Architectural Analysis of Deep Learning on Edge Accelerators

Luke Kljucaric, Alex Johnson, Alan D. George
Department of Electrical and Computer Engineering, University of Pittsburgh
NSF Center for Space, High-performance, and Resilient Computing (SHREC)
Pittsburgh, PA, USA
{luke.kljucaric, alex.johnson, alan.george}@nsf-shrec.org

***Abstract*—As computer architectures continue to integrate application-specific hardware, it is critical to understand the relative performance of devices for maximum app acceleration. The goal of benchmarking suites, such as MLPerf for analyzing machine-learning (ML) hardware performance, is to standardize a fair comparison of different hardware architectures. However, there are many apps that are not well represented by these standards that require different workloads, such as ML models and datasets, to achieve similar goals. Additionally, many devices feature hardware optimized for data types other than 32-bit floating-point numbers, the standard representation defined by MLPerf. Edge-computing devices often feature app-specific hardware to offload common operations found in ML apps from the constrained CPU. This research analyzes multiple low-power compute architectures that feature ML-specific hardware on a case study of handwritten Chinese character recognition. Specifically, AlexNet and a custom version of GoogLeNet are benchmarked in terms of their streaming latency for optical character recognition. Considering these models are custom and not the most widely used, many architectures are not specifically optimized for them. The performance of these models can stress devices in different, yet insightful, ways that generalizations of the performance of other models can be drawn from. The NVIDIA Jetson AGX Xavier (AGX), Intel Neural Compute Stick 2 (NCS2), and Google Edge TPU architectures are analyzed with respect to their performance. The design of the AGX and TPU devices showcased the lowest streaming latency for AlexNet and GoogLeNet, respectively. Additionally, the tightly-integrated NCS2 design showed the best generalizability in performance and efficiency across neural networks.**

***Index Terms*—Machine learning, Inference, Benchmarking, Accelerator architectures, Low-power**

I. Introduction

Convolutional neural networks (CNNs), along with other variations of neural networks, are increasingly popular algorithms for machine learning (ML). More specifically, deep neural networks (DNNs), or neural networks with more than one layer, are employed to realize low-latency object classification due to their limited preprocessing requirements compared to other similar algorithms [1]. This characteristic makes DNNs attractive for real-time object classification on video streams where the typical image capture rate of most devices is 60 Hertz, or about 16 milliseconds per frame. To keep up with these real-time requirements, hardware acceleration is often necessary due to the large compute complexity of DNNs, typically on the order of giga operations (GOPs) per inference.

Many emerging devices include hardware for accelerating ML and artificial intelligence (AI) apps to meet the demands of this rapidly growing domain. Neural networks fundamentally consist of many vector and matrix operations, so devices are designed to compute these operations as fast as other instructions. NVIDIA's Volta, Turing, and Ampere architectures all feature Tensor cores, which are dedicated hardware to accelerate these operations, with each generation supporting more data types and parallelism than the previous [2]. Google's TPU architectures are complete accelerator devices designed specifically for deep-learning (DL) apps [3], which employ DNNs. These architectures, along with many other accelerator devices, aid the CPU in offloading the extensive computational requirements associated with DNNs.

Embedded CPUs are even more constrained than server- or desktop-class CPUs and therefore could benefit even more from these ML-oriented accelerators. Many system-on-chip (SoC) designs integrate ML-specific hardware or feature it as a coprocessor in the same package to reduce the strain on the CPU. Embedded CPUs often do not have the compute or memory resources to reach real-time performance alone, so these accelerators enable low-power platforms to achieve real-time performance. Edge-based ML apps' goals are often computer vision related, requiring video stream processing. Many of these apps, like autonomous navigation, are critical by nature, which makes real-time performance a hard requirement, and necessitates hardware acceleration. Therefore, it is crucial to understand the relative performance of these unique platforms on DL apps as they emerge.

MLPerf has been a growing industry standard for benchmarking devices using DNNs [4]. However, not all apps require the use of the same complex networks and datasets as defined by MLPerf. Additionally, many devices support optimizations for data types other than the MLPerf standard of 32-bit floating-point numbers. Optical character recognition, applied to Chinese characters, is a task that requires DNNs to achieve a high accuracy. However, this task does not require the deeper, more complex networks and larger precision used for the ImageNet recognition task defined by MLPerf. This research does not aim to set a standard for benchmarking like

This research was supported by SHREC industry and agency members and by the IUCRC Program of the National Science Foundation under Grant No. CNS-1738783.

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

MLPerf but does analyze the streaming latency of multiple embedded architectures on the task of handwritten Chinese character recognition (HCCR) to build on previous work defined in [8].

II. Background

The architectures, platforms, and frameworks used in this research vary widely, but serve to illustrate the performance that each type of device can achieve and build a comparison between them. This section outlines how these critical pieces are related and how they fit into the overall scope of this research.

A. Embedded GPU Accelerators

GPU architectures are widely used in embedded platforms to offload critical, data-parallel computation, such as graphics rendering, that would otherwise significantly strain the limited CPU resources. These GPU architectures are often embedded into the same chip as CPUs for a more efficient SoC configuration. One of NVIDIA's latest embedded platforms is the Jetson AGX Xavier (AGX) featuring the Xavier SoC. The Xavier SoC features a custom octa-core NVIDIA Carmel ARM-based CPU that is configured as four dual-core, heterogeneous CPU clusters. Additionally, a GPU based on NVIDIA's Volta architecture, which contains Tensor Cores for accelerating matrix-multiply-and-accumulate (MMAC) operations, is featured on the Xavier SoC. Many apps, such as those consisting of DNNs, significantly rely on matrix operations. Therefore, these Tensor Cores are designed to accelerate the common case (matrix operations). To further improve on DL performance, the Xavier SoC also features a Deep-Learning Accelerator (DLA) that is meant to accelerate convolutional layers found in DNNs [5]. The DLA features two pipelines for additional throughput of the convolutional operations, which can often be performed in parallel. The Xavier SoC also features other accelerators for vision and video-processing apps; however, they are not critical to the performance of this research.

B. Vision Processing Unit (VPU)

VPUs are a class of processors designed to accelerate computer-vision apps, which are apps that require a computer to recognize its environment. Intel's Neural Compute Stick 2 (NCS2) platform featuring the Movidius Myriad X VPU is a USB accelerator for low-power, high-performance DL. The Myriad X SoC combines, on the same chip, SPARC CPU cores and a variety of accelerators for power-efficient, low-latency computer-vision apps. The Neural Compute Engine (NCE) is one such accelerator that is designed with an array of multiply-accumulate (MAC) blocks to accelerate fundamental neural-network computations (common case). In addition to the NCE, 16 Streaming Hybrid Architecture Vector Engine (SHAVE) cores, which are lightweight vector processors, can be used to accelerate custom neural-network layers. The NCE, SHAVE cores, and other vision accelerators are connected to the same intelligent memory architecture. This memory subsystem is capable of supplying necessary data to processors fast enough to avoid memory bottlenecks often seen in other architectures when performing neural-network computations [6]. Similar to the Xavier SoC, the Myriad X SoC also features other accelerators for vision and video-processing apps.

C. Tensor Processing Unit (TPU)

To expand the idea of accelerating the common case for neural-network computations, Google designed its cloud TPU with a large systolic array (128×128) of MAC units in each of its scalar, vector, and matrix units (MXUs). The details of Google's Edge TPU are not specified; however, it is assumed the edge version resembles the cloud versions on a significantly smaller scale. Depending on the cloud TPU version, a TPU core can contain one (v2) or two (v3) MXUs. Both TPU versions v2 and v3 have 2 cores per chip [3]. Coral, a company specializing in AI products, has developed solutions with Google that feature the Edge TPU processor. Coral's Dev Board (TPU-DEV) features the Edge TPU as a coprocessor together with the NXP i.MX 8M SoC on their system-on-module (SoM) design. Additionally, Coral offers a USB accelerator (TPU-USB) that features the same Edge TPU to serve as a coprocessor for any system [7].

D. Caffe

Caffe is an open-source framework for creating apps using neural networks developed by Berkeley AI Research. Caffe is used specifically in this research as an extension to the work presented in [8] to compare with other accelerators. NVCaffe is a fork of Caffe that is developed by NVIDIA to efficiently leverage NVIDIA hardware such as GPUs and Tensor Cores found in the AGX platform. For more information, the reader is referred to [9] and [10].

E. TensorFlow

Similar to Caffe, TensorFlow (TF) is another open-source platform for ML apps developed by Google Brain. TensorFlow Lite (TFLite) is a lightweight version of TF designed for inferencing on embedded platforms. For more information, the reader is referred to [11].

F. OpenVINO

OpenVINO is a toolkit provided by Intel to accelerate apps using neural networks efficiently on Intel-specific hardware. OpenVINO takes pretrained models from Caffe, TF, or other ML frameworks and creates configuration files to run the given models on the target Intel device. For more information, the reader is referred to [12].

III. Related Research

This research is an extension of the work presented in [8]. While the previous work looks at the maximum throughput of high-performance devices, this research studies the same app with a focus on single-image latency. Embedded devices are not well suited for maximizing throughput due to their restricted resources; therefore, single-image latency is more interesting to examine. Additionally, many ML-based apps that use embedded platforms focus on video-streaming, which

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

involves processing frames as they become available one after the other as opposed to multi-image classification.

The models used in this research were originally presented in [13]. This previous work showed that AlexNet and a custom version of GoogLeNet could classify handwritten Chinese characters with high accuracies. The AlexNet model is a neural network with common layers that are featured in most other neural-network models and can be seen in Fig. 1 [14]. The GoogLeNet model is a custom variant of Inception v2 that uses the first 14 layers, as opposed to 22 layers in the original design. The previous work showed the additional layers were not necessary as the overall accuracy stops improving for a significant cost of additional computation. The Inception modules featured in the GoogLeNet architecture result in wider layers, which recruit more available parallel hardware [15]. The Inception modules featured in the GoogLeNet architecture perform multiple convolutions of varying kernel size at the same layer, as seen in Fig. 2.

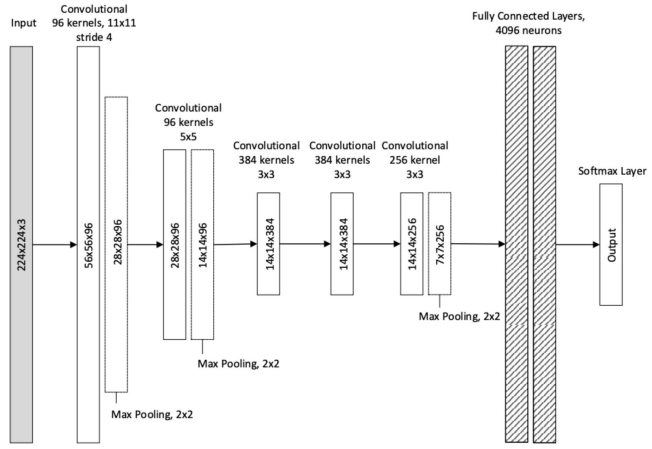

Fig. 1. AlexNet Architecture [16]

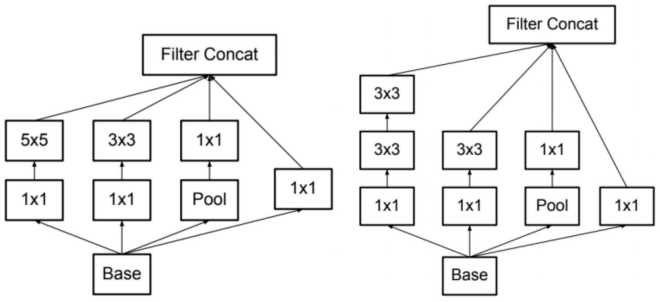

Fig. 2. Naive (Left) Inception v2 (Right) Inception Module Structure [16]

The MLPerf Inference Benchmark is an increasingly-adopted standard for benchmarking neural networks and hardware platforms [4]. While this research does not use the specific models, datasets, and data types that are outlined in the MLPerf guidelines, this research adheres to the core philosophy of the MLPerf benchmark by presenting a fair comparison across models and devices. This research cannot be compared directly with MLPerf performance results.

Other research has been presented that outlines the current state-of-the-art for resource-limited ML and general best practices when creating apps [17] [18] [19]. A similar benchmarking study to this research, albeit on spike-based computing as opposed to CNN inference, showed that NCS2 performed significantly worse in terms of latency and efficiency than the Edge TPU device [20]. Additionally, other research showcases the performance of the Edge TPU alongside a mobile CPU and the NVIDIA Jetson Nano [21]. This research showed that the Maxwell-based GPU, which does not feature ML-specific hardware, performs significantly worse than the other devices in the study on Inception v2 using TFLite. Significant research also has been presented that outlines architectures to optimize deep-learning performance [22], which supplements the architectural studies presented in our research.

IV. Methodology

The goal of this research is to investigate streaming latency and efficiency of high-performance, low-power accelerators. The HCCR app is developed using ML-frameworks such as Caffe, TFLite, and OpenVINO for productivity and optimized accelerator performance. A C++ version of the app is used for Caffe and OpenVINO, while a Python version is used for TFLite. The app uses 252,545 images from the CASIA database for classification considered in [8] and [13]. The batch size is limited to one image because streaming apps process a single frame at a time from a video stream, assuming required real-time performance. The average inference performance is taken after classifying the entire image set for 50 iterations. All operations are performed using 16-bit floating-point precision except in the case of the TPU devices, which only support 8-bit integer precision. Although the data types are not consistent across all devices, the devices contain hardware for accelerating operations at the precision used. Converting all models to 8-bit integer precision may show performance improvements, but it is not necessarily guaranteed. Executing all models at 32-bit floating-point precision would show a significant performance impact because device-specific hardware could no longer be used efficiently.

A. NVIDIA AGX

The original version of the app described in [8] focuses on the use of Caffe on many different accelerators. Similarly, NVCaffe is used on the AGX platform to benchmark the Xavier SoC. The Carmel CPU featured on the Xavier SoC is benchmarked using Caffe as a baseline, non-accelerator performance metric.

B. Coral Google Edge TPU

The Edge TPUs support only TFLite models. The Caffe models are manually converted to TFLite to benchmark both TPU platforms. In the event that a layer cannot be executed on the TPU, the app falls back to the CPU for computation before restoring execution on the TPU. The performance of

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

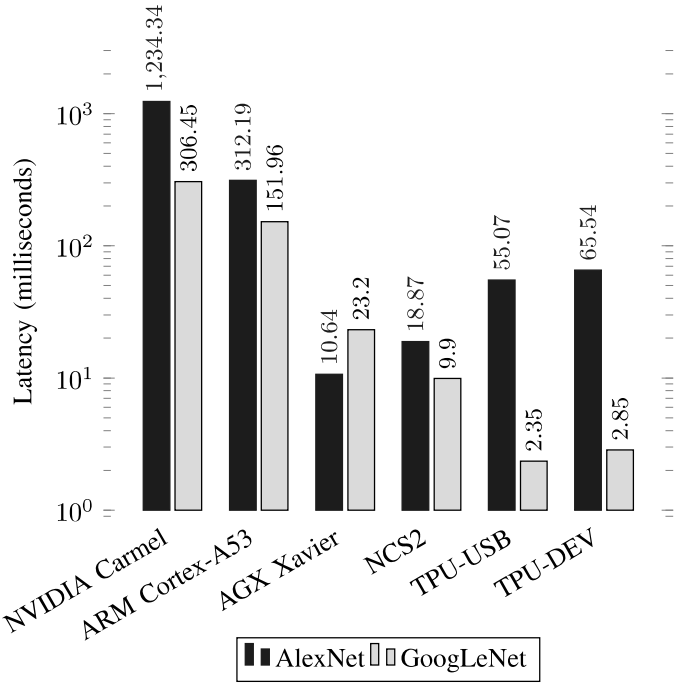

Fig. 3. HCCR Streaming Latency Across Devices and Models

the Arm Cortex-A53 CPU featured on the Coral Dev Board is benchmarked using the same TFLite version of the app for a baseline performance on another low-power CPU. The Caffe models are converted to the TFLite format and quantized to 8-bit integer types. The USB accelerator connects to a system featuring an Intel Core i7-7700 running at 3.6 GHz with 16GB of 2400 MHz DDR4 main memory.

C. Intel NCS2

Similar to the TPU, The Intel NCS2's programmabilty is limited to the use of Intel's OpenVINO toolkit. The original Caffe models are converted using the OpenVINO model optimizer, which both converts the models and creates configuration files for proper mapping on the target device. The NCS2 is connected to the same system as the TPU USB accelerator.

V. Results

The main performance metric is the latency of single-image classification. Fig. 3 displays the streaming latency, or the latency of the neural networks with a batch size of one image, across all devices. Smaller latency values are preferred.

A. Design Power

Table I lists the expected power consumption of each device as thermal design power (TDP). The dynamic power consumption for the app on each accelerator is not easily measurable, especially on SoC platforms. We assume the peak power potential in most cases. For the NCS2 and TPU, the TDP is explicitly given. For the AGX platform's 30W power mode, each core of the Carmel processor uses up to a maximum 1.5W of power. Assuming this is the case, the the GPU and DLA on the SoC can be roughly estimated to be using 18W, which is a better estimate than the full 30W. For the NXP i.MX 8M platform featuring the ARM Cortex-A53, we used a power benchmark of relatively similar floating-point computations.

TABLE I
TDP of Devices

Device	TDP (W)
NVIDIA Carmel CPU	12.00 [5]
ARM Cortex-A53 CPU	1.25 [23]
NVIDIA AGX Xavier GPU + DLA	18.00 [5]
Intel NCS2 (Myriad X)	1.50 [24]
Google Edge TPU	2.00 [7]

B. Efficiency

Efficiency is described in terms of throughput-per-power, as opposed to latency-per-power. Therefore, we use the reciprocal of latency as the throughput in terms of images-per-second. This throughput metric is still limited to one image per batch to calculate the streaming efficiency in terms of images-per-second-per-Watt. With our power assumptions, efficiency would increase as the batch sizes are increased. The batch sizes were scaled up until the devices exhausted main memory to get a sense of how much additional parallel hardware is available. The TPU devices are limited to one image per batch. All other devices were limited ($< 1.5\times$) in relative improved performance at larger batch sizes besides the AGX platform, which saw a throughput increase of $16.3\times$ and $18.6\times$ for AlexNet and GoogLeNet, respectively. Efficiency metrics for each device can be seen in Fig.4. Larger efficiency values are preferred.

VI. Discussion

The performance of the devices on various models depends on the devices' architectural design. This sections analyzes why the performance of the models are impacted by different architectures. Investigation into the efficiency of devices related to architecture is also discussed.

A. Accuracy

The accuracies for each model over each iteration were recorded to verify the models' performance. These accuracies are not critical to the nature of this research. However, the average Top-1 accuracies for AlexNet and GoogLeNet across devices that use 16-bit floating-point precision were 94.9% and 96.6%, respectively. The average Top-1 accuracies for AlexNet and GoogLeNet across TPU devices were 90.3% and 93.8%, respectively. This reduction in accuracy is expected and consistent with [25] and [26] for quantization and reduced precision of DNNs. The size of parameters needed for the AlexNet model is $5.8\times$ and $4.6\times$ larger than the GoogLeNet model for Caffe and TFLite, respectively.

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

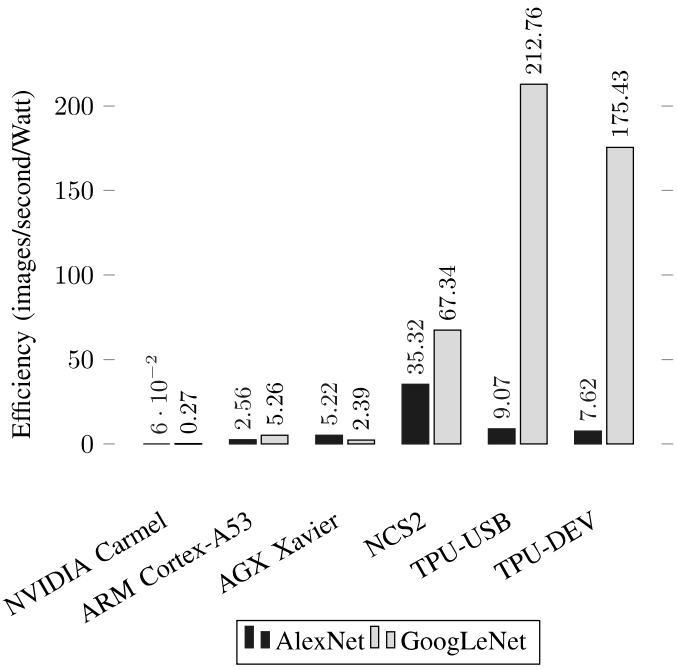

Fig. 4. HCCR Streaming Efficiency Across Devices and Models

B. Performance

Comparing the performance of the CPU baselines, we observe that the Carmel's performance is significantly lower than the A53's performance. The A53 processors are designed for maximizing efficiency, so the Carmel processor should display better performance given the same app [27]. The reason for the performance disparity is because of the use of Caffe on the Carmel CPU versus TFLite on the A53 CPU. TFLite is specifically designed for devices with limited compute, memory, and power resources and is therefore more optimized for embedded platforms than Caffe. AlexNet's performance on both platforms is significantly worse than GoogLeNet's due to its larger model size, resulting in a memory-bottleneck.

In the case of GoogLeNet, all of the accelerator architectures perform within the same order of magnitude, besides the AGX. Conversely, the AGX shows the best performance out of all devices in the case of AlexNet. Based on both of these results, we can infer that the top AlexNet performance is most likely due to the dual-DLA pipeline. If the DLA pipeline was utilized in both models, the GoogLeNet model's performance would suffer due to the parallel nature of the Inception modules requiring more than two pipelines for convolution acceleration, which is what we observe. The NCS2 does not have a specific convolutional accelerator similar to the DLA. However, the tightly-coupled NCE, SHAVE cores, and memory fabric can achieve similar functionality at a slightly degraded performance. The TPU accelerator performs significantly worse than the other accelerators on AlexNet. Since the TPU is a coprocessor and not embedded into the SoC similar to the other devices, memory accesses incur much larger penalties without the main memory residing on-chip. The AlexNet case makes this design characteristic apparent because of its significantly larger model size, requiring the accelerator to continuously wait for off-chip data.

The best performance observed out of all devices comes from the TPU accelerator on the GoogLeNet model. More specifically, the USB accelerator variant performs better than the Dev Board variant. This is mostly likely due to the larger system memory and higher-performance CPU, although the communication latency to those systems is longer over USB. The details of the TPU architecture are not publicly available. However, the featured MXU would enable the TPU to process the GoogLeNet Inception modules in parallel, enabling much higher performance at each layer. Because of the reduced GoogLeNet model size, a layer could be computed without needing to go to main memory multiple times. Additionally, memory latencies could be hidden while computing a previous layer. The TPU, most likely, needs to fetch multiple data partitions for a single layer of the AlexNet model, which results in the observed performance. An important caveat with this result is that inference was performed at a reduced precision with a quantized model, significantly reducing the complexity of the app. The TPU devices fail on this task at 16-bit floating-point precision because it does not have hardware to support that precision.

While the NCS2 does not perform the best in either case, it does perform the best in the average case across models meaning its performance generalizes well to the contrasting models architectures. The NCS2's design is specific to neural-network performance without the massive parallelism most likely found in the TPU. However, its memory, processor, and accelerator subsystems are significantly more integrated, similar to the AGX. This specific architecture enables the NCS2 to perform well on both models, not optimizing for one over the other. No device achieves the real-time performance required for both networks; however, the NCS2 is the only device that achieves real-time performance in the average case showcasing good performance on both networks instead of just one. The NCS2 only achieves real-time performance on GoogLeNet, while only slightly missing the performance requirement for AlexNet.

C. Efficiency

The efficiency metrics are difficult to discuss without exact numbers to support the performance achieved. However, in low-power apps, it is critical to discuss the efficiency of the devices studied. The Carmel CPU suffers in efficiency due to it being the worst performing device. In contrast, the A53 CPU's low-power characteristics make it competitive with the AGX GPU and DLA due to the AGX platform having the highest power consumption characteristics. As mentioned before, the AGX platform's throughput scales significantly higher than the rest of the devices when increasing the number of images per batch, using more parallel hardware. Therefore, the actual power consumption for the streaming latency would be significantly lower than what is noted for the AGX GPU.

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

However, this scaled throughput does not apply in the case of video-streaming apps.

The NCS2 and TPU accelerators both feature similar low-power profiles, with the NCS2 being slightly lower power. The TPUs both showcase the best GoogLeNet efficiency, with the USB Accelerator being more efficient, due to the significantly lower latency than the NCS2. However, the NCS2 displays the best AlexNet efficiency due to its competitive, low-latency performance at a lower power than both the TPU and AGX. The TPU accelerators prove to be the best in the average case, although with a much higher standard deviation than any of the other platforms. The significance of this large standard deviation is that the TPU has poor generalization characteristics. The TPU performs well on models with a small number of parameters but struggles significantly on models with a larger number of parameters. This consideration is in contrast to the high generalizability of the Intel NCS2 that shows consistent efficiency across both model types.

VII. Conclusions

While this research cannot be compared to a standard benchmark like MLPerf, which restricts the data, models, and data types used and is not representative of all ML apps, many insights can be drawn from this case study. TFLite displays significantly better performance than Caffe for memory-, compute-, and power-constrained platforms like ARM CPUs. There was no single accelerator that performed the best in every case or achieved real-time performance for both neural networks. The AGX's massively parallel architecture enabled it to perform the best on the larger-memory AlexNet model. The TPU architecture, optimized for neural-network computations, enabled it to perform the best on the smaller-memory GoogLeNet model, avoiding costly memory-access penalties. Additionally, the NCS2 displayed the best average performance across both models because of its tightly-coupled, ML-focused architecture, showcasing its strong generalization characteristics. While the efficiency characteristics of the devices cannot be precisely evaluated, the TPU devices proved to be the most efficient on the GoogLeNet model and in the average case despite having the largest standard deviation. As many different architectures, especially those designed for ML and AI, keep evolving, it is crucial to understand their relative performance to leverage optimal hardware for critical apps.

VIII. Future Work

One future direction for this research would be to standardize all of the frameworks and data types used. Caffe is used predominantly as an extension of previous work, so moving all devices over to 8-bit integer precision, TFLite models would standardize this evaluation. All software methods used in this research were optimized for their respective hardware platforms, besides the case of the CPUs. Additionally, more models, such as MobileNet and ResNet, can be included for a closer representation of the MLPerf suite. Many other devices and platforms will be studied as they become available.

References

[1] M. Egmont-Petersen, D. de Ridder, H. Handels, "Image processing with neural networks - a review," Pattern Recognition, vol. 35, no. 10, pp. 2279-2301, 2002.

[2] NVIDIA, "NVIDIA A100 Tensor Core GPU Architecture," June 2020. [Whitepaper] Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf [Accessed June 2020]

[3] Google LLC, "System Architecture," June 2020. [Datasheet] Available: https://cloud.google.com/tpu/docs/system-architecture [Accessed June 2020]

[4] P. Mattson, H. Tang, Et. Al, "MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance," 2020 IEEE Micro, vol. 40, no. 2, pp. 8-16, February 2020. DOI: 10.1109/MM.2020.2974843

[5] D. Franklin (NVIDIA), "Jetson AGX Xavier and the New Era of Autonomous Machines Webinar," June 2020. [Presentation] Available: https://info.nvidia.com/jetson-xavier-and-the-new-era-of-autonomous-machines-reg-page.html [Accessed June 2020]

[6] Intel, "Intel® Movidius™ Myriad™ X VPUs," June 2020. [Datasheet] Available: https://www.intel.com/content/www/us/en/artificial-intelligence/movidius-myriad-vpus.html [Accessed June 2020].

[7] Google LLC, "Coral Dev Board Datasheet Version 1.3," January 2020. [Datasheet] Available: https://coral.ai/static/files/Coral-Dev-Board-datasheet.pdf [Accessed June 2020].

[8] L. Kljucaric, A. George, "Deep-Learning Inferencing with High-Performance Hardware Accelerators," 2019 IEEE High Performance Extreme Computing Conference. September 2019. DOI: 10.1109/HPEC.2019.8916463

[9] Y. Jia, E. Shelhamer, Et. Al, "Caffe: Convolutional Architecture for Fast Feature Embedding," in 22nd ACM international conference on Multimedia MM, 2014.

[10] NVIDIA, "NVCaffe," January 2020. [Technical Manual] Available: https://docs.nvidia.com/deeplearning/frameworks/caffe-user-guide/index.html [Accessed June 2020].

[11] M. Abadi, A. Agarwal, Et. Al, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Software] Available: https://www.tensorflow.org/ [Accessed June 2020].

[12] Intel. "OpenVINO Toolkit," June 2020. [Technical Manual] Available: https://docs.openvinotoolkit.org/ [Accessed June 2020].

[13] Z. Zhong, L. Jin, Z. Xie, "High Performance Offline Handwritten Chinese Character Recognition Using GoogLeNet and Directional Feature Maps," 13th International Conference on Document Analysis and Recognition. 2015.

[14] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Neural Information Processing Systems, vol. 25, no. 2, 2012. DOI: 10.5555/2999134.2999257

[15] C. Szegedy, W. Liu, Et. Al, "Going Deeper With Convolutions," in IEEE Computer Vision and Pattern Recognition (CVPR), 2015. DOI: 10.1109/CVPR.2015.7298594

[16] L. Tsochatzidis, L. Costaridou, and I. Pratikakis, "Deep Learning for Breast Cancer Diagnosis from Mammograms—A Comparative Study," in The Journal of Imaging, vol. 5, no. 37, 2019. DOI: 10.3390/jimaging5030037

[17] A. Sobecki, J. Szymański, Et. Al, "Deep Learning in the Fog." in International Journal of Distributed Sensor Networks, August 2019. DOI:10.1177/1550147719867072.

[18] S. Pouyanfar, S. Sadiq, Et. Al, "A Survey on Deep Learning: Algorithms, Techniques, and Applications," in ACM Computing Surveys, Article 92, January 2019. DOI:https://doi-org.pitt.idm.oclc.org/10.1145/3234150

[19] N. D. Lane, S. Bhattacharya, Et. Al, "An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices," in Proceedings of the 2015 International Workshop on Internet of Things towards Applications (IoT-App '15). Association for Computing Machinery, pp. 7–12. DOI: https://doi-org.pitt.idm.oclc.org/10.1145/2820975.2820980

[20] J. Sengupta, R. Kubendran, Et. Al, "High-Speed, Real-Time, Spike-Based Object Tracking and Path Prediction on Google Edge TPU," in 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Genova, Italy, 2020, pp. 134-135, doi: 10.1109/AICAS48895.2020.9073867

[21] Z. Hu, A. B. Tarakji, Et. Al, "DeepHome: Distributed Inference with Heterogeneous Devices in the Edge," in The 3rd International Workshop on Deep Learning for Mobile Systems and Applications

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

(EMDL '19). Association for Computing Machinery, pp. 13–18. DOI:https://doi.org/10.1145/3325413.3329787

[22] N. D. Lane, S. Bhattacharya, Et. Al, "Squeezing Deep Learning into Mobile and Embedded Devices," in IEEE Pervasive Computing, vol. 16, no. 3, pp. 82-88, 2017, doi: 10.1109/MPRV.2017.2940968

[23] NXP Semiconductors, "i.MX 8M Quad Power Consumption Measurement Rev. 2," August 2018. [Application Note] Available: https://www.nxp.com/docs/en/nxp/application-notes/AN12118.pdf [Accessed June 2020].

[24] M. Antonini, T. Huy Vu, Et. Al, "Resource Characterisation of Personal-Scale Sensing Models on Edge Accelerators," The First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things. Association for Computing Machinery, pp. 49–55. November 2019. DOI: https://doi.org/10.1145/3363347.3363363

[25] N. Wang, J. Cho, Et. Al, "Training Deep Neural Networks with 8-bit Floating Point Numbers," in The Conference on Neural Information Processing Systems (NIPS). 2018. Available: https://papers.nips.cc/paper/7994-training-deep-neural-networks-with-8-bit-floating-point-numbers.pdf [Accessed June 2020].

[26] P. Judd, J. Albericio, Et. Al, "Reduced-Precision Strategies for Bounded Memory in Deep Neural Nets," CoRR, vol. abs/1511.05236, 2015. [Online]. Available: http://arxiv.org/abs/1511.05236 [Accessed June 2020]

[27] ARM. "ARM Cortex-A53 MPCore Processor Technical Reference Manual," June 2020. [Technical Manual] Available: https://developer.arm.com/docs/ddi0500/g [Accessed June 2020].

978-1-7281-9219-2/20/$31.00 ©2020 IEEE