Deep-Learning Inferencing with High-Performance Hardware Accelerators

Luke Kljucaric, 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, alan.george}@nsf-shrec.org

***Abstract*— In order to improve their performance-per-watt capabilities over general-purpose architectures, FPGAs are commonly employed to accelerate applications. With the exponential growth of available data, machine-learning apps have generated greater interest in order to more comprehensively understand that data and increase autonomous processing. As FPGAs become more readily available on cloud services like Amazon Web Services F1 platform, it is worth studying the performance of accelerating machine-learning apps on FPGAs over traditional fixed-logic devices, like CPUs and GPUs. FPGA frameworks for accelerating convolutional neural networks (CNN), which are used in many machine-learning apps, have begun to emerge for accelerated-application development. This research aims to compare the performance of these forthcoming frameworks on two commonly used CNNs, GoogLeNet and AlexNet. Specifically, handwritten Chinese character recognition is benchmarked across multiple FPGA frameworks on Xilinx and Intel FPGAs and compared against multiple CPU and GPU architectures featured on AWS, Google's Cloud platform, the University of Pittsburgh's Center for Research Computing (CRC), and Intel's vLab Academic Cluster. All NVIDIA GPUs have proven to have the best performance over every other device in this study. The Zebra framework available for Xilinx FPGAs showed to have an average 8.3 times and 9.3 times performance and efficiency improvement, respectively, over the OpenVINO framework available for Intel FPGAs. Although the Zebra framework on the Xilinx VU9P showed greater efficiency than the Pascal-based GPUs, the NVIDIA Tesla V100 proved to be the most efficient device at 125.9 and 47.2 images-per-second-per-Watt for AlexNet and GoogLeNet, respectively. Although currently lacking, FPGA frameworks and devices have the potential to compete with GPUs in terms of performance and efficiency.**

Keywords*— *machine learning, FPGA, inference, xfDNN, Volta, AWS, F1, PAC, OpenVINO,

I. Introduction

The explosive growth of available data for training machine-learning models has driven a heavier focus on the development of artificial-intelligence apps. This growth in data requires faster, more efficient, and more intelligent processing. Machine-learning apps for image processing often use convolutional neural networks (CNNs) in their models for processing new, unclassified data autonomously. CNNs are attractive for these types of applications because they require minimal preprocessing in comparison to other methods in order to extract image features [1]. The goal of CNNs is to extract features from the input images, which is necessary in order to have a common representation of images associated with a class. An image feature is a measurable property of an image, such as outlines of shapes and patterns among sets of images. The CNN is trained to recognize these features and associate the same patterns to similar classes of images. The features should be unique between classes and common within a class, so the CNN can make clear inferences. The parallel nature of CNNs, consisting of convolutions and matrix multiplications, make them highly amenable for GPU and FPGA acceleration.

Traditional acceleration of CNNs on FPGAs has been performed by implementing a specific neural-network processor in hardware [2] [3] [4] [5]. This process can lead to lengthy design times and limited flexibility when the model or application domain is changed. Frameworks for accelerating CNNs on FPGAs for use with machine-learning frameworks, like Caffe and MxNet, are being developed to address these issues. The same apps that use GPUs for acceleration can leverage these frameworks to use FPGAs instead, with limited configuration of the FPGA required.

Limited research has been presented on studying these FPGA frameworks for accelerating CNNs. It is important to understand the performance of these emerging frameworks to optimally use FPGAs in machine-learning app acceleration. While other architectures, like GPUs, are also popular for accelerating machine-learning apps, it is beneficial to compare the performance of these FPGA frameworks to GPU and fixed-logic devices, for example, to investigate reductions in energy consumption. Many different toolkits and frameworks exist to leverage Intel and Xilinx devices in different ways. It is challenging, yet important, to be able to compare different frameworks on different architectures.

In this research, we evaluate and compare current architectures and frameworks for CNN acceleration on various FPGAs, GPUs, and CPUs with a case study in Chinese character recognition. This evaluation will aid in the understanding of the relative performance in terms of throughput and efficiency of many different acceleration platforms. As focus begins to shift from machine-learning training to inferencing, it is important to understand the architectures and frameworks to best accelerate machine-learning inferencing apps and effectively design high-performance computing (HPC) systems oriented to machine learning.

II. Background

Many different concepts, tools, frameworks, and devices are used in this study to understand the current HPC machine-learning inferencing domain. This section aims to explain all

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.

the components necessary for the app acceleration on different architectures, frameworks, and platforms.

A. Machine-Learning Inference

With the trained network, inferencing can be performed on new data. For more information, the reader is referred to [6], [7], [8], [9], [10], [11], [12], [13].

B. Caffe Machine-Learning Framework

Caffe is an open-source framework for developing machine-learning apps. In this research, we use Caffe because the CNN acceleration frameworks for FPGAs currently only fully support Caffe models [14] [15] [16]. For more information, the reader is referred to [17].

C. Field-Programmable Gate Arrays (FPGAs)

Unlike fixed-logic devices such as CPUs and GPUs, FPGAs are reconfigurable-logic devices. FPGAs are capable of realizing dedicated data-paths that map to application functions, resulting in more efficient processing compared to fixed-logic devices. These custom data-paths often give FPGAs an advantage over fixed-logic devices in terms of performance-per-watt. Many different data-paths can be instantiated onto the FPGA in parallel, which makes these devices amenable for accelerating CNN-based apps. Although the data-paths on FPGAs are typically longer than fixed-logic devices, the energy-efficiency comes from the parallelism in the design [8].

D. Xilinx Framework for Deep Neural Networks (xfDNN)

The xfDNN v2 framework, also referred to as xDNN, aims to accelerate CNNs on Xilinx FPGAs. The framework has support for custom neural networks, which has allowed for more general usage of the framework [14]. Xilinx provides a compiler tool which maps layers of the CNN being used in an application to xfDNN for proper acceleration. This compilation is one of the few extra steps required for accelerating an application with xfDNN. The xfDNN framework has multiple configuration profiles. The two main profiles are the 4×28×32 and 2×56×32 configurations. The difference between these configurations is the number of processing elements (PE) being used. A processing element is the main computational unit of xfDNN. There are 4 and 2 processing elements in the 4×28×32 and 2×56×32 configurations, respectively. The labels 28×32 and 56×32 signify the DSP array configuration used for each PE. The differences between the two designs are that the 56×32-labeled core can process higher-resolution images at a lower latency, whereas the 28×32-labeled core is designed for maximum throughput [18]. Caffe is used on the CPU side of the application, which then makes reference to xfDNN for FPGA acceleration. There is no source provided for xfDNN, only a precompiled binary.

E. Mipsology Zebra

Zebra is a closed-source framework for Xilinx FPGAs which was developed by Mipsology. Mipsology claims that Zebra can take any existing Caffe application for CPUs and GPUs and execute it using the Zebra runtime on Xilinx FPGAs [15]. This portability is an attractive feature when trying to port existing applications to different device architectures quickly. Similar to xfDNN, Zebra can be configured with a different number of "cores." No documentation exists for the usage or details of the cores, but the default is set to six cores. Additionally, like xfDNN, the main application makes calls to the Caffe API which then references the Zebra framework.

F. Graphics Processing Units (GPUs)

GPUs have been widely used in machine-learning apps for their highly parallel nature. GPUs are typically comprised of thousands of lightweight cores which allow for acceleration of massively-parallel math operations, similar to those found in CNNs. The Volta architecture featured in the 12-nm Tesla V100 was designed with machine-learning apps in mind. The convolutional layers in CNNs are computed through matrix multiplication and accumulation operations. The Volta architecture on the V100 contains over 600 "Tensor cores" that each perform four-by-four, half-precision matrix multiplication and full-precision accumulation in a single clock cycle. These Tensor cores give the Volta architecture a significant advantage in machine-learning apps versus previous GPU architectures [19]. To leverage GPU-specific hardware, like Volta Tensor cores, a fork of Caffe has been developed by NVIDIA known as NVCaffe. For more information, the reader is referred to [20].

G. Many-Core CPUs

With the intention of being clusters-on-chip, it is important to include many-core CPUs in this comparison to understand their increasingly parallel performance in the machine-learning domain against other HPC devices. For more information on the devices in this study, the reader is referred to [21], [22], [23], [24], [25].

H. Intel Machine-Learning Software

Intel also develops different tools to optimally leverage their different devices and architectures for machine-learning apps. The first is a fork of Caffe known as Intel-Optimized Caffe or Caffe*. For more information, the reader is referred to [26].

The OpenVINO toolkit from Intel provides another method for accelerating machine-learning apps on Intel CPUs, GPUs, FPGAs, and other accelerators. For more information, the reader is referred to [27].

I. Handwritten Chinese Character Recognition (HCCR)

In order to test the FPGA frameworks fully, the machine-learning app must be challenging enough to require a deep network with many layers. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a competition in which neural networks classify images from typically 200 categories. This competition poses as a standard benchmark for stressing neural-network performance. There is a demand to understand the performance of optical character recognition on FPGAs, so classifying Chinese characters from thousands of different classes could potentially be as challenging as ILSVRC. Specifically, the Institute of Automation of Chinese Academy of Sciences has a handwritten database of 7653 different Chinese characters [28]. Classifying images from 7653 distinct classes is a difficult task that requires large neural networks in order to accurately extract distinct, relevant features from the handwritten images.

J. Caffe-Accelerator Model

Many of the frameworks and devices serve similar roles in the acceleration of the HCCR app. Figure 1 visualizes the acceleration model between Caffe, accelerators, frameworks, CNN models, and data. The Caffe forks, such as NVCaffe and Intel Caffe*, can be substituted for Caffe on the CPU in this

model. OpenVINO is the only instance where the framework is the same for both CPU and accelerator combination. In the case of Intel Caffe*, the Intel MKL is used for accelerating the application on the CPU itself, so no offloading occurs. In Figure 1, the underlined values are used in this research and nonunderlined values show other common examples.

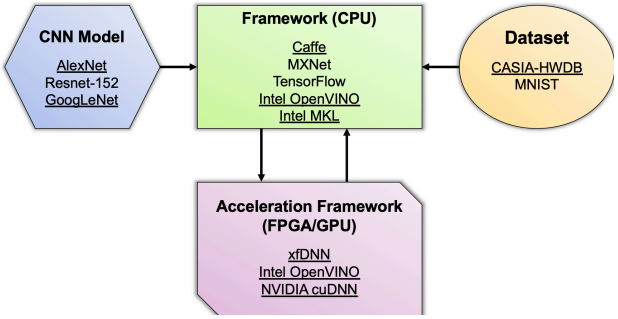

Figure 1 Caffe Accelerator Model.

III. Related Work

Previous work has informed this research by exploring complex classification problems such as HCCR. In order to fully understand the capabilities of the frameworks in this study, it is important to stress them in many ways. While the developers of the acceleration frameworks claim strong performance on well-known CNN models like AlexNet and GoogLeNet, it is important to understand how these frameworks perform with custom CNNs. Using custom CNNs will remove optimizations the frameworks have for specific CNNs. Previous work has shown that AlexNet and a variant of GoogLeNet can be used to perform high-performance, online, handwritten Chinese character recognition at high accuracies [29] [6]. These DCNN models demand high-performance from the frameworks which stress their capabilities. The custom variant of GoogLeNet was developed because the full depth of GoogLeNet is not required for the HCCR application [6]. This variant uses 14 layers as opposed to the 22 layers in the standard version.

For more information on previous work accelerating CNNs on FPGAs, the reader is referred to [2], [3], [4], [5], [30].

IV. Methodology

The main focus of this research is to benchmark existing frameworks for accelerating CNNs on FPGAs, GPUs, and CPUs for a performance comparison of the architectures on machine-learning apps. Two different versions of the same HCCR app are used. The first is a C++ app that loads bitmap image files from a directory and uses the Caffe API to classify the images in batched mode, evaluating a specific number of images at each execution step. The second C++ app is similar to the first, except that it uses the OpenVINO API in order to classify the batched images. In each case, the app is executed using batch sizes of 1, 2, 4, 8, 16, 32, 64, 96, 128, 256, 512, 1024, and 2048 where applicable. For the OpenVINO results, batch sizes are limited to 8 for the general variant and 96 for the optimized variants. The app classifies 252,545 images which are a subset of the CASIA database. The app runs 50 iterations of classifications on the entire dataset before averaging the resulting performance. Previous research has been done on this application by [6] and [29]. They showed that variants of GoogLeNet and AlexNet can be used to accurately classify handwritten Chinese characters. In order to fairly evaluate the frameworks and platforms, 16-bit operations were used for inferencing using the same handwritten database, CNNs, and pretrained models as the previous work. The OpenVINO toolkit does not support 16-bit operations with the CPU plugin, resulting in the OpenVINO Xeon CPU operations being 32-bit. Additionally, the GTX 1080 Ti upgrades FP16 operation to FP32, so the results for this device also use 32-bit, floating-point precision [31].

A. Xilinx FPGA Acceleration

The Xilinx FPGA studied is the Xilinx Virtex UltraScale+ (XCVU9P). Two frameworks, xfDNN and Zebra, will be evaluated on this device using the Caffe-based app on AWS. The specific xfDNN version is using the 4×28 PEs, and Zebra is configured using six soft cores. The Zebra configuration was left unchanged as recommended by the documentation. First, the xfDNN compiler is run using both AlexNet and GoogLeNet, which creates the resulting JSON files for proper network-specific acceleration on the xfDNN platform. Next, the xfDNN quantizer is run to create additional JSON files that specify scaling factors for the layers within each corresponding CNN to calculate the network using 16-bit operations. The xfDNN binary is loaded onto the Xilinx FPGA on AWS. Next, the Caffe-based app then loads the xfDNN library with the proper compiler and quantizer JSON files to accelerate inferencing on the xfDNN platform. When running the same app using the Zebra framework, no additional compiler or quantizer is required to generate additional files. Similar to xfDNN, the Zebra binary is loaded onto the Xilinx FPGA on AWS. Additionally, like xfDNN, the Caffe-based app loads the Zebra library and accelerates inferencing on the Zebra platform.

B. Intel FPGA Acceleration

The Intel FPGA studied resides on the Programmable Acceleration Card (PAC), which features an Arria 10 GX (10AX115N). The OpenVINO toolkit is used on the Intel FPGA on the Intel vLab cluster because Intel-based Caffe support does not exist for Intel FPGAs. In order to run the OpenVINO-based app, the CNNs are given to the OpenVINO model-optimizer application to create corresponding XML files for proper acceleration on the target device. For the Intel PAC, 16-bit operation model-optimizer files are created. In order to run the OpenVINO-based app on the Intel PAC, the 16-bit generic or network-optimized version of the OpenVINO binary is loaded onto the PAC on vLab. Finally, the OpenVINO-based app is run using the network-specific model-optimizer XML files in heterogenous mode, accelerating the app on the Intel PAC.

C. Intel CPU Acceleration

In order to compare the OpenVINO and Caffe results from different architectures, the Caffe-based app will also be executed on the Xeon CPU (SKL 8180) in addition to the OpenVINO-based app. From this comparison, we will be able to compare how the Caffe and OpenVINO frameworks perform on the same architecture and application to infer how the performance of the other architectures compare. For the Xeon CPU running the OpenVINO-based app, the

OpenVINO toolkit does not support 16-bit operations on the CPUs resulting in 32-bit operations. To run the OpenVINO-based app on the Xeon CPU, no additional steps are required such as loading additional binaries, so the app is run in CPU mode with the network-specific model-optimizer XML files. When running the Caffe-based app, the Xeon CPU specifically uses the Intel Optimized Caffe* and the Intel MKL. The rest of the CPUs in this case study, KNL (7250) and KNM (7295) provided by vLab, will only run the Caffe-based app using the Intel Optimized Caffe* and the Intel MKL.

D. NVIDIA GPU Accleration

The GPUs used in this study, as mentioned previously, are the NVIDIA Tesla P100 provided by Google Cloud, NVIDIA Tesla V100 provided by AWS, and the GTX 1080 Ti provided by CRC. All of these devices will run the Caffe-based app using NVCaffe and cuDNN. No additional steps are required when using the GPU platforms such as the xfDNN compiler and quantizer or the OpenVINO model-optimizer.

V. Results

The main metric of the study is performance in terms of images-per-second. Accuracy is not focused on specifically in this study because the performance of the neural networks should be similar no matter the network input. That said, this research did observe the Top-1 accuracies for AlexNet and GoogLeNet to vary between 94-96% and 96-97%, respectively, across the devices studied.

Table 1 shows the breakdown of xfDNN, NVIDIA Tesla V100, Tesla P100, GTX 1080 Ti, Intel PAC, Xeon Skylake 8180, Xeon Phi KNL 7250, and Xeon Phi KNM 7295 performance in terms of total operations-per-second. Similar metrics were not provided by Mipsology for Zebra. Additional information is included about another framework for accelerating CNNs on Micron boards featuring Xilinx FPGAs known as Snowflake; however, hardware was not available to benchmark [32].

Figure 2 shows the performance of GoogLeNet and AlexNet across the different frameworks and devices at their respective maximum performing batch sizes. The xfDNN framework fails to run with AlexNet, so data is not present. Similarly, the Tesla P100 fails to run the custom-variant GoogLeNet at a batch size greater than 1, limiting throughput.

Figure 3 shows the efficiency characteristics in terms of performance-per-Watt of each network across varying frameworks and platforms. The total device power (TDP) for the devices in the study can be found in Table 1. For the Zebra framework, documentation claims the maximum power consumption is below 40W, where the TDP for the XCVU9P FPGA is around 65W [33]. As Xilinx gives no guarantee about power consumption, we use the TDP of the FPGA, 65W, for xfDNN. AWS does not provide access to FPGA power information. We use TDP to compare each device because of the potential each device has to use peak power.

Table 1 Maximum FP16 OPS Performance of Frameworks/Devices and Power Consumption

Device Configuration	FP16 Giga-Operations-per-Second-per-Core	Total Number of Cores	FP16 Giga-Operations-per-Second	Device Power (W)
xfDNN v2 – 2 PE [18]	1702.4	2	3,404.8	65
xfDNN v2 – 4 PE [18]	896	4	3,584	65
Mipsology Zebra (2018) [15]	N/A	6	N/A	40
Tesla V100 [19]	195	640 (Tensor)	125,000 (Tensor)	300
Tesla P100 [34]	5.2	3,584	18,700	300
GTX 1080 Ti [31]	3.2	3,584	11,340 (upgrade FP32)	250
Snowflake – 512-510 [32]	0.37	512	191	24
Snowflake – 1k-511 [32]	0.5	1,024	512	48
Snowflake – 1k-852 [32]	0.5	1,024	512	150
Intel PAC [16] [35]	N/A	N/A	1,500	45
Xeon Skylake 8180 [25] [36]	80	56	4,480	410
Xeon Phi KNL 7250 [22]	46.2	68	3,141	215
Xeon Phi KNM 7295 [23]	49.5	72	3,564	320

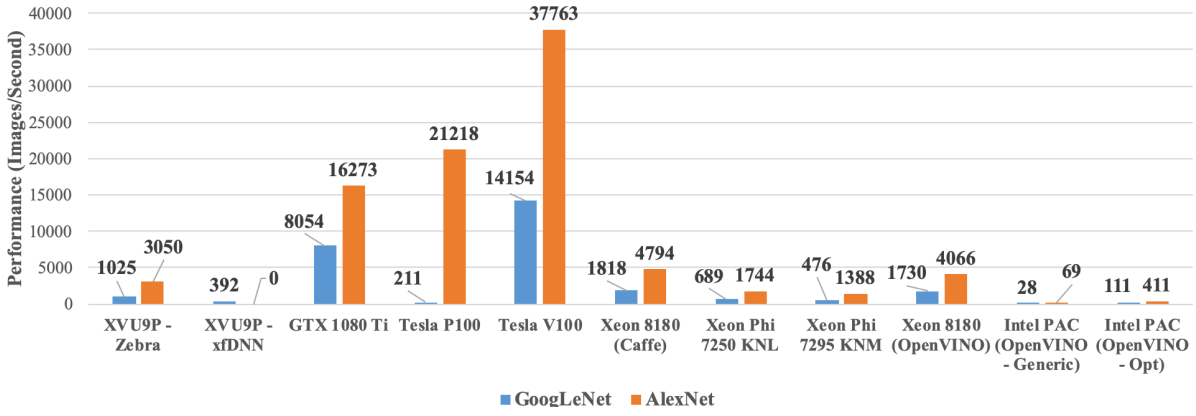

Figure 2 Maximum Throughput Performance of Frameworks/Devices at Batch Size for Maximum Performance

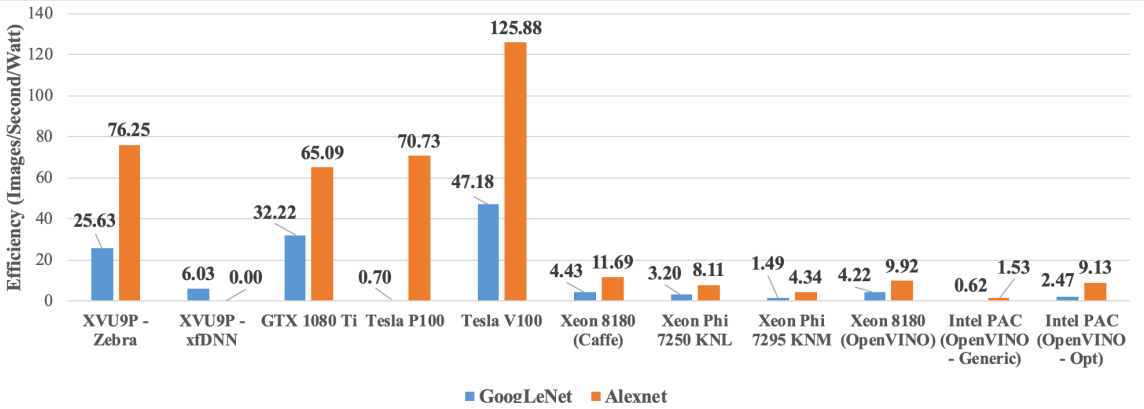

Figure 3 Efficiency of Frameworks/Devices at Batch Size for Maximum Performance

VI. Discussion

When comparing the neural networks, GoogLeNet and AlexNet, we can see that AlexNet consistently achieves higher performance than GoogLeNet. This higher performance of AlexNet is because of the shorter latency AlexNet has from input to output layers having only five layers compared to GoogLeNet's 14 layers, allowing for faster image classification. Since AlexNet has fewer layers than GoogLeNet, more RAM can be used for the images being classified, which allows for larger batch sizes. The smaller number of layers gives AlexNet higher performance at the slightly lower accuracy. For this application, we have observed the average Top-1 accuracies of AlexNet and GoogLeNet to be similar as 95.3% and 96.5%, respectively.

A. Device Performance

Comparing Xilinx FPGA frameworks, we can see that Mipsology Zebra outperforms xfDNN across both neural networks. As Zebra also provides much more portability than xfDNN, this feature gives a greater advantage to Zebra over xfDNN for accelerating machine-learning apps on Xilinx FPGAs.

According to the results from the GPUs, the performance of the Tesla V100 is significantly higher than the other GPUs in the study. It is clear that the parallel nature of the Tesla V100, and its architecture featuring Tensor cores, greatly helps throughput performance.

From the CPU results, the Xeon device has a significant performance advantage over the Xeon Phi devices, even though it features a smaller number of cores. However, the cores of the Xeon devices operate at a maximum of 3.8GHz versus 1.6GHz of both the KNL and KNM Xeon Phi devices [22] [23] [25]. Additionally, comparing the Xeon Phi devices, KNL slightly outperforms KNM consistently between both CNNs and batch sizes. As KNM is targeted at acceleration machine-learning apps, this result is concerning [37]. Although, preliminary data shows that backward-pass timing, as opposed to forward-pass or inferencing, on the KNM significantly outperforms KNL across CNNs and batch sizes. This data means that the KNM devices show much better performance in terms of CNN training than inferencing. In all cases of the CPU testing, the system did not run out of memory, but this out-of-memory error was a source of crashes in the FPGA and GPU cases. The overall app execution time became very slow and thus we limited the batch size to 2048 since no other framework or device achieved more.

For the Intel PAC FPGA results, there is an advantage to using the network-specific optimized OpenVINO binaries over the generic variant at every batch size and when comparing maximum performance. The generic binary is limited to eight images-per-batch which hurts overall parallelism when trying to accelerate a custom CNN with OpenVINO. Next, we observe the Xeon Skylake CPU's performance with OpenVINO against the PAC FPGA. We can see the Xeon CPU outperforms the PAC FPGA at every batch size and in terms of maximum performance. The OpenVINO network-specific optimized binaries for the PAC FPGA are limited to a maximum batch size of 96 images. This limit, again, hurts overall parallelism when trying to accelerate one of these CNNs on the PAC FPGA.

In order to get an understanding of the performance characteristics of both Caffe and OpenVINO, we compared the maximum performance of each framework on the Xeon Skylake CPU. We can see from the results that both frameworks perform similarly. OpenVINO has a slight performance advantage over Caffe when running GoogLeNet; however, Caffe has a more significant performance advantage over OpenVINO when running AlexNet. From this comparison, we can conclude that the framework implementations are similar enough to justify a comparison of the PAC FPGA results with the other devices in the study. Observing that Zebra on the FPGA outperforms OpenVINO on the Intel PAC FPGA by an average of 8.3×, the OpenVINO framework is not competitive when accelerating CNNs on FPGAs. This performance gap could be due to the technology node disparity, 16-nm and 20-nm for the XVU9P and Intel PAC respectively, and the limited batch sizes supported with OpenVINO.

Comparing the results of the Xilinx FPGA using Zebra and the Tesla V100 using cuDNN, we see there is a large disparity in the performance between the Tesla V100 and the FPGA. Our results indicate that the FPGA framework would need to consume less than 22W of power in order to be more efficient in terms of performance-per-Watt. To calculate the power required for better efficiency, the performance of the Zebra framework is divided by the efficiency of the Tesla V100. When comparing the performance of individual cores of xfDNN and the Volta architecture, the xfDNN cores can achieve higher theoretical performance. The main reason why the performance gap is so large is that xfDNN only instantiates four cores on the FPGA whereas, the Volta architecture contains 80 streaming multiprocessors, each with eight Tensor cores.

When comparing the Zebra performance to the Xeon Skylake CPU, we can see that there is less of a disparity between the two than what was observed with Zebra and the V100. However, the Xeon device still significantly outperforms the Zebra framework. Naturally, this performance of the Xeon device means that the V100 is expected to outperform the Xeon device, which is what is observed in the next comparison. The V100 outperforms the Xeon device by an average factor of 2.6×.

B. *Device Efficiency*

In terms of the efficiency of each device and framework, we can see the V100 significantly outperforms every other device and framework, even at a large power package of 300W. This efficiency at 300W shows how much higher the V100 performs compared to each of the other devices and frameworks.

Interestingly, the FPGA using Zebra has similar to greater performance-per-Watt capabilities against both Pascal-based architectures, the Tesla P100 and GTX 1080 Ti. The large performance disparity between the Pascal and Volta architectures is due to the inclusion of the Tensor cores in the Volta architecture. The development of these frameworks for accelerating CNNs on FPGAs is clearly relevant since they are capable of being more efficient than general-purpose architectures that lack specific accelerators for this domain.

According to the results, the Intel products, including all Xeon and Xeon Phi CPUs, as well as the PAC FPGA, perform the worst in terms of efficiency across Caffe, OpenVINO, and different CNNs. These results are a magnitude less than the rest of the Xilinx and NVIDIA device results, besides xfDNN, which perform at around the same efficiency as the Intel devices. The main reason for this poor efficiency on the CPU side is the large power packages of the CPUs, similar to GPUs, without the performance to match the GPUs. In the case of OpenVINO and the PAC FPGA, the power package is one of the lowest in the study; however, the performance is not close to any of the other devices and frameworks.

VII. Conclusions

In this research, a machine-learning inferencing app was developed to leverage many different HPC architectures and frameworks, designed to compare these technologies to one another. CNNs such as AlexNet and a custom 14-layer version of GoogLeNet were used to classify handwritten Chinese characters. The Caffe framework was used to leverage Xilinx FPGAs, NVIDIA GPUs, and Intel Xeon and Xeon Phi CPUs. The Intel platform-agnostic OpenVINO framework was used with Intel PAC FPGAs and additionally with Intel Xeon CPUs to gain an understanding of OpenVINO versus Caffe performance.

It is clear that the Tensor cores significantly accelerate the performance of machine-learning inference on NVIDIA GPUs. Without significant improvements in performance to FPGA frameworks for accelerating CNNs, FPGAs may need to add additional hardware, similar to Tensor cores, to be more competitive in the machine-learning domain. In fact, the next-generation Xilinx architecture, known as Versal, is designed with new "AI engines" consisting of long instruction word and single instruction, multiple data processing engines [38].

Intel devices and frameworks are also lacking in the machine-learning inferencing domain. CPUs are the most general-purpose device in the study, posing significant overhead, especially in terms of efficiency. It is challenging for CPUs to tailor to one domain as they serve all computing domains. Being the worst in every category, the OpenVINO framework for PAC FPGAs needs significant improvements in order to be competitive in this domain as well.

Some of these performance disparities may also be due to the technology node of each device. The Volta architecture is the smallest at 12-nm. The worst-performing architecture in this study, the Arria 10, is also the largest at 20-nm. This factor can of course have significant implications on performance of the devices.

Overall, GPUs dominate performance and efficiency when accelerating CNNs with Caffe. The next most efficient devices, Xilinx FPGAs, need significant improvements for accelerating machine-learning apps, especially since they currently cannot perform training. Mipsology has mentioned that they do plan to support training in the future [15]. The Tesla V100 has significantly better performance with both AlexNet and GoogLeNet at 12.38× and 13.81×, respectively, over Zebra's performance. Similarly, the Tesla V100 has better efficiency with both AlexNet and GoogLeNet at 1.65× and 1.84×, respectively, when compared to Zebra's efficiency. Although the Versal architecture is not set to be released until late 2019, data from Xilinx shows Versal performing at 2× over the Tesla V100 using GoogLeNet for machine-learning inference with maximum batch size [38]. This architecture, in combination with the next release of xfDNN v3 and Zebra, has potential to make FPGAs more competitive with GPUs for machine-learning inference and significantly more efficient.

This research has provided insight on throughput performance and efficiency characteristics of a practical, deep-learning app across many different architectures and frameworks. The development of these apps can be continually used as architectures and frameworks evolve to understand their respective, relative performance. As focus shifts from machine-learning training to inferencing acceleration, this research provides critical information to prepare app acceleration for the future of the machine-learning domain.

Acknowledgments

This work was supported by SHREC industry and agency members and by the IUCRC Program of the National Science Foundation under Grant No. CNS-1738783. We would also like to thank AWS, Google Cloud, Pitt CRC, and Intel vLab for access to their computing resources in the cloud. Finally, we would like to thank Dr. Bryant Lam of the NSA.

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] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," in ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2015.

[3] K. Ovtcharov, O. Ruwase, Et. Al, "Accelerating Deep Convolutional Neural Networks Using Specialized Hardware," Microsoft, February 2015. [Online]. Available: https://www.microsoft.com/en-us/research/publication/accelerating-deep-convolutional-neural-networks-using-specialized-hardware/ . [Accessed November 2018].

[4] C. Farabet, C. Poulet, J. Y. Han and Y. LeCun, "CNP: An FPGA-based processor for Convolutional Networks," International Conference on Field Programmable Logic and Applications, September 2009.

[5] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in IEEE International Symposium on Circuits and Systems, Paris, 2010.

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

[7] Google, "Using GPUs for Training Models in the Cloud," 10 October 2018. [Online]. Available: https://cloud.google.com/ml-engine/docs/tensorflow/using-gpus. [Accessed November 2018].

[8] J. Fowers, G. Brown, P. Cooke, G. Stitt, "A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-Window Applications," in ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), 2012.

[9] L. Deng, D. Yu, "Deep Learning: Methods and Applications," Foundations and Trends in Signal Processing, vol. 7, no. 33-34, pp. 1-99, 2014.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in IEEE, 1998.

[11] X. Glorot, A. Bordes, Y. Bengio, "Deep Sparse Rectifier Neural Networks," in Fourteenth International Conference on Artificial Intelligence and Statistics (PMLR), 2011.

[12] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Neural Information Processing Systems, vol. 25, no. 2, 2012.

[13] C. Szegedy, W. Liu, Et. Al, "Going Deeper With Convolutions," in IEEE Computer Vision and Pattern Recognition (CVPR), 2015.

[14] Xilinx, "Adaptive Inference Acceleration," November 2018. [Online]. Available: https://www.xilinx.com/applications/megatrends/machine-learning.html. [Accessed November 2018].

[15] Mipsology, "Zebra," August 2017. [Online]. Available: http://www.mipsology.com/zebra.html. [Accessed November 2018].

[16] Intel, "OpenVINO Whitepaper," November 2018. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/intel-vision-accelerator-design-with-FPGA-wp.pdf. [Accessed November 2018].

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

[18] E. Delaye, "Integrating AI into Your Accelerated Cloud Applications," 2018.

[19] NVIDIA, "NVIDIA TESLA V100 GPU ARCHITECTURE," August 2017. [Online]. Available: http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf. [Accessed November 2018].

[20] NVIDIA, "NVCaffe," November 2018. [Online]. Available: https://docs.nvidia.com/deeplearning/dgx/caffe-user-guide/index.html. [Accessed November 2018].

[21] Google, "AI & Machine Learning Products," November 2018. [Online]. Available: https://cloud.google.com/tpu/. [Accessed November 2018].

[22] Intel, "KNL Product Specifications," 2018 November. [Online]. Available: https://ark.intel.com/products/94035/Intel-Xeon-Phi-Processor-7250-16GB-1_40-GHz-68-core. [Accessed 2018 November].

[23] Intel, "KNM Processor Specification," November 2018. [Online]. Available: https://ark.intel.com/products/128690/Intel-Xeon-Phi-Processor-7295-16GB-1-5-GHz-72-Core-. [Accessed November 2018].

[24] Intel, "Intel Knight's Mill Microarchitecture," November 2018. [Online]. Available: https://en.wikichip.org/wiki/intel/microarchitectures/knights_mill.

[25] Intel, "Skylake Processor Specificaitons," November 2018. [Online]. Available: https://ark.intel.com/products/120496/Intel-Xeon-Platinum-8180-Processor-38-5M-Cache-2-50-GHz-. [Accessed November 2018].

[26] Intel, "Intel Optimized Caffe*," November 2018. [Online]. Available: https://software.intel.com/en-us/articles/training-and-deploying-deep-learning-networks-with-caffe-optimized-for-intel-architecture. [Accessed November 2018].

[27] Intel, "OpenVINO," November 2018. [Online]. Available: https://software.intel.com/en-us/articles/OpenVINO-InferEngine. [Accessed November 2018].

[28] C.-L. Liu, F. Yin, D.-H. Wang, Q.-F. Wang, "Online and Offline Handwritten Chinese Character Recognition: Benchmarking on New Databases," Pattern Recognition, vol. 46, no. 1, pp. 155-162, 2013.

[29] S. Lai, L. Jin, W. Yang, "Toward high-performance online HCCR: A CNN approach with DropDistortion, path signature and spatial stochastic max-pooling," Pattern Recognition Letters, vol. 89, February 2017.

[30] R. DiCecco, G. Lacey, Et. Al, "Caffeinated FPGAs: FPGA framework For Convolutional Neural Networks," in International Conference on Field-Programmable Technology (FPT), 2016.

[31] NVIDIA, "GEFORCE GTX 1080 Ti," August 2017. [Online]. Available: https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080-ti/ . [Accessed November 2018].

[32] FWDNXT, "Snowflake," 2018. [Online]. Available: http://www.fwdnxt.com . [Accessed November 2018].

[33] Xilinx, "Xilinx Power Estimator," November 2018. [Online]. Available: https://www.xilinx.com/products/technology/power/xpe.html. [Accessed November 2018].

[34] NVIDIA, "NVIDIA TESLA P100 ARCHITECTURE," August 2017. [Online]. Available: https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-PCIe-datasheet.pdf . [Accessed November 2018].

[35] Intel, "Intel PAC," 16 October 2018. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/documentation/hhf1507759304946.html#vjb1508359354353. [Accessed November 2018].

[36] J. Hennessy, D. Patterson, "Computer Architecture: A Quantitative Approach," 6 ed., San Francisco, CA: Morgan Kaufmann, 2017.

[37] Intel, "Knight's Mill: New Intel Processor for Machine Learning," 2017. [Online]. Available: https://www.hotchips.org/wp-content/uploads/hc_archives/hc29/HC29.21-Monday-Pub/HC29.21.40-Processors-Pub/HC29.21.421-Knights-Mill-Bradford-Intel-APPROVED.pdf. [Accessed November 2018].

[38] Xilinx, "Versal: The First Adaptive Compute Acceleration Platform (ACAP," 2 October 2018. [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp505-versal-acap.pdf. [Accessed November 2018.