gold 6152@2.1 GHz; Software: Ubuntu v-18.04;
Software: SynapseAI v-0.11.0-447;

The Goya processor delivers high throughput on the SQuAD task at extreme low latency. As the BERT model is foundational to many NLP applications, we expect similar inference speedups for other NLP applications based on it with their own fined tune layers. Moreover, the Goya architecture is scalable and keep its high efficiency while increasing the batch size.

6. Summary

Deep learning revolutionizes computing, impacting enterprises and the services and experiences they can deliver across multiple industrial and consumer sectors. Deep neural networks are becoming exponentially larger and more complex, driving massive computing demands and costs. Modern neural networks are too compute-intensive for traditional CPUs, and even GPUs. The future belongs to dedicated high throughput, low latency, low power AI processors.

Inference performance is measured by throughput, power efficiency, latency and accuracy, all of which are critical to delivering both data center efficiency and great user experiences. The **Goya is a world class, high performance, AI processor for inference workloads.** It is supported by state-of-art software development tools and a full software stack.

An effective deep learning platform must have the following characteristics:

• A processor that is custom-built for deep learning
• That's software-programmable

Habana Labs' Goya AI Inference Platform meets all these requirements.