Summary of In-Datacenter Performance Analysis of a Tensor Processing Unit

Qiyue Sun (jerrysun), Tianrong Zhang (ztr)

Problem and Motivation

The problem this paper tried to address was that the hardware at that time was not suitable for the inference jobs of DNN models. And there are two aspects:

- The number of weights in the DNN models was so large that it was quite **expensive to use conventional CPUs** to do the inference. One example showed in this paper was that "people searching by voice for three minutes a day using speech recognition DNNs would double the datacenters' computation demands."
- Many of the DNN applications are "parts of end-user-facing services," which leads to a **rigid response-time limit for the inference**. Given the latency limits, hardware at that time was not suitable because both CPU and GPU "help average throughput more than guaranteed latency."

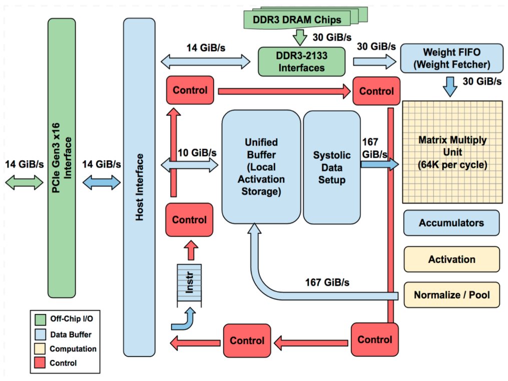
Given the above problems, the authors tried to develop a custom ASIC that is especially quick for DNN inference. In the end, the new hardware can improve the cost-performance compared with GPU and CPU. The major focus of this paper is on the comparison between TPU, CPU, and GPU with respect to latency, throughput, energy performance, etc.

Hypothesis

- Their observations are based on the workload in their datacenters, which may not be the same for other companies. For example, when choosing the benchmarks, they choose three types of NN (MLPs, CNNs, and LSTMs) that represent 95% of the NN inference workload in their datacenters.
- The Roofline Performance model is used in this paper to illustrate the performance of the benchmarks. The assumption behind the model is that applications don't fit in on-chip caches, so they are either computation-limited or memory bandwidth-limited.

Solution Overview

Matrix Multiply Unit is the main computation unit in TPU. Matrix Multiply Unit consists of 256x256 MACs (multiply accumulation computation), which can "perform 8-bit multiply-and-adds on signed or unsigned integers." The output of the Matrix Multiply Unit is collected in the ***Accumulators***, which are later used by the ***Activation Pipeline*** to perform the non-linear functions. The weights for the matrix multiplication are staged through an on-chip Weight FIFO that reads from an off-chip ***Weight Memory***. The intermediate results are held in the on-chip ***Unified Buffer***, which can serve as inputs to the Matrix Multiply Unit.



The instruction set is quite small, and in this paper five key instructions are mentioned, including reading data from CPU host memory (`Read_Host_Memory`), reading weights from weight memory (`Read_Weights`), performing matrix multiplication (`MatrixMultiply`), performing the non-linear activation function (`Activate`), and writing data to the host memory (`Write_Host_Memory`). Here are major 3 key insights of this design:

- The design is quite simple and regular compared with CPU or GPU. As said in this paper, "minimalism is a virtue of domain-specific processors." The datapath, including the Unified Buffer and Matrix Multiply Unit, is nearly two-thirds of the die, while control is only 2%, which reduces the complexity of the design. Besides, compared with CPU and GPU, TPU is single-threaded and does not have sophisticated microarchitectural features, such as branch prediction, Out-of-order execution, etc. The major reason is that TPU is designed to focus on one job (matrix multiplication and convolution) and improve the 99th-percentile case, while CPU and GPU (especially CPU) are designed to deal with a lot of different situations and improve the average case.
- According to the paper, "the philosophy of the TPU microarchitecture is to keep the Matrix Multiply Unit busy." The 4-stage pipeline of the CISC instructions hides the execution of other instructions by overlapping with the MatrixMultiply instruction. For example, Read_Weights can finish execution before the weights are actually read from the memory. Once the address is resolved, Read_Weights can complete.
- Since reading a large SRAM consumes a lot of power, the Matrix Multiply Unit is designed to use systolic execution in order to save energy by reducing reads and writes of the Unified Buffer. The basic idea of systolic execution is to let the data go through multiple processing elements (PE) at regular intervals before writing back to memory. Thus, systolic execution can balance the fast computation with the slow memory I/O. Also, because the Matrix Multiply Unit in TPU consists of 512x512 PE, systolic execution also exploits the concurrency very well.

Limitations and Possible Improvements

The Matrix Multiple Unit is designed for dense matrices. The support for sparse matrices are omitted for time-to-deployment reasons. One possible improvement is to make the TPU compatible with sparse matrices, which can be found in the paper Sparse-TPU: Adapting Systolic Arrays for Sparse Matrices.

Summary of Class Discussion

The speed gap between computation units and memory bandwidth has been the bottleneck for various hardware designs. Many techniques are developed to try to migrate this gap. One example is the systolic execution (or systolic array) used by TPU. Also, with the quick development of NN fields and the rise of domain-specific processors, there are increasingly new requirements for the hardware, which can potentially make the hardware manufacture difficult to catch up with. But some fundamental principles in hardware design remain the same (such as balancing computation and I/O).

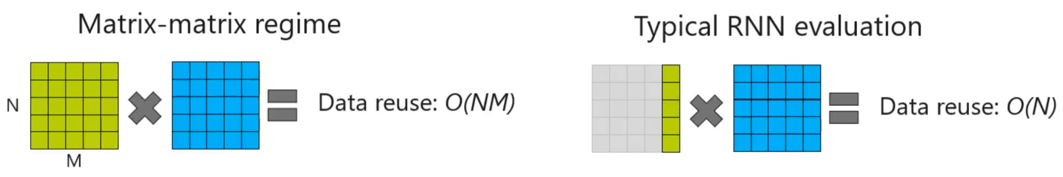
Summary of "Serving DNNs in Real Time at Datacenter Scale with Project Brainwave"

Tianrong Zhang (ztr)

Problem and Motivation

Growing demand for real-time AI serving calls for cloud services with ultra low inference latency and the solutions based on GPGPU or hard NPU suffers from the following drawbacks:

- Rely heavily on data reuse, for example represent operations like 2D convolution with matrix multiplication, which are less abundant in sequential network architecture like RNNs and that is known to hurt accelerator-CPU and computation/data ratio.

  ▲ Illustration provided in ISCA 2018 Lightning Talk: Project Brainwave

  - Another more important example of data reuse in model serving.

    Reliant on large batch sizes, while in inferencing at serve-time, batching means asynchronous request answering which largely increases latency. So the batch is often kept very small (Batch-1 situation is of most interest and investigation) but that leads to similar degradation as in the previous point.

    Walkarounds like reshaping input from vector to matrix were proposed but the authors claimed they didn't improve performances.

- Non-extensible ISA which doesn't suit the emerging new techniques proposed in the AI field. The hardware needs new design to be optimized for the new operations.

  Also calls for a fast cooperative computing device complex such that more flexible devices like CPU can hop in when certain operations are not available.

- Non-configurable or limited independent memory. In inferencing, the weight matrices are fixed and pinning them on chip (i.e. store in accelerator) is desirable but not possible due to poor scalability.

While FPGA features low latency and programmability which solves all the abovementioned problems at a time, Microsoft created Brainwave that instantiate an FPGA based solution.

Hypothesis

For FPGAs to overcome the problems GPGPUs and hard NPUs are facing,

- NFUs (neural function units) in NPU (neural processing unit) highlights matrix-vector multiplication.
- FPGAs are connected in the datacenter network through hierarchy of ToR (top of rack switches) such that when the model cannot fit into a single FPGA, more FPGAs can be used at a negligible increase in latency.
- NPU has relatively large on-chip memory and each NFU has independent memory to store the model parameters.

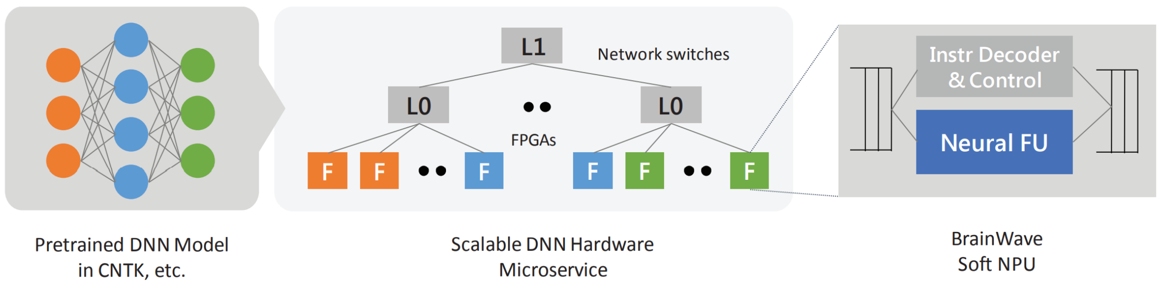
Another idea in the ecosystem but doesn't directly come to solve the previously discussed problems is to proposed a different data type that targets DNN specifically which preserves accuracy better than other short data types.

Solution Overview

The Project Brainwave comprises of

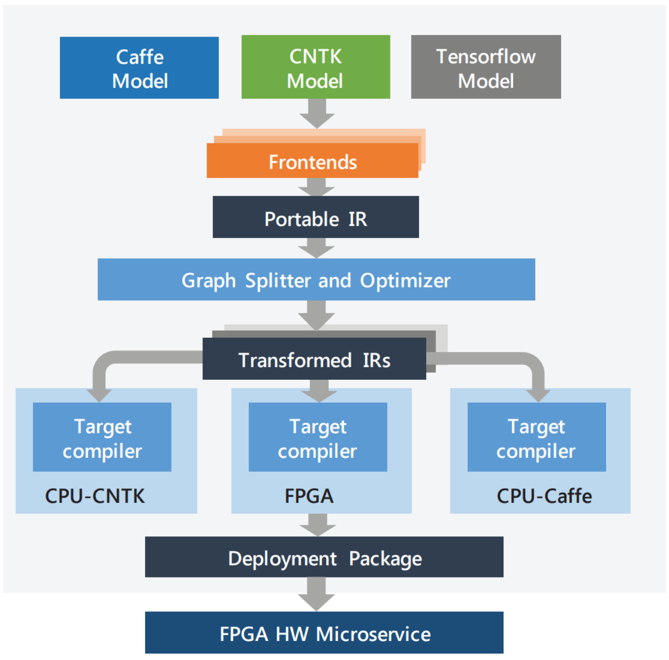
- Compilation workflow that maps data flow graph to device-specific instructions.
- Federated runtime plus scheduler for integrating the combined use of devices. This creates a microservice and open the entry point to subscribing devices.
- Underlying NPU architecture in FPGA.

Corresponding to the so-called 3-layer architecture



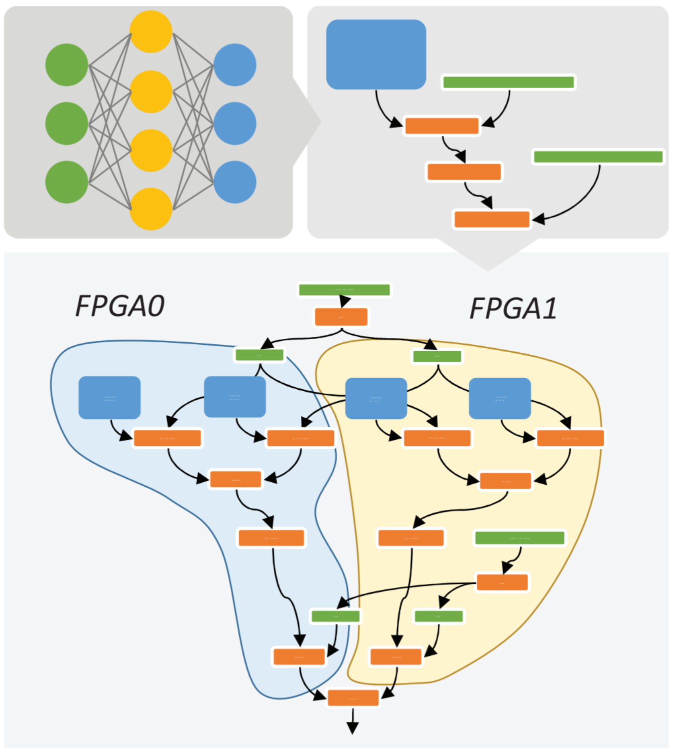
Compilation & Runtime

In my personal view. This part is of less importance in terms of exploiting the merits of FPGA. It is more of describing a higher level overview of the workflow and highlighting to users that no knowledge about FPGAs are required to use Brainwave.



Up until "Transformed IRs" in the diagram above are similar to conventional graph splitting and optimizing process. The major difference is that the splitter not only split the graph to fit the device capacity; it also decides which backend device is responsible for a subgraph based on FPGA's supported operations. CPU serves as catch-all but according to the authors, "over time however, nearly all performance-critical operators will be served on FPGAs."

When a subgraph is too large for a single FPGA, the author gives the following example



This describes splitting a subgraph depending on common input data into 2 FPGAs when the independent input activation vector is too large. It not clear what if the weight matrix is too large but the intuitive thing to do is to split the operation like in tiled matrix multiplication.

There is not much valuable information about the federated runtime in the paper beyond repeating its desired behavior.

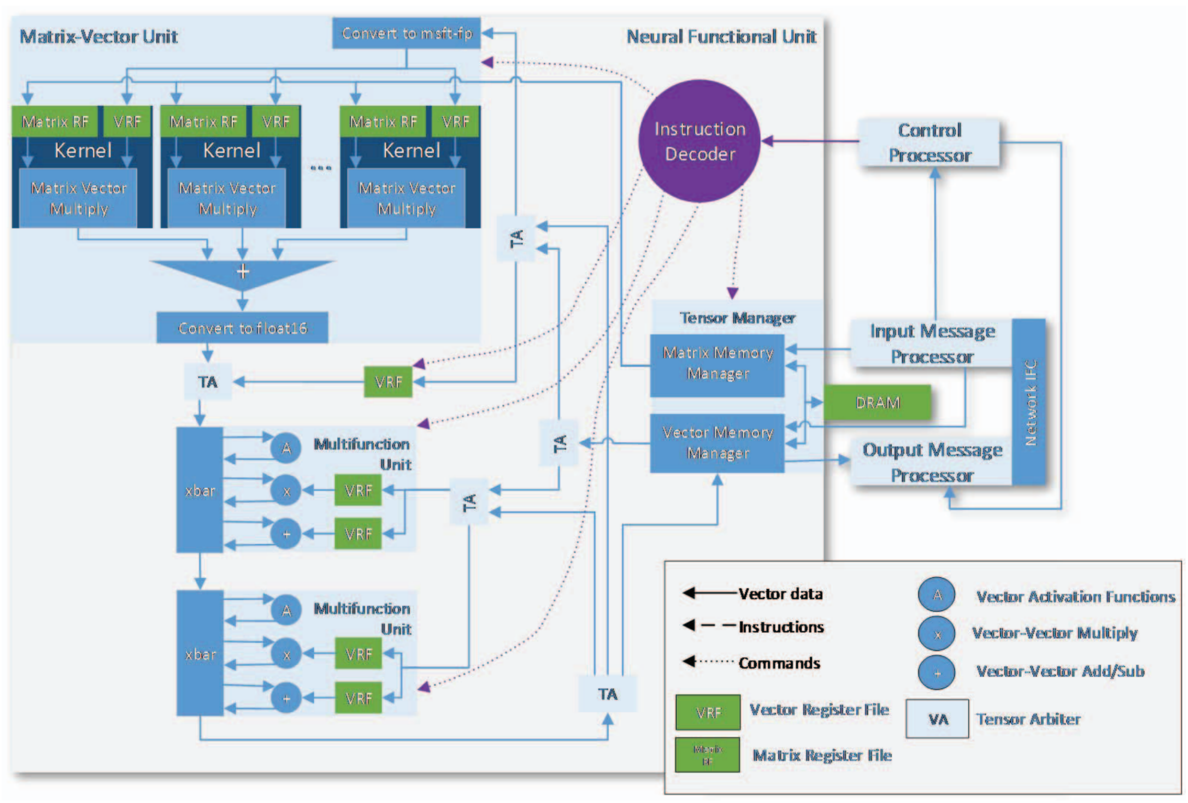
Soft ISA

To include custom logics, the NPU support custom CISC instructions though the Nios embedded chip serving as a sequential control processer.

According to [Nios II Custom Instruction User Guide] from Intel, it can instantiate custom logic bounded to the ALU (NFU here I guess). The custom logic is specified by the type of the instruction and the corresponding ports. The author didn't provide more details but I think chances are that Microsoft provides a tool that compiles a sequential program into an optimized custom logic and rely on Intel Platform Designer to instantiate the hardware.

The is claimed to be a compromise proposal between synthesizing the entire logic for an instruction using fixed ISA and designing complicated RTL (register transfer language) for all custom instructions.

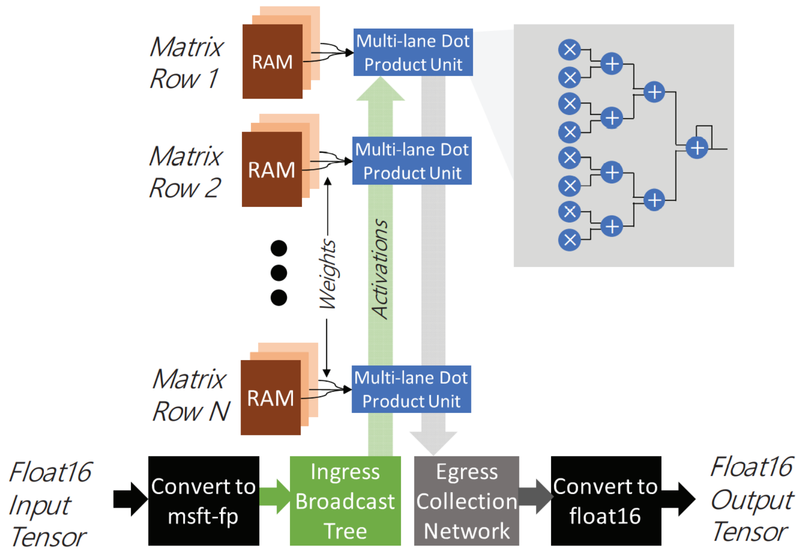
NPU



The paper comes with a strangely low-resolution diagram of the NPU with typos in the legend calling "Tensor Arbiter" "VA" instead of "TA". The arbiter is never mentioned in the paper anyway. It should be a mux controlled by the instruction decoder to decide which of the input data is selected.

This diagram is rather brief about the memory interaction but gives more details about the functioning unites majorly comprised of Matrix-Vector multiplication (MAV) units and multifunctional unit (for Vector-Vector operations or non-linear activations).

The idea of Matrix-Vector multiplication is actually just a tiled multiplication with tile size of 1, i.e. dot product. The dot product is computed with the MAC (multiply and accumulate) operations which is known to reduce the error introduced by rounding product before addition.



The above is a closer look at the MAV unit. This is also where Microsoft's proprietary data representation comes into play.

Narrow Precision

The authors claimed that their proposed ms-fp8 and ms-fp9.

- Floating point numbers

  The paper says nothing about the details of the representation. We can only deduce that comparing to IEEE mini-float (8-bit), one bit of mantissa is replaced with an extra bit for exponentials. The convers a wider range than mini-float at a cost lost less significands.

- Integers

  By introducing floating point, the number has a larger range and a dynamic quantization step-size.

The results shows a small loss in accuracy when reducing precision and a short retrain suffice to recover the performances. The comparison with integers reveals a increase in flops.

Limitations and Possible Improvements

Much of the gate arrays are reserved for storing data and the register filters are speeded throughout the array. This can limit the space for implementing logic. The so-called custom instruction set still calls for heavy programming, especially given that no much details are presented and if there is a simple translation scheme, Intel would have provided themselves. Without official libraries, a closed project like this is less likely to truly develop good implementations for new operations in AI. This project was released in 2018 and almost no further updates can be found on the internet while TPU has evolved to third generation. Chances are that though Microsoft claims the system is easy to use but it is actually not. For example when using ms-fp, the author mentioned retraining to recover performances while Brainwave is incapable of training. Other accelerator networks that enables ms-fp is needed to support Brainwave.

Summary of Class Discussion

Doubts

- Why is reshaping not improving performance? The authors made this claim without any reference.
- The amazingly good preservation of accuracy when using ms-fp seems dubious. No direct comparison with float8 is provided; all experiments are performed over RNNs; the baseline accuracy is 1 which means the model can be too simplified; in model 3, the accuracy after retraining exceeds 1.
- Fixed point arithmetic is often considered the fastest. How possible is floating point or same and even larger width induce higher flops? The floating point representation itself have nothing to do with parallelism.

Questions

- How is the sequential controller implemented? How good is it compared with RTLs?
- Each FPGA can be cheaper than a GPU, but in Brainwave, massive number of FPGAs are required. The on-chip memory is at most 30MB per FPGA, if a large model is pinned to the FPGA, it is going to need a lot of FPGAs. It's said that each server in the datacenter is equipped with a FPGA (often used for networking according to other FPGA enabled cloud services like in Tencent) so is the FPGA-CPU ratio too imbalanced?