Machine Learning at the Edge Using Neural Network Processor

Edwin Lee
Raytheon Technology
Hanover, MD, United States
edwin.lee@ rtx.com

Michael Parker
Raytheon Technology
El Segundo, CA, United States
michael.a.parker@ rtx.com

Michael Cervantes
Raytheon Technology
Tewksbury, MA, United States
macervantes@rtx.com

Ben Plotner
Raytheon Technology
Hanover, MD, United States
Benjamin.F.Plotner@rtx.com

***Abstract*—Increasingly, power, space, and cooling constrained embedded computing assets need to run machine learning applications to make autonomous, low latency decisions within operational environments. Where GPUs, CPUs, and FPGAs aren't suitable for these constraints, custom ASICs with their non-standard development processes and frameworks are the best available option for bringing the power of machine learning to the edge. This paper discusses a neural network processing architecture and why it surpasses all size, power, throughput, and weight requirements while maintaining performance and allowing interoperability with commercial vendors license comprehensive development environment and machine learning libraries. We compare this neural network processor architecture with existing AI and machine learning platforms that leverage common architectures such as GPUs, CPUs, and FPGAs. The plug and play nature of the neural network processor architecture is optimized for applications such as EO-IR (Electro-optical/Infrared) Sensor and Radar Systems without respinning the hardware, reducing costs. This paper explores best practices for deploying common AI and machine learning models for object detection, super resolution, and natural language processing on a neural network processor at the edge. The neural network processor offer an alternative for deployment of autonomous machine learning at the edge that brings the power of a robust development ecosystem together with an architecture favorable for power and space constrained use cases.**

Keywords—Neural Network Processor, Small Form Factor, ONNX, Natural Language Processing, RF Embedded Systems

I. Introduction

Machine learning, including deep learning, applications have gained traction in both the military and commercial sectors in recent years, enabling machine learning in tactical environments. Models for these environments are trained on large datasets, leveraging commodity computing environments in preparation to be deployed on data closer to the sensor. To apply models over data in real time in increasingly tactical environments, the processor close to the sensor itself must be small, light, and have a low power footprint. Standard CPU (Central Processing Unit) and GPU (Graphics Processing Unit) based-systems do not support these requirements. To get around this FPGA (Field Programmable Gate Arrays) and ASIC (Application-Specific Integrated Circuit) designers customize processors to perform machine learning applications suited for constrained power, space, and weight requirements. However, long development and manufacturing time on these custom hardware chips prevents companies from deploying algorithms for many mission critical applications. ASIC and FPGA developers are restricted to programming in RTL language without the benefit of a comprehensive development environment that provides well-tested and robust library support for key aspects of algorithm implementation. This lack of support dramatically increases development and compilation time as well as the burden of proper validation and testing of each custom component.

Employing neural network processors to support these machine learning applications near the sensor combines a plug-and-play architecture on an ASIC while supporting the small form factor and power footprint constraints of these operational settings. This brings the capability of demanding commercial applications such as smart phones, smart cameras, and autonomous driving systems to these processors running machine learning models at the sensor. Several commercial vendors license comprehensive development environments and machine learning libraries as well as graph support, integrated using standard interfaces all operable with the C programming language.

In addition, the licensable, deployment-ready neural network processor outputs dramatically more TOPS (Tera Operations Per Second) processing power per clock cycle under 1W as compared to standard CPU/GPU/FPGA systems by an order of magnitude given the same semiconductor manufacturing process. The advent of this technology streamlines development cost when enabling machine learning applications at the edge without materially impacting the power, space, and weight burden.

II. Compare Neural Network Processors to Known Processing Units

To optimize performance on a general-purpose CPU for neural network processing, the SIMD (Single Instruction Multiple Data) hardware design allows for a single instruction to process multiple data inputs within one instruction clock cycle. While the amount of data the SIMD hardware can process per clock cycle is large, there is a limit, and standard neural network applications require several instruction clock cycles. This increases processing time since neural network applications tend to ingest large amounts of noisy data.

General purpose GPUs enable massively parallel data processing because the architecture leverages multiple processing cores, thus solving the latency problems in sequential processing in CPUs. In GPUs, a large portion of the processing cores used in parallel computing applications are floating point processing cores. This quickly escalates the space and power consumption for standard neural network applications. Thus,

edge computing assets covered in this paper often cannot sustain the size and power requirements for GPU-enabled neural network applications.

Neural network processors use one instruction in parallel across many simplified operation units, also known as MACs (Multiple- Accumulates), to process high volumes of data with a single instruction, bringing together the benefits of the parallel processing of the GPU and the simplified single instruction framework of the CPU. The neural network processor's operation units have closely coupled memory to evaluate the neural network weights adjacent to lower the latency in comparison to the centralized memory of the GPU and CPU. It also optimizes the multiplication and addition operations, reducing the need for floating point operations to reduce size and power consumption.

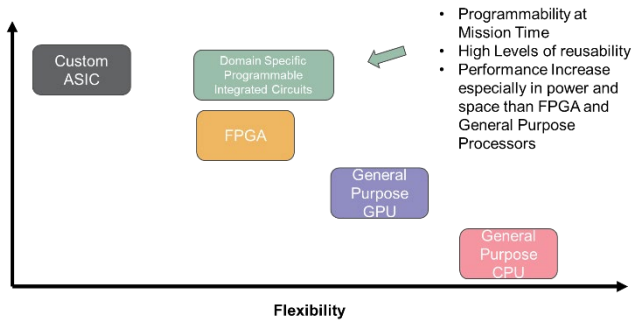

Fig. 1. Domain-specific programmable integrated circuits performance and flexibility compared to other microelectronics approach

Domain-specific programmable ASICs, such as the neural network processing unit discussed here are able to process more data per Watt than CPUs and GPUs. However, by definition, ASICs have a less flexible architecture, restricting their usability across use cases beyond neural network processing. This limited flexibility can cause delayed development cycles due to the high-customized nature of firmware design needed to tailor the hardware to the use case. Neural network processors from certain vendors enable firmware integration into widely available machine learning development and deployment frameworks.

III. Compare Neural Network Processor Across Vendors

As noted in the previous section, most neural network processors have favorable processing performance as measured in operations per Watt as compared to GPU and CPU processors. Among neural network processors, there is variation in feature integration and performance across vendors.

This research first suggests three common neural network models against which to benchmark the performance of various neural network processors, each testing different aspects of the processor's capability. The performance of these models can be measured in frames per second. Object detection models such as the Yolo-v3 (You Only Live Once) model test the capacity of the processor against standard convolution neural network problems– especially the ability of the processor to handle high volumes of multiplication and addition operations as well as contiguous memory burst transfers [7].

Testing various neural network processors against the BERT large (Bidirectional Encoder Representations from Transformers) natural language processing model demonstrates the ability of the processor to efficiently compute complex operations from the Scaled Dot-Product Attention shown in (1) such as square root and matrix transpose operations [1] [6]. These operations are not leveraged in the conventional convolution networks.

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

EDSR (Enhanced Deep Super-Resolution) models provide a helpful benchmark because these models have much greater computational and memory cost than the previous models [5]. Moreover, super resolution networks have proven to aid object detection model accuracy, making a good candidate to test on the neural network processor.

The diversity of use cases helps determine if the neural network processor performs consistently across the common variation in neural network use cases for maximum generalization. For example, if the processor does not have an agile memory bus controller, we may see that a processor tends to perform well in convolution neural network models but poorly in transformer-intensive models such as those used in natural language processing.

TABLE I. Example Networks to Benchmark Neural Network Processors

Networks	Type	Image Size	Reason to Use
Yolov3	Obj Detection	416x416x3	Normal Capability
BERT	Transformer	384x1x1	Processing and Memory Flexibility
EDSR	Super Res	256x256x3	Computational and Storage Intense

The output accuracy of neural network processors across model types is another component to consider when comparing neural network processors. Most neural network processor compilers will analyze the network model upon compilation and optimize the network by erasing a layer in a network it deems doesn't affect the accuracy of the result significantly. This is also known as pruning [3]. Pruning decreases the amount of operations and weights needed in the neural network, decreasing the amounts of memory and MACs required to run a model. However, pruning a network degrades the accuracy of models. A good neural network processor compiler will make a good tradeoff, pruning the network while maintaining the network model accuracy.

As mentioned before, floating point operations often increase the size and power consumption of the processor. Therefore, it is critical to convert weights, bias, and activations values of a neural network from floating point into lower precision fixed point [4]. This method is known as quantization,

and the processor's ability to attempt these conversions without jeopardizing the networks accuracy demonstrates how well a neural network processor compiler performs.

Some processor also uses the Winograd Transform to decrease the number of operations needed for the convolution operation. Some compilers will replace convolution operations with the Winograd Transform which creates a less accurate solution when pruning the model. Because of these cases, one should always look at the accuracy of the results and not just the performance in frames per second or completion time of the neural network processor.

A neural network processor should also support multiple fixed-point precision and floating-point precision data processing. The more data types that the processor supports, the more range of performance and accuracy is available. For instance, if a processor has eight-bit fixed point precision, fewer operations are required, but the models have poorer accuracy when compared to processors that allow for floating point precision. Comparing neural network processors across common model types, holding constant the compiler optimizations for the test case, as well as ensuring floating point precision capabilities, allows the system designers to ensure maximum performance in neural network use cases.

IV. Robust Development Ecosystem

One key benefit of using a domain specific processor like a neural network processor is that, because these processors are integrated in consumer devices elsewhere in industry at massive scale, there are robust development frameworks to integrate machine learning model software with this optimized hardware. The availability of these frameworks for some vendors can drive down development, deployment, and maintenance costs of the applications built on the processors.

Most machine learning algorithms are trained on a large server environment using common machine learning frameworks such as PyTorch and TensorFlow. Deployment of these machine learning algorithms on embedded processors require a developer to convert Python code to the C language and assembly language because Python is not very optimized for embedded platform applications. In present day, neural network processor vendors translate libraries in common machine learning frameworks such as Pytorch and Tensorflow into their proprietary vendor's assembly language. However, when new functions are added to the learning framework, the vendor may need to reoptimize their compiler to include this function.

To offload the work done to optimize new functions in hardware by the vendor, the Open Neural Network Exchange (ONNX) was formed to use an open standard to represent neural network models [2]. Almost all machine learning frameworks have a conversion from their framework to the ONNX standard for hardware optimization. By using optimizing neural network processor firmware based on the ONNX standard, the vendor can offload compiler development effort associated with evolving functionality in multiple common machine learning frameworks. With the ONNX framework, data scientists can focus more on algorithm development rather than hardware optimization on the processor, speeding up development and deployment of neural network processors and machine learning models for embedded processing at the edge. In an ideal scenario the ONNX framework middleware would bring the ease of porting a functional neural network model from standard server to the highly optimized embedded platforms.

V. Architecture Favorable for Power and Space Constrained Use Cases

Traditional use cases for embedded processing on the edge can center around RF processing for sensing the signal environment. The power, space, and cooling requirements for complex processing and decision making at the edge have commonly constrained the capabilities of the sensor. Deploying neural network processors at the edge has enabled new critical capabilities like spoofing and jamming detection, object tracking, and signal characterization. The advantages of neural network processors for these uses cases as discussed above, encourage the integration of these processors within the RF processing architecture.

Typically, RF systems take in the signal environment with a data converter that converts analog to the digital domains. These data converters have throughput that exceeds the capacity of the neural network processor. In FPGA and ASIC based designs, there is feature aggregation and preprocessing of this high throughput data. As discussed, these architectures are costly to develop and maintain with low flexibility. This paper proposes a COTS (Commercial Off The Shelf) vector processor to perform this preprocessing.

As shown on the architecture below in Fig 2., the vector DSP processors is placed between the data converter and the neural network processor to perform important ore-processing functions like AoA (Angle Of Arrival), beamforming, frequency domain processing, and decimation. Then, the neural network processor uses this processed signal from the vector DSP processor to perform detection, classification, identification and characterization functions on the edge.

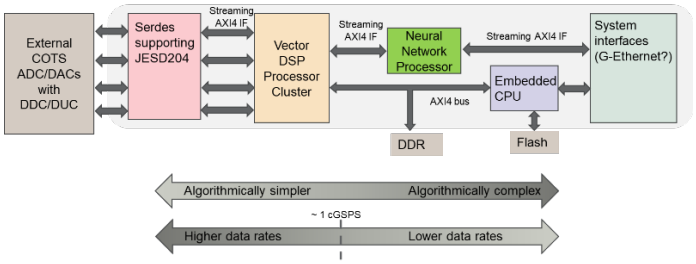

Fig. 2. Domain Specific Programmable Integrated Circuits performance and flexibility compared to other microelectronics approach

Vendors who provide neural network processors commonly have compatible vector DSP processors which reduces the need for additional trade studies on compatibility and capacity.

There are several major benefits to such an architecture. As the system is highly programmable, the same ASIC may be used for multiple programs and applications, allowing for rapid development and amortization of ASIC costs across multiple programs. The programmability also enables firmware upgrades after system deployment. As the ASIC is largely composes of licensable COTS processors and intellectual property, there is considerable flexibility regarding ITAR (International Traffic in

Arms Regulations) on ASIC fabrication choice, which can allow for the leading semiconductor processes to be used, maximizing the throughput per Watt.

VI. Conclusion

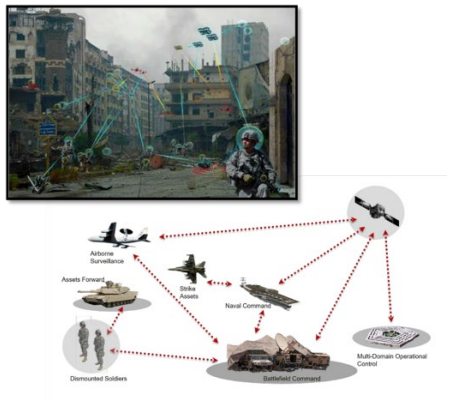

Fig. 3. Scenarios like these require flexible processing units to cover multiple tactical applications in a communication restricted environment

Moving processing power to the edge enables embedded assets, such as those in tactical military environments, to achieve higher degrees of autonomy, leveraging machine learning, in tactical environments where the communications environment is restricted. Further, military systems which are power and space constrained face limitations in both the processing power and the complexity of decision-making algorithms deployed in operational settings where this latency in communication is a factor, restricting the uses cases for autonomous systems in tactical environments to only those that are on very large and power intensive frameworks.

Various firmware-based architectures that are customized to narrow use cases have emerged to fill one-off gaps across the mission space, each with their own highly specialized staff required to maintain and customize the applications. This drives up the cost for implementing these capabilities in additional or extended mission space.

In military applications, the demand for increased complexity in algorithms processed on embedded assets at the tactical edge is expanding. The rapid development cycle needed to train and deploy models for new operational use cases doesn't fit inside the long, unique firmware development process typically used for computing assets in power and space constrained environments.

Neural network processors provide a unified and extensible architecture that can extend scalable use of machine learning models on constrained embedded platforms to accelerated deployment on systems like autonomous unmanned vehicles and guided missiles. This architecture enables faster high processing units on the edge, as well as a more flexible and agile design to rapidly deploy models on new assets.

References

[1] K. Han et al., "A Survey on Vision Transformer," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 1, pp. 87-110, 1 Jan. 2023, doi: 10.1109/TPAMI.2022.3152247.

[2] "Faster Scalable ML Model Deployment Using ONNX and Open Source Tools," 2020 IEEE Infrastructure Conference, San Francisco, CA, USA, 2020, pp. i-i, doi: 10.1109/IEEECONF47748.2020.9377615.

[3] S. Kim, J. Lee, S. Kang, J. Lee, W. Jo and H. -J. Yoo, "PNPU: An Energy-Efficient Deep-Neural-Network Learning Processor With Stochastic Coarse–Fine Level Weight Pruning and Adaptive Input/Output/Weight Zero Skipping," in IEEE Solid-State Circuits Letters, vol. 4, pp. 22-25, 2021, doi: 10.1109/LSSC.2020.3041497.

[4] K. Desappan, M. Mody, M. Mathew, P. Swami and P. Eppa, "CNN Inference: Dynamic and Predictive Quantization," 2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), Berlin, Germany, 2018, pp. 1-4, doi: 10.1109/ICCE-Berlin.2018.8576251.

[5] B. Lim, S. Son, H. Kim, S. Nah and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 2017, pp. 1132-1140, doi: 10.1109/CVPRW.2017.151.

[6] J. He, L. Zhao, H. Yang, M. Zhang and W. Li, "HSI-BERT: Hyperspectral Image Classification Using the Bidirectional Encoder Representation From Transformers," in IEEE Transactions on Geoscience and Remote Sensing, vol. 58, no. 1, pp. 165-178, Jan. 2020, doi: 10.1109/TGRS.2019.2934760.

[7] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.