# Deep-Learning Inferencing with High-Performance Hardware Accelerators

by

**Luke Edwin Kljucaric**

B.S. Computer Engineering, University of Pittsburgh, 2016

Submitted to the Graduate Faculty of

the Department of Electrical and Computer Engineering

Swanson School of Engineering in partial fulfillment

of the requirement for the degree of

Master of Science in Electrical Engineering

University of Pittsburgh

2018

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

**Luke Edwin Kljucaric**

It was defended on

November 28, 2018

and approved by

Samuel Dickerson, Ph.D., Director and Assistant Professor
Department of Electrical and Computer Engineering

Jun Yang, Ph.D., Whiteford Professor
Department of Electrical and Computer Engineering

Thesis Advisor, Committee Chair: Alan D. George, Ph.D., Mickle Chair Professor of ECE
Department of Electrical and Computer Engineering

**Deep-Learning Inferencing with High-Performance Hardware Accelerators**

Luke Kljucaric, M.S.

University of Pittsburgh, 2018

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 better understand that data and increase autonomous processing. As FPGAs become more readily available through 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, which are used in many machine-learning apps, have started emerging for accelerated-application development. This thesis aims to compare the performance of these emerging frameworks on two commonly-used convolutional neural networks, GoogLeNet and AlexNet. Specifically, handwritten Chinese character recognition is benchmarked across multiple currently available 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× and 9.3× better performance and efficiency, respectively, over the OpenVINO framework available for Intel FPGAs. Although the Zebra framework on the Xilinx VU9P showed better 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.

## Table of Contents

## List of Tables

## List of Figures

## Preface

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.

I would like to thank many of those at the different cloud computing centers AWS, Google Cloud, Pitt CRC, and Intel vLab. Without the hardware resources made available through various research and trial programs, this research would not have been possible. Critical support for software and device configuration was also provided by the different cloud platforms which helped accelerate this research.

Additionally, I would like to thank Dr. Bryant Lam who helped guide this research focus. It was his guidance that led to the exploration of CNNs on FPGAs. The HCCR application was used because of his desire to focus on the CNN application towards optical character recognition (OCR).

Finally, I would like to thank all of those in NSF SHREC who have helped support and review this thesis. Specifically, I would like to thank my advisor, Dr. Alan George, who has spent his valuable time reviewing and guiding this thesis as well.

# 1.0 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 are trained on a certain set of data in order to process new, unclassified data autonomously. Image classification is one example app in a broad range of machine-learning apps. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a competition where algorithms compete to classify images at a large scale with high accuracy [1]. Object detection is another example of a machine-learning app in which a computer attempts to extract specific objects from a scene. YOLO is a machine-learning model used for real-time object detection to classify and predict bounding boxes around objects in one execution step [2]. Convolutional neural networks (CNNs) have seen success winning ILSVRC multiple years and are used in the latest YOLO 9000 algorithm [3] [4].

Machine-learning apps for image processing often use CNNs in their models. CNNs are attractive for these types of applications because they require minimal preprocessing in comparison to other methods in order to extract image features [5]. 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 it very amenable for GPU and FPGA acceleration.

Traditional acceleration of CNNs on FPGAs has been done by implementing a specific neural-network processor in hardware [6] [7] [8] [9]. 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 TensorFlow, have been emerging 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 these FPGA frameworks' performance against other architectures. Many different toolkits and frameworks exist to leverage Intel devices in different ways. It is challenging, yet important, to be able to compare different frameworks on different architectures.

In this thesis, 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, efficiency, and cost-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 when designing machine-learning-oriented high-performance computing (HPC) systems.

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

## 2.1 Machine-Learning Inference

In common machine-learning apps, there are at least two distinct phases: training and inferencing. Training a CNN involves determining weight values in a network. CNNs are comprised of different layers that have certain weight values, which activate different neurons based on the input to the neural network. The specific weight values for a CNN vary based on the application, or input dataset. During a training phase, a set of labeled images is classified using a neural network. Then, the actual output is compared to the expected output and this difference in outputs is known as the loss. This loss is then passed to the backpropagation algorithm, which incrementally updates the weights of the network to categorize the input data more appropriately. This process is repeated many times until a minimum loss value is reached. At the same time, it is important to know when to cease training. If the network is trained too long on the input data, the model could "overfit" the training data; meaning, it may have high accuracy for the training data but have poor accuracy on new or testing data. The reader is referred to [10] for more information on the backpropagation algorithm.

With the trained network, inferencing can be performed on new data. An unknown image is given to the CNN and the output neurons are activated based on the trained weights. The CNNs in this study's output neurons are all activated with certain probabilities [11]. The higher the activation, the higher the probability that the specific image belongs to a certain class. The vast majority of the lifespan of an image-classification app is spent in the inferencing stage, since training CNNs is typically a one-time event. Thus, this thesis focuses on machine-learning inferencing due to its prevalence and because backpropagation has no support on FPGAs with the frameworks studied in this research at the time of this writing.

CNNs have typically been accelerated using GPU architectures because of the embarrassingly-parallel nature of the computations [12]. GPUs are single-instruction, multiple-data architectures designed to perform many of the same computations in parallel on different data at the same time. This makes performing convolutions and matrix multiplications on this architecture straightforward to parallelize. Studies have shown that FPGAs are capable of achieving better performance-per-watt than GPUs for various applications [13]. FPGAs can potentially accelerate CNNs with a similar datapath to GPUs, but they require much less power than what is seen on today's high-end GPUs.

## 2.2 Deep Learning

Deep learning is a specific type of application within the field of machine learning. What defines machine learning as deep is the use of deep neural networks (DNNs), meaning the network has multiple "hidden" layers with each layer's output providing input to the next layer [14]. This research uses two commonly-used deep CNNs (DCNNs) known as AlexNet and GoogLeNet.

AlexNet is a 7-layer CNN based on the LeNet design [15]. It was the one of the first CNNs to use the rectified linear unit (ReLU), an unbounded activation function as opposed to sigmoid, tanh, etc. The ReLU attempts to mimic the neuron activity in the human brain where strongly activated neurons diminish the effects of surrounding neurons. This behavior typically allows for better feature extraction and faster training times [16]. AlexNet, seen in Figure 1, was designed with a dual data path specifically so that each data path could be trained using a separate GPU [17].

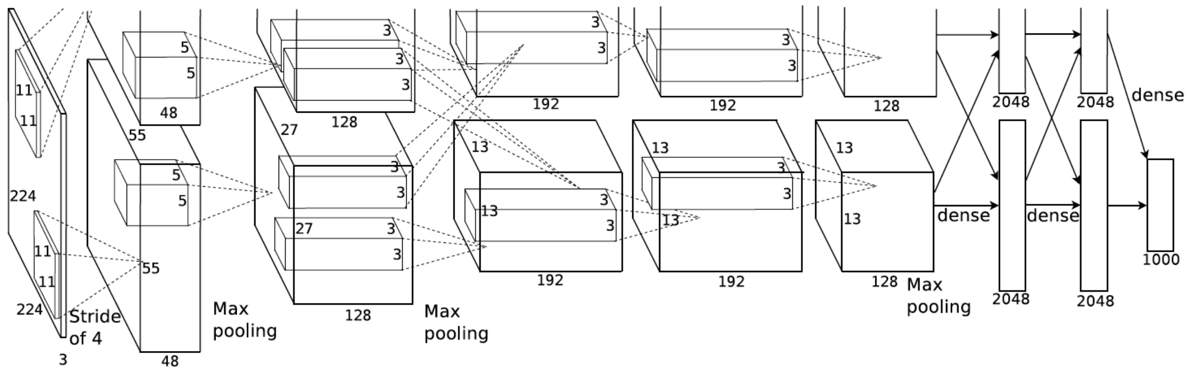

**Figure 1** Dual Data-path Design of AlexNet [17]

GoogLeNet is a 22-layer CNN based on work from LeCun and Krizhevsky [15] [17]. It features the use of “inception” layers which attempt to process more image area while retaining small image details by performing multiple convolutions of different sizes in one layer. This inception feature attempts to reduce the total number of calculations per prediction for use with smartphones [18].

## 2.3 Caffe Machine-Learning Framework

Caffe is an open-source framework for developing machine-learning apps. It was developed by The University of California Berkeley Vision and Learning Center (BVLC). It provides support for performing machine-learning training and inferencing on both CPUs and GPUs. Caffe supports batched classification which allows for multiple images in one batch to be classified in parallel, increasing the overall classification performance [19]. In this thesis, we use Caffe instead of other machine-learning frameworks because the CNNs acceleration frameworks for FPGAs currently only fully support Caffe models [20] [21] [22]. Limited support exists for most other frameworks on FPGAs.

## 2.4 Cloud Computing Platforms

Amazon Web Services (AWS) and Google's Cloud platform are commercial cloud-computing platforms with many different exotic devices for on-demand usage. Expensive and hard-to-obtain devices become readily available through AWS and Google Cloud.

Additionally, customers do not need to configure hardware or software environments in order to properly use these devices. A range of machine images are available to use on these platforms that come preconfigured with software and hardware environments. AWS provides Xilinx FPGAs and NVIDIA GPUs, while Google Cloud provides a broader range of NVIDIA GPU architectures than AWS, but it lacks FPGAs.

Other academic clusters such as The University of Pittsburgh Center for Research Computing (CRC) and The Intel vLab Academic Cluster attempt to achieve a similar goal. While not typically profit-oriented, these clusters allow exploration of hardware and software tools in a shared computing environment. Users typically submit jobs to run in a shared resource cluster, waiting for open hardware. This job submission methodology is in contrast to the commercial clusters like AWS and Google Cloud where a user can access a specific machine on-demand. Interactive sessions are common in both academic and commercial clusters; however, there may be a wait to use open resources in an academic cluster.

The academic clusters often have accelerators not featured in commercial clusters because of their specific research requirements. CRC features different Intel Xeon architectures with Omni-path connections and InfiniBand networks as well as many different GPU architectures [23]. The vLab cluster features Intel architectures not found on other clusters. These devices include high-performance Intel Xeon Skylake CPUs, Intel Xeon Phi Knight's Landing and Knight's Mill many-core CPUs, Xeon Broadwell plus FPGA (Arria 10) packages, as well as newer Intel Programmable Acceleration Cards (PACs) that feature Arria 10 FPGAs.

## 2.5 Field-Programmable Gate Arrays (FPGAs)

Unlike fixed-logic devices like 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 datapaths often give FPGAs an advantage over fixed-logic devices in terms of performance-per-watt. Many different datapaths can be instantiated onto the FPGA in parallel, which makes it amenable for accelerating machine-learning apps that use CNNs. Although the data-paths on FPGAs are typically longer than fixed-logic devices, the energy-efficiency comes from the parallelism in the design [13]. The

XCVU9P board, which features Xilinx's 16-nm Ultrascale+ architecture, that is featured on AWS is one of the newest FPGA boards Xilinx currently offers.

The resources on vLab feature two different acceleration models using FPGAs. In both cases, they feature Intel's 20-nm Arria 10 architecture. The first is a Xeon Broadwell plus Arria 10 package. It features a Xeon E5-2643 v4 CPU on the same package as an Arria 10 GX1150 FPGA (10AX115U) connected via Intel's Ultra Path Interconnect. The second is the Intel PAC which features an Arria 10 GX FPGA (10AX115N) connected to the system via PCIe x8.

## 2.6 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 [20]. 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 differences between the 28×32- and 56×32-labeled cores are the 56×32-labeled core can process higher resolution images and process images at a lower latency, whereas the 28×32-labeled core is designed for maximum throughput [24]. 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.

## 2.7 Mipsology Zebra

Zebra is also 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 [21]. 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 6 cores. Additionally, like xfDNN, the main application makes calls to the Caffe API which then references the Zebra framework.

## 2.8 Graphics Processing Units (GPUs)

GPUs have been 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. Google Cloud provides access to many of the different NVIDIA architectures such as Pascal and Volta. With Volta being a new architecture, many production systems still employ Pascal-based GPUs, thus there is a demand to see how FPGA frameworks for accelerating CNNs compare in performance to older Pascal-based GPUs like the 16-nm Tesla P100. AWS, as well as Google Cloud, also provides access to

NVIDIA's latest server-grade Tesla GPU known as the V100. 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 six hundred "Tensor cores" which 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 [25].

## 2.9 NIVIDIA CUDA Deep Neural Network Library (cuDNN)

NVIDIA has developed its own framework for accelerating deep neural networks on NVIDIA GPUs using CUDA known as cuDNN. The cuDNN 7.4 framework from NVIDA accelerates machine-learning apps by taking advantage of the Tensor cores in the Volta architecture. Through the University of Pittsburgh's Center for Research Computing (CRC), the NVIDIA GTX 1080 Ti, based on the 16-nm Pascal architecture, uses the cuDNN 6 framework due to lack of support for cuDNN 7.X at CRC. Like the Xilinx-based FPGA frameworks, Caffe runs on the CPU which then references cuDNN for GPU offloading and acceleration. Backpropagation, certain activation functions, among other things are supported for GPU acceleration with cuDNN where they are not supported by the FPGA frameworks [26].

## 2.10 NVIDIA Optimized Caffe (NVCaffe)

A fork of Caffe has been developed by NVIDIA known as NVCaffe. This version of Caffe has all the functionality of Caffe which makes existing Caffe-based apps portable to it. NVIDIA has optimized this version of Caffe to perform best on NVIDIA GPUs [27].

## 2.11 Google Tensor Processing Unit (TPU)

Google's 28-nm TPU is an architecture specifically aimed at accelerating machine-leaning apps built with the TensorFlow framework. This architecture, like Volta from NVIDIA, uses the idea of making the matrix or "tensor" operations fast and efficient. A single Cloud TPU v3 Beta has a maximum performance of 420 Teraflops [28], compared to the maximum tensor performance of 125 Teraflops of NVIDIAs Tesla V100 [25].

## 2.12 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 performance in the machine-learning domain against other HPC devices. Intel's line of many-core CPUs, called Xeon Phi, are featured on vLab. There are two different architectures, the 14-nm Knight's Landing (KNL) 7250 and the 14-nm Knight's Mill (KNM) 7295 which feature 68 cores and 72 cores, respectively [29] [30]. The key changes from KNL to KNM are the inclusion of three, machine-learning specific, operations. These include a single-precision fused multiply accumulation, vector neural network, and a doubleword/quadword vector population count [31].

These CPUs also support hyper-threading up to 4 threads-per-core [29] [30]. This ability makes many-core CPUs able to handle a thread count of 272 and 288 for KNL and KNM, respectively. With such a high degree of parallelism, it is important to understand how this architecture is able accelerate parallel matrix-operations.

The vLab cluster also features high-end Xeon Skylake CPUs. Specifically, vLab features a dual-socket machine with two 14-nm Xeon Platinum 8180 processors with 28 cores each. Each processor has the ability to support hyper-threading up-to 2 threads-per-core [32]. This hyper-threading ability gives this machine the ability to support up to 112 threads at once, much higher than traditional multi-core CPUs.

## 2.13 Processor Metrics

Calculating processor metrics is one way of evaluating different architectures at a glance. Computation density (CD) is one such metric that determines the underlying computational capacity of a device. CD measures the theoretical maximum amount of parallel operations-per-cycle. This metric is calculated assuming 50% add and 50% multiply instructions. Specifically, half-precision floating point CD is used to compare the Intel Xeon Phi processors to the rest of the architectures in this study. CD is calculated by multiplying the core frequency times the sum of the number of instructions of type i divided by the cycles-per-instruction (CPI) of instruction type i, as seen in Equation 1.

$$CD = f \times \sum \frac{\boldsymbol{N(i)}}{\boldsymbol{CPI(i)}} \qquad \textbf{2-1}$$

## 2.14 Intel Open Programmable Acceleration Engine (OPAE)

Intel provides and supports many tools that assist in optimally and efficiently using their hardware. OPAE is one such framework that aids in better using Intel FPGAs. OPAE is intended to create an abstraction layer between generation and architecture specific details of FPGAs and user apps in order to quickly develop apps and allow for portability between devices [33]. The relationship between OPAE and the rest of the Intel Acceleration Stack can be seen in Figure 2.

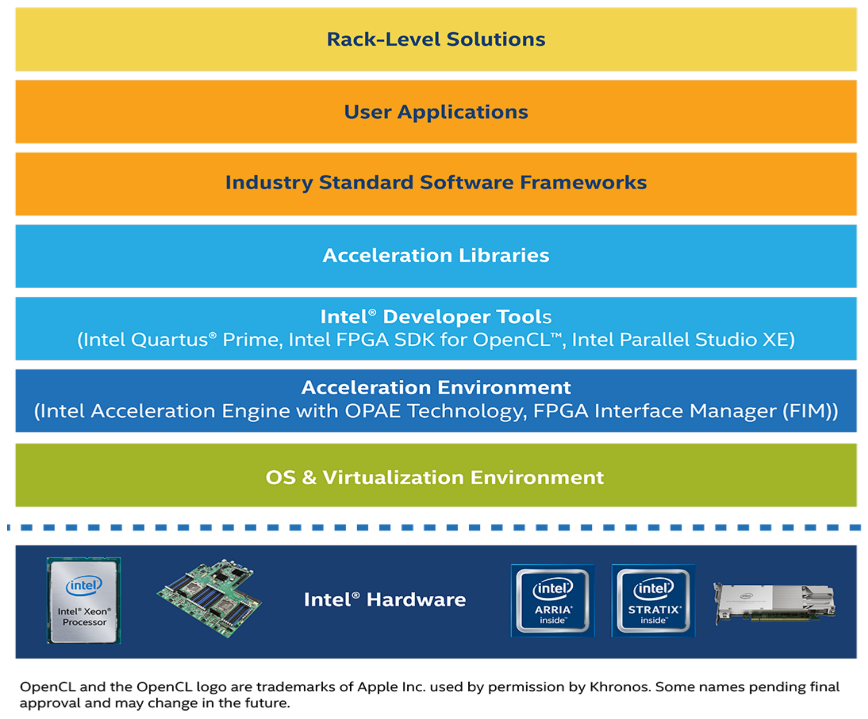
Rack-Level Solutions
User Applications
Industry Standard Software Frameworks
Acceleration Libraries
Intel® Developer Tools
(Intel Quartus® Prime, Intel FPGA SDK for OpenCL™, Intel Parallel Studio XE)
Acceleration Environment
(Intel Acceleration Engine with OPAE Technology, FPGA Interface Manager (FIM))
OS & Virtualization Environment
Intel® Hardware
Intel® Xeon® Processor
ARRIA inside
STRATIX inside
OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. Some names pending final approval and may change in the future.

**Figure 2** Intel Acceleration Stack Featuring OPAE [33]

OPAE can be configured differently based on the apps targeting the FPGA. It features RTL-optimized and OpenCL-optimized configurations. The main goal of OPAE is to simplify software acceleration on FPGAs by allowing developers to compile and synthesize their design once targeting OPAE and have the ability to port the same design to multiple different FPGA architectures that feature OPAE with minimal effort. OPAE also tries to take advantage of device specific characteristics, such as unique memory interfaces, to take full advantage of devices while still being portable with minimal overhead.

## 2.15 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*. It is integrated with the Intel Math Kernel Library (MKL) optimized for Advanced Vector Extensions instructions in the Intel Xeon and Xeon Phi processors. The MKL helps accelerate CNN computations more efficiently than standard Caffe. It contains all the functionality of Caffe so there is seamless portability of a Caffe-based app [34]. This framework only supports Intel CPUs.

The OpenVINO toolkit from Intel provides another method for accelerating machine-learning apps on Intel CPUs, GPUs, FPGAs, and other accelerators. It also supports heterogenous computations of CNNs across different architectures. Similar to Caffe*, OpenVINO is integrated with the Intel MKL. Although, OpenVINO does not support Caffe API calls, it does support the use of Caffe models. A Caffe model is a file generated from Caffe after training that contains network information like weight values and hyper-parameters. This Caffe model allows for the use of the same networks across different accelerators. Existing Caffe-based apps are not portable to this framework and must be rewritten to support OpenVINO-specific calls. Like xfDNN, OpenVINO provides a model optimizer that takes the Caffe model as input and determines how best to accelerate that model. The model optimizer output is provided to the OpenVINO inference engine for acceleration. The inference engine for the FPGA uses OPAE for FPGA support and programming. Similar to both xfDNN and Zebra, there are only precompiled binaries of the OpenVINO FPGA plugin. Several binaries exist such as a generic, AlexNet, GoogLeNet, and other network-specific variants. Each binary has two versions, one for FP11 and one for FP16 support. For the generic version, the batch size is limited to 8 images. For the network-specific, or optimized variants, the batch size is limited to 96 images. There is limited documentation on the specifics of the different binaries [35].

## 2.16 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 English alphabet, with only 26 characters, is not a difficult enough task with 62 classes, counting lower-case, upper-case, and digits 0-9. 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 [36]. 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.

## 2.17 Caffe-Accelerator Relationship

Many of the frameworks and devices serve similar roles in the acceleration of the HCCR app. Figure 3 visualizes the relationship 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 relationship. 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.

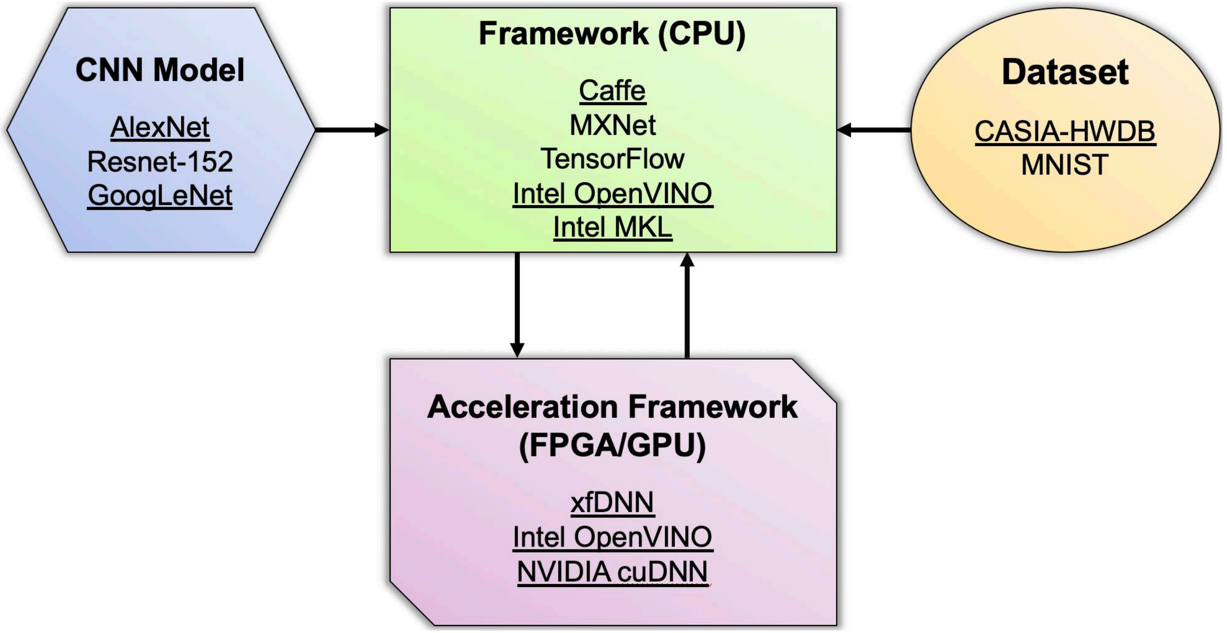

**Figure 3** Caffe-Accelerator Relationship. Underlined Values Used in This Research.

# 3.0 Related Work

Previous work has informed this research by exploring complex classification problems such as handwritten Chinese character recognition. 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. This use of a custom CNN will help avoid any optimization to these networks that may not apply with a custom CNN. 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 [37] [11]. The architectures of the HCCR-AlexNet and HCCR-GoogLeNet can be seen in Figure 4 and Figure 5, respectively. 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 [11]. This variant uses 14 layers as opposed to the 22 layers in the standard version.

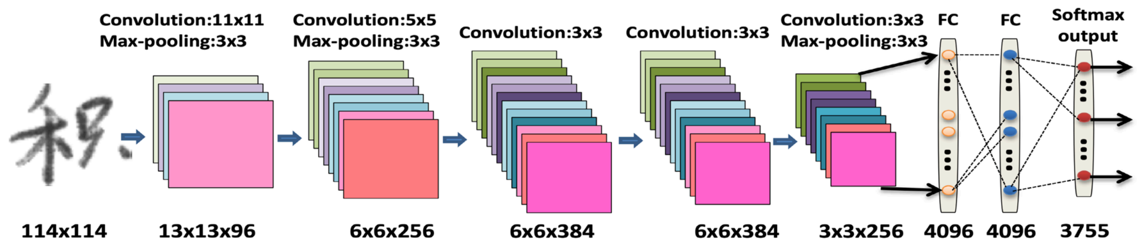

**Figure 4** Architectural Design of HCCR-AlexNet [11] Referred to as Alexnet in This Research

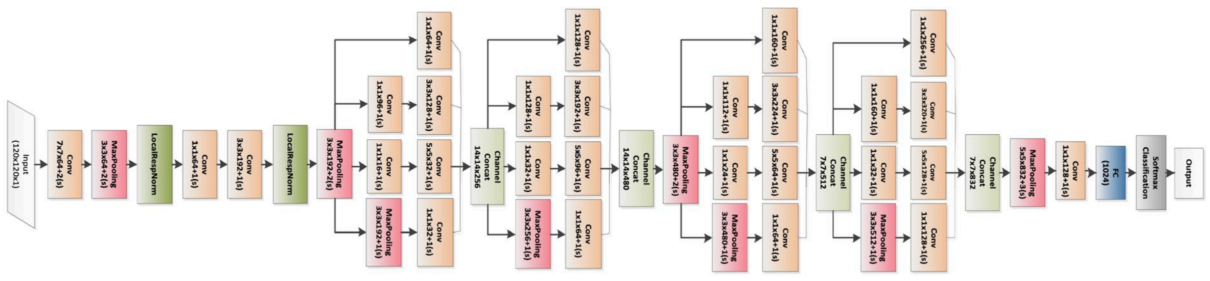

**Figure 5** Architectural Design of HCCR-GoogLeNet [11] Referred to as GoogLeNet in This Research

Other research has explored accelerating CNNs on Xilinx FPGAs in similar fashion. The work done by [38] provides a framework for using FPGAs with Caffe; however, its performance is significantly behind the CPU and GPU variant in the same study. The framework, performing at 50 GOPS, also fails to outperform previous work in [6], which showcases an engine for accelerating CNNs on FPGAs performing at 61.62 GOPS. The work in [6] lacks the compatibility with Caffe apps and models, making it more challenging to use.

Additional research has been done with Catapult and Intel FPGAs [7]. The work showed that the configurable framework was still lacking in performance compared to the NVIDIA Tesla K40 platform based on the Kepler architecture; three generations before the Volta architecture in this study.

# 4.0 Methodology

The main focus of this thesis 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 handwritten Chinese character recognition are used. The first is a C++ app that loads bmp image files from a directory and uses the Caffe API to classify the images in batched mode, evaluating a specific number of images at one 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 run using batch sizes of 1, 16, 32, 64, 128, 256, 512, 1024, and 2048 where applicable. For the OpenVINO results, batch sizes are limited so batch sizes of 2, 4, 8, and 96 are also used in addition to 1, 16, 32, and 64. 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 [11] and [37]. 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 [39].

## 4.1 Xilinx FPGA Acceleration

As mentioned previously, the Xilinx FPGA being used is the Xilinx 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 6 soft cores. The Zebra configuration was not changed as documentation recommends it remaining the same. 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. Then, Caffe-based app 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.

## 4.2 Intel FPGA Acceleration

The Intel FPGA studied is the PAC, which features and Arria 10 GX. The OpenVINO toolkit is used on the Intel FPGA on vLab 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 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.

## 4.3 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 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, 32-bit operation model-optimizer files are created using the OpenVINO model-optimizer since the OpenVINO toolkit does not support 16-bit operations on the CPUs. To run the OpenVINO-based app on the Xeon CPU, no additional steps are required like 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 and KNM provided by vLab, will only run the Caffe-based app using the Intel Optimized Caffe* and the Intel MKL.

## 4.4 NVIDIA GPU Acceleration

The GPUs used in this case 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.

## 5.0 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; meaning, the network should expect to see similar performance between two different images assuming similar resolutions. 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. 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 [40].

**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 [24] | 1702.4 | 2 | 3,404.8 | 75 |
| xfDNN v2 – 4 PE [24] | 896 | 4 | 3,584 | 75 |
| Mipsology Zebra (2018) [21] | N/A | 6 | N/A | 40 |
| Tesla V100 [25] | 195 | 640 (Tensor) | 125,000 (Tensor) | 300 |
| Tesla P100 [41] | 5.2 | 3,584 | 18,700 | 300 |
| GTX 1080 Ti [39] | 3.2 | 3,584 | 11,340 (upgrade FP32) | 250 |
| Snowflake – 512-510 [40] | 0.37 | 512 | 191 | 24 |
| Snowflake – 1k-511 [40] | 0.5 | 1,024 | 512 | 48 |
| Snowflake – 1k-852 [40] | 0.5 | 1,024 | 512 | 150 |
| Intel PAC [22] [42] | N/A | N/A | 1,500 | 45 |
| Xeon Skylake 8180 [32] [43] | 80 | 56 | 4,480 | 410 |
| Xeon Phi KNL 7250 [29] | 46.2 | 68 | 3,141 | 215 |
| Xeon Phi KNM 7295 [30] | 49.5 | 72 | 3,564 | 320 |

## 5.1 Architecture Batch Scaling Performance

The first graph, Figure 6, shows the performance of GoogLeNet and AlexNet with the Caffe-based app across the different FPGA frameworks with varied batch sizes. The xfDNN framework fails to run with AlexNet because of memory access faults at every batch size. The error occurs within xfDNN and, as it is a precompiled binary, the ability to fix such an error is limited. Besides this error and the one specified with GoogLeNet on the Tesla P100, any missing data points in any of the figures are from the devices running out of memory at that batch size or, in the case of OpenVINO on the PAC FPGA, the batch size is limited.

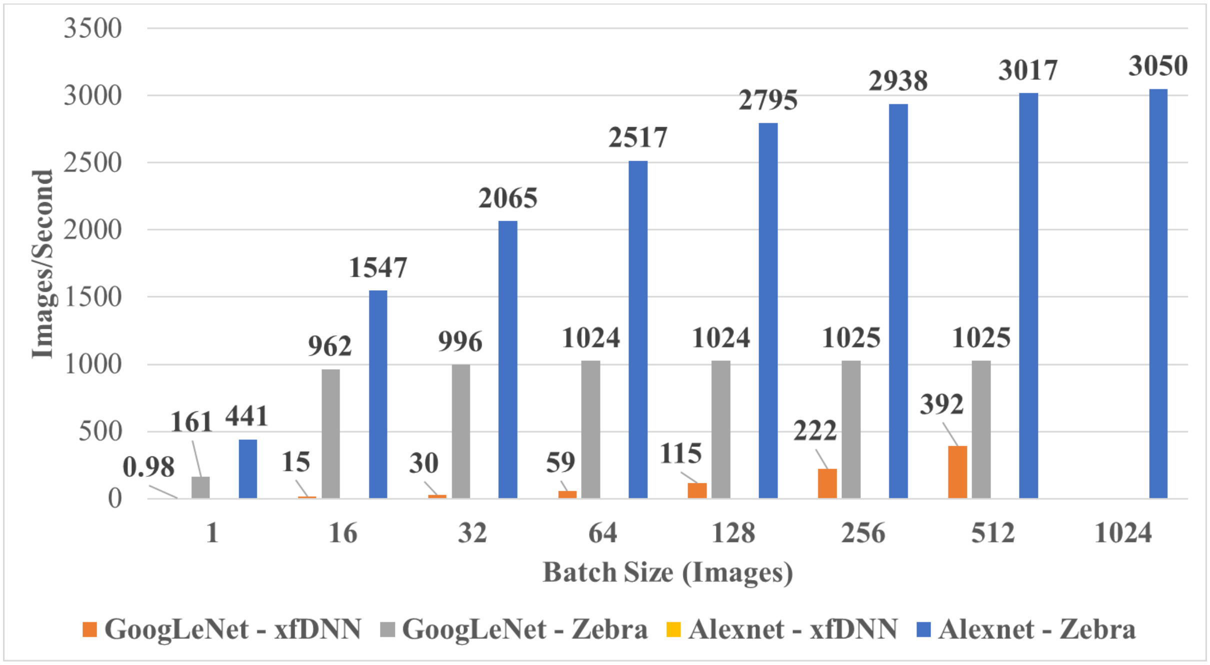

**Figure 6** CNN Performance Across Xilinx FPGA Frameworks with Varied Batch Sizes

The next graph, Figure 7, shows the performance of GoogLeNet and AlexNet with the Caffe-based app across different GPUs with varied batch sizes. As mentioned previously, a similar memory error is seen with cuDNN and GoogLeNet on the P100, where cuDNN is also closed-source; however, in this case, it happens at a batch size of 16, but a batch size of 1 executes until completion.

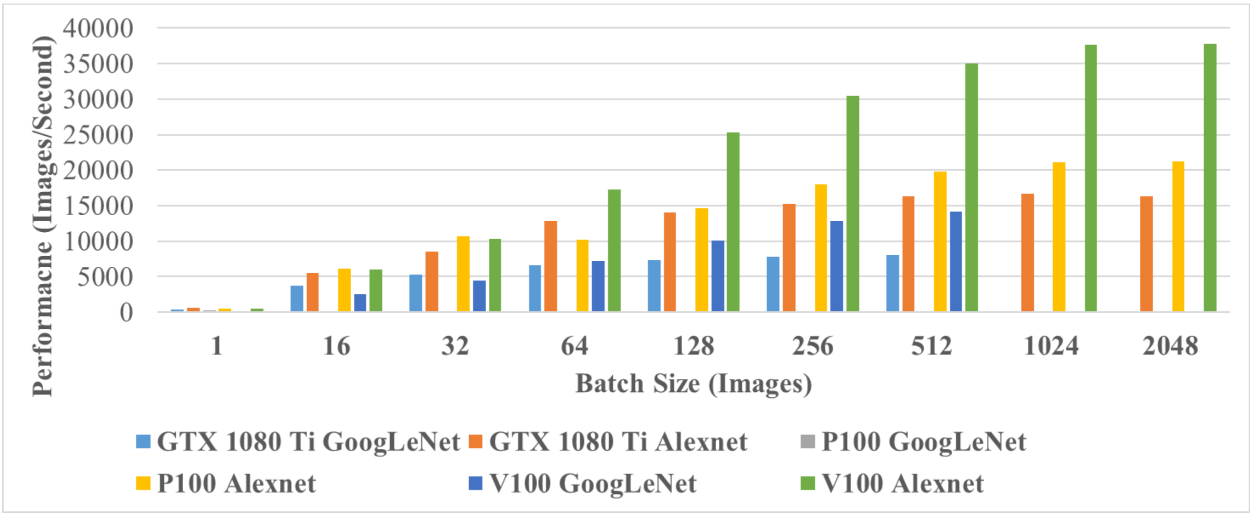

**Figure 7** CNN Performance Across GPU Devices with Varied Batch Sizes (See Appendix for Values)

Figure 8 shows a similar comparison of the CPUs in the study performance of GoogLeNet and AlexNet with the Caffe-based app at every batch size. Figure 9 shows the same comparison as Figure 8; however, since the CPUs support hyper-threading, Caffe is using the maximum number of supported threads on the CPU. The thread count is 128, 272, and 288 for the Xeon Skylake, Xeon Phi KNL, and Xeon Phi KNM respectively.

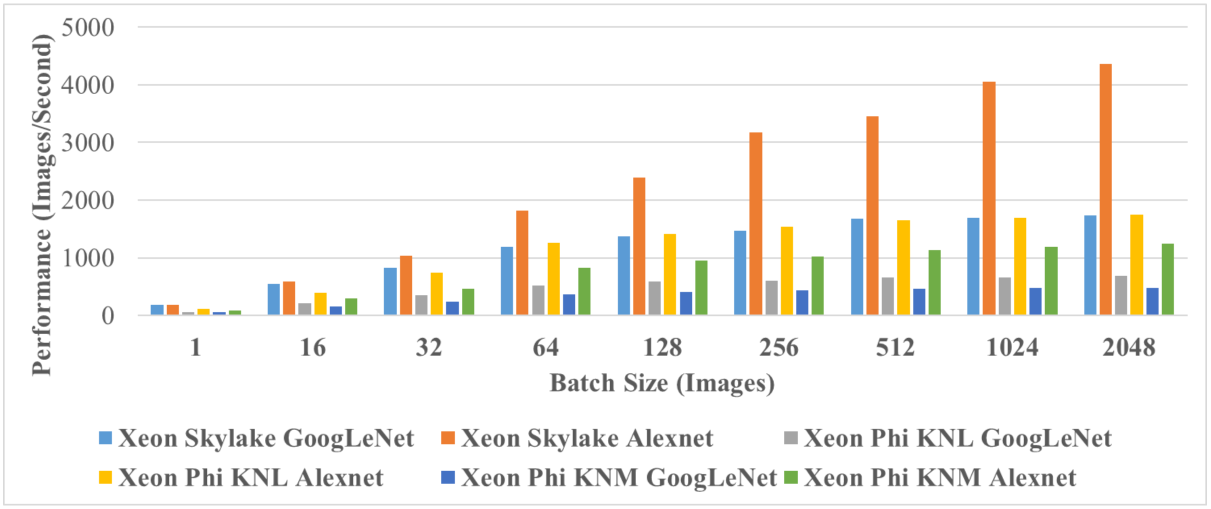

**Figure 8** CNN Performance Across CPU Devices with Varied Batch Sizes (See Appendix for Values)

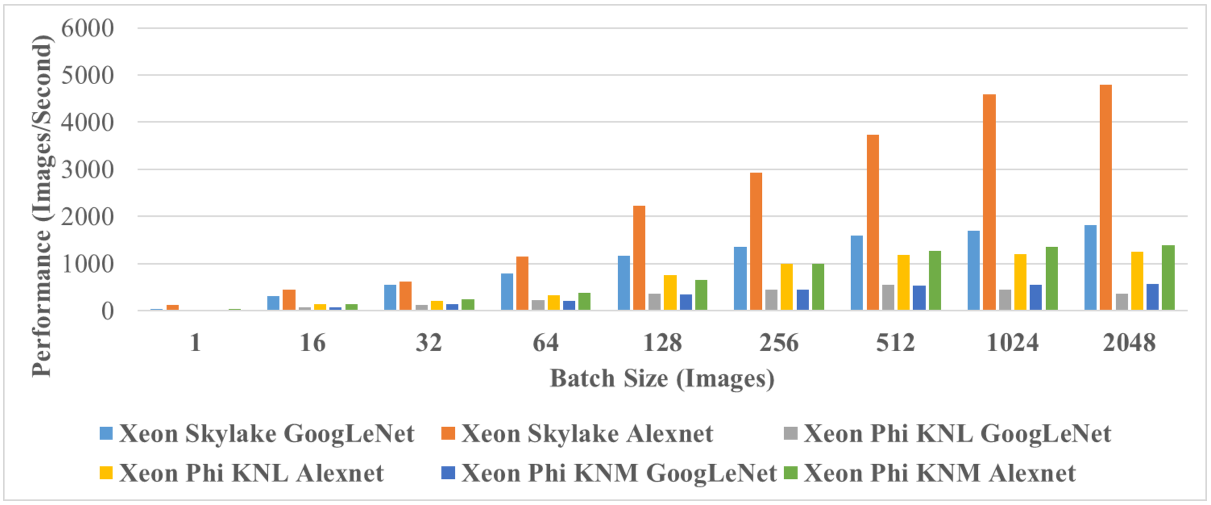

**Figure 9** CNN Performance Across CPU Devices using Hyper-threading with Varied Batch Sizes (See Appendix for Values)

The next graph, Figure 10, shows the performance of GoogLeNet and AlexNet on the Intel PAC FPGA with different OpenVINO binaries at every batch size up to 64 and including 96. Since the generic FP16 binary is limited to a batch size of 8, batch sizes 2, 4, and 8 are also included for comparison.

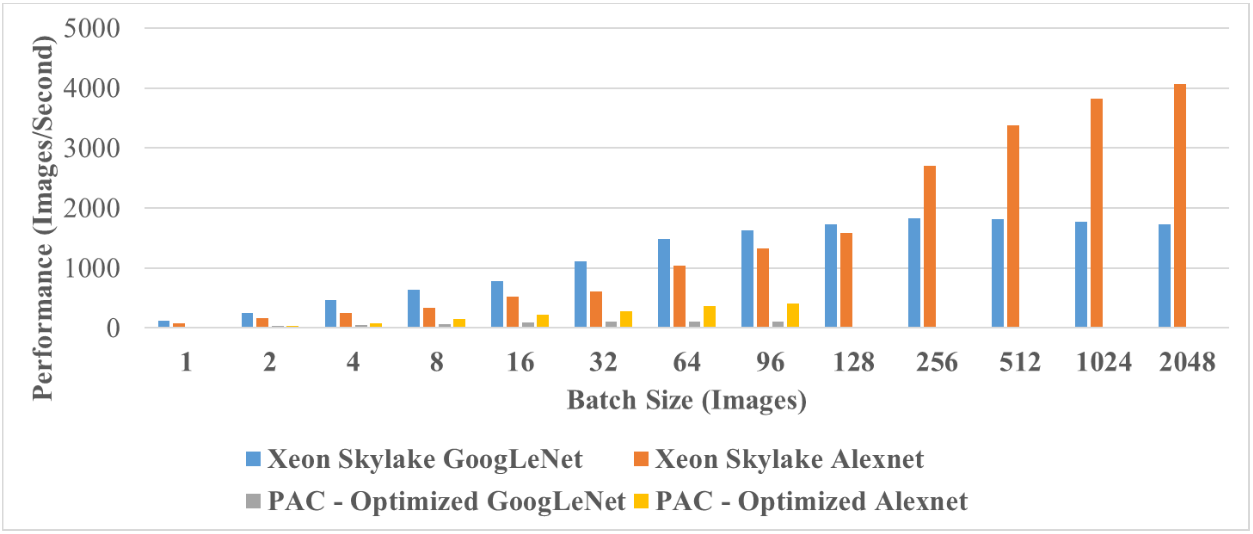

**Figure 10** CNN Performance Across Intel OpenVINO Binaries with Varied Batch Sizes

Figure 11 shows the comparison of the OpenVINO-based app on the Xeon Skylake and the Intel PAC FPGA across every batch size. Only the optimized versions of the OpenVINO FPGA binaries are included since the generic variant lacks in performance.

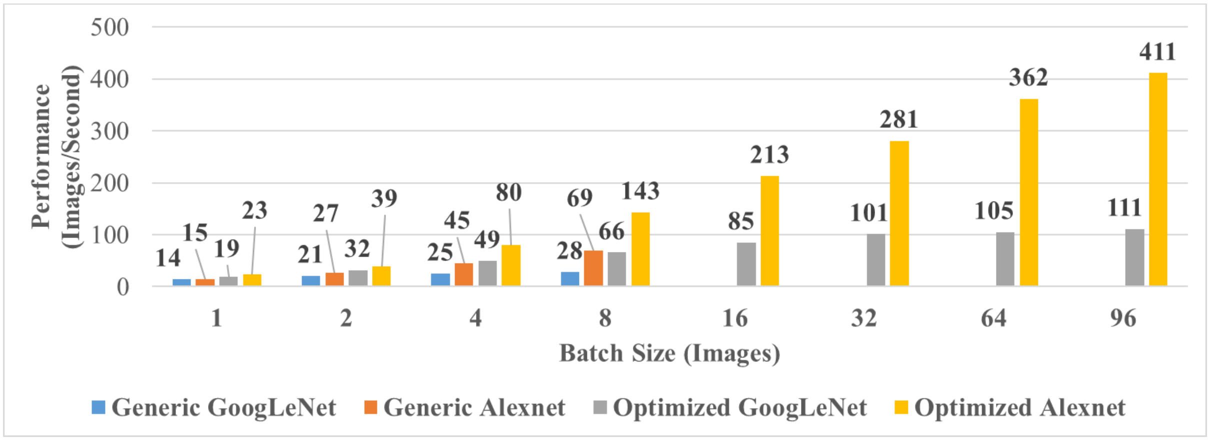

**Figure 11** CNN Performance Across Intel Devices using OpenVINO with Varied Batch Sizes (See Appendix for Values)

## 5.2 Performance Comparisons

Comparing the performance of the FPGA devices and frameworks, the maximum performance of xfDNN, Zebra, and OpenVINO are shown in Figure 12. Both GoogLeNet and Alexnet performance is shown with the frameworks' and networks' highest performing batch size.

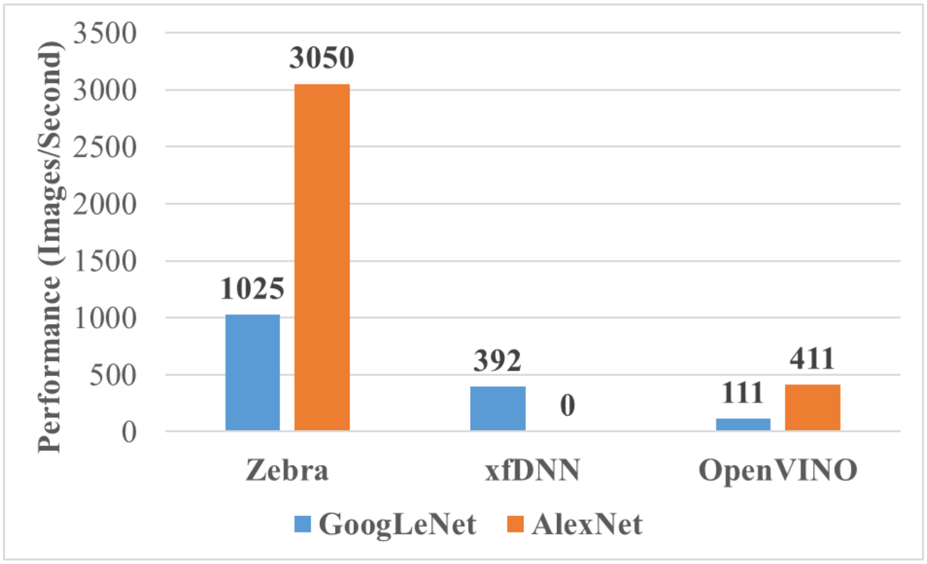

**Figure 12** Performance of Caffe-based Frameworks on Xilinx FPGA and OpenVINO on Intel PAC FPGA

The performance of both the Caffe-based and OpenVINO-based app on the Xeon Skylake CPU can be seen in Figure 13. This comparison shows the maximum performance of each app using hyper-threading or otherwise in the case of the Caffe-based app at the highest performing batch size.

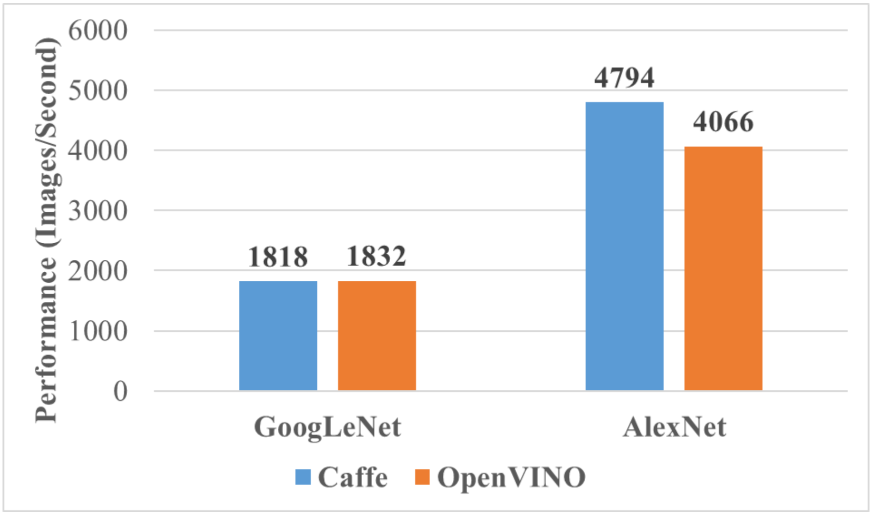

**Figure 13** CNN Performance on Xeon Skylake CPU using Caffe and OpenVINO

Since the Mipsology Zebra framework has higher performance than what is seen with both xfDNN and the PAC with OpenVINO, the performance of Zebra is compared to the GPUs using the Caffe-based app. Figure 14 shows the maximum performance of GoogLeNet and AlexNet on the FPGA using Zebra versus the maximum performance of GoogLeNet and AlexNet on the Tesla V100 since the V100 shows the highest performance out of the GPU group.

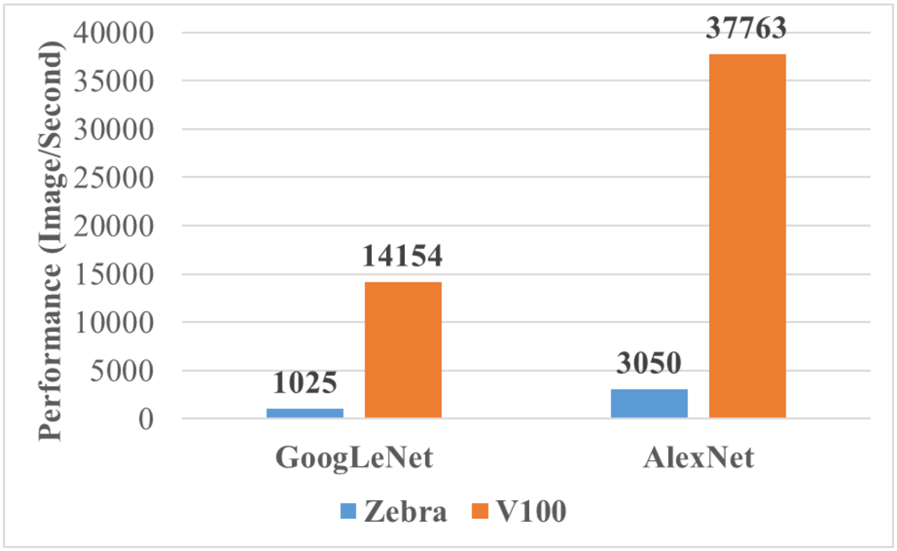

**Figure 14** CNN Performance on Xilinx FPGA Using Zebra and Tesla V100 GPU

Figure 15 shows the comparison of the maximum performance of the FPGA and CPU groups. The Xeon Skylake is used from the CPU group and again, Zebra is used for the FPGA group. Only the Caffe-based app results are included for the Skylake CPU since the OpenVINO-based app also shows similar performance.

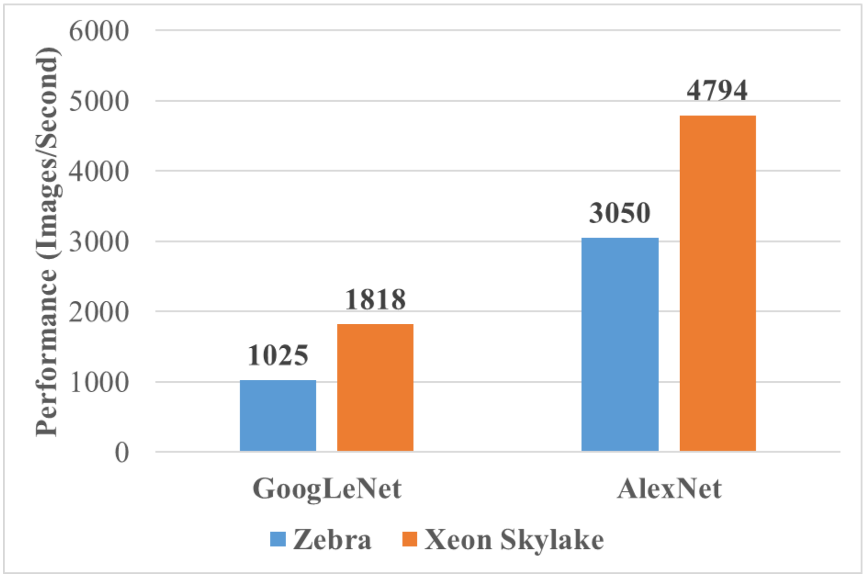

**Figure 15** CNN Performance on Xilinx FPGA Using Zebra and Xeon Skylake CPU

The next comparison shows the maximum performance of CPU and GPU group with the Caffe-based app in Figure 16. Again, the Xeon Skylake is used from the CPU group and the Tesla V100 is used from the GPU group each at their highest performing batch size.

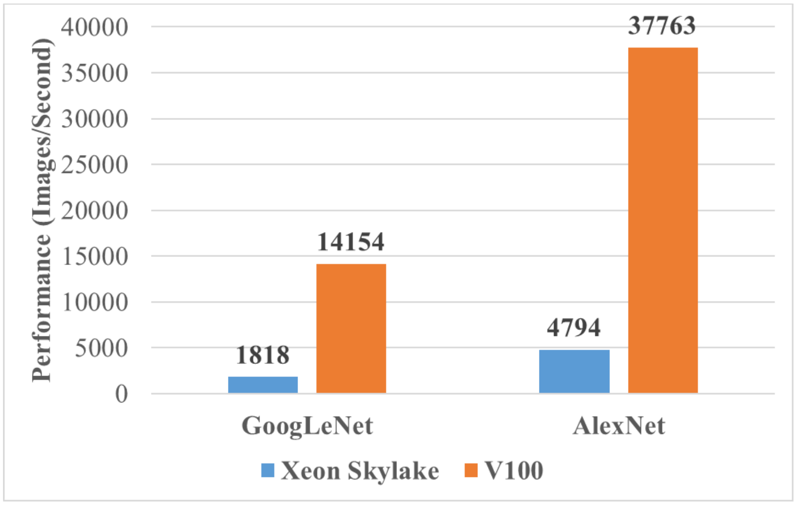

**Figure 16** CNN Performance on Xeon Skylake CPU and Tesla V100 GPU

## 5.3 Efficiency Comparisons

Figure 17 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 [44]. 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.

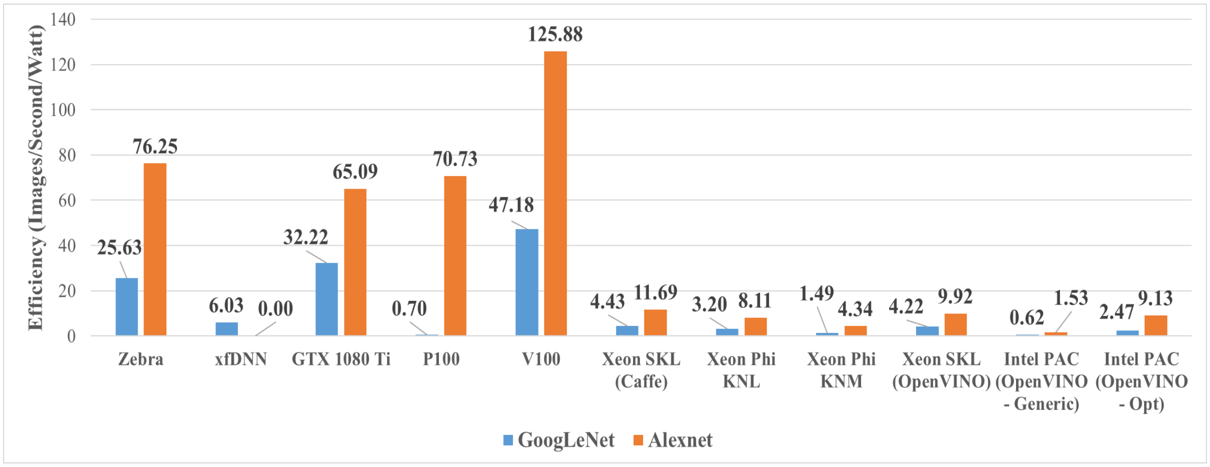

**Figure 17** Efficiency Characteristics of CNNs Across All Devices and Frameworks

Table 2 shows the cost per device in the study. This cost is used to evaluate the cost-efficiency and energy cost-efficiency of each device. The cost-efficiency of each device is in terms of performance-per-thousand dollars. The energy cost-efficiency of each device is in terms of performance-per-Watt-per-dollar. The cost-efficiency and energy cost-efficiency of each device can be seen in Figure 18 and Figure 19, respectively.

**Table 2** Current Cost of Devices in USD$

| Device | Cost (USD $) |
|---|---|
| Xilinx XCVU9P [45] | 13,687.75 |
| NVIDIA Tesla V100 [46] | 10,664.00 |
| NVIDIA Tesla P100 [46] | 9,428.00 |
| NVIDIA GTX 1080 Ti [39] | 699.00 |
| Intel Xeon Skylake 8180 [32] | 10,009.00 |
| Intel Xeon Phi KNL 7 [29] | 2,436.00 |
| Intel Xeon Phi KNM [47] | 4,876.00 |
| Intel PAC [48] | 8,740.00 |

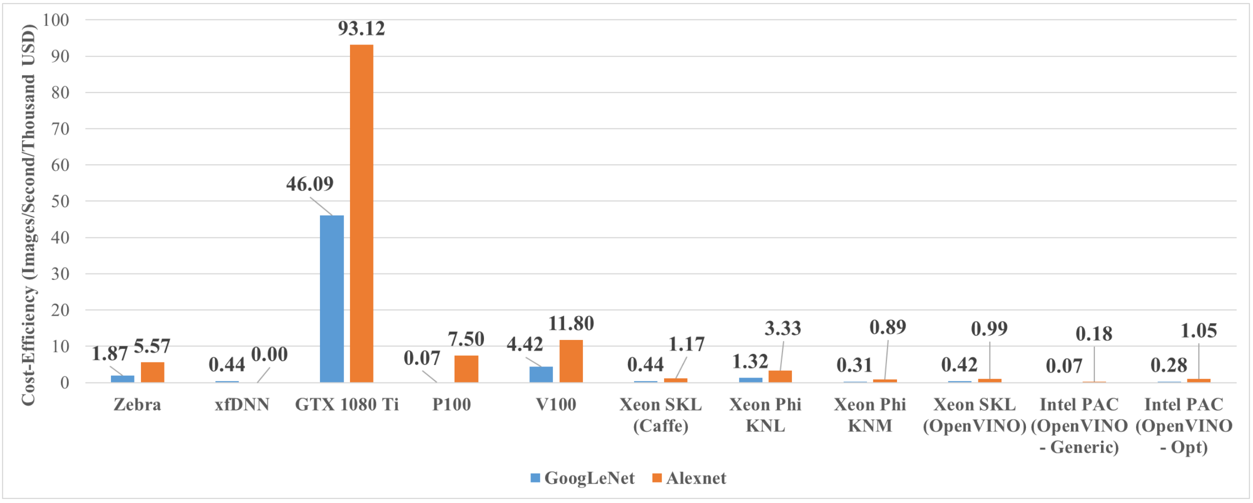

**Figure 18** Cost-Efficiency Characteristics of CNNs Across All Devices and Frameworks

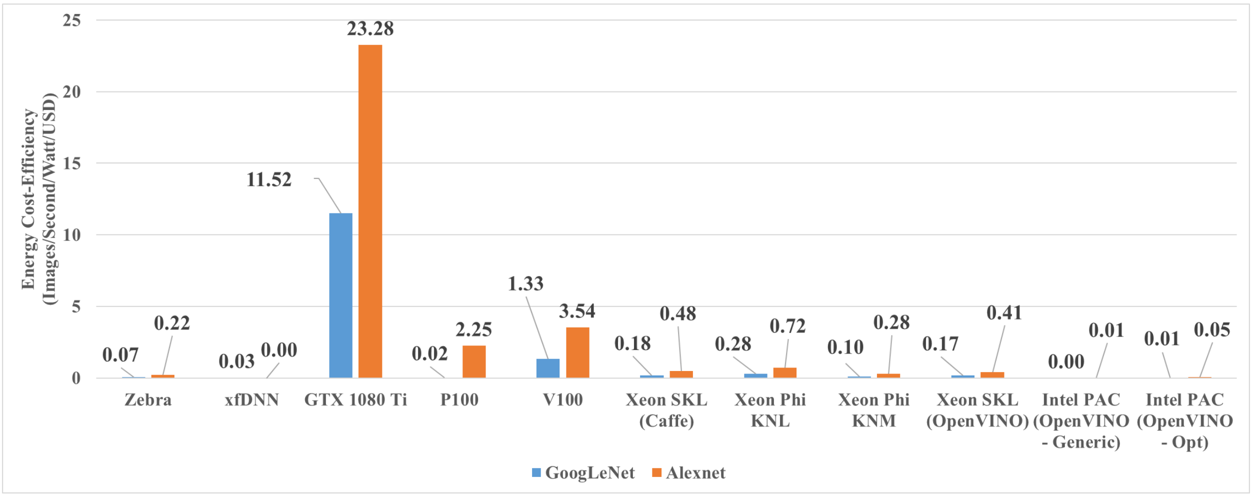

**Figure 19** Energy Cost-Efficiency Characteristics of CNNs Across All Devices and Frameworks

# 6.0 Discussion

When comparing the neural networks, GoogLeNet and AlexNet, we can see that AlexNet consistently achieves better performance than GoogLeNet. This higher performance of AlexNet is because of the shorter latency AlexNet has from input to output layers having only 5 layers compared to GoogLeNet's 14 layers, allowing for faster image classification. Since AlexNet has less layers than GoogLeNet, more on-board 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've observed the average Top-1 accuracies of AlexNet and GoogLeNet to be similar as 95.3% and 96.5%, respectively.

## 6.1 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.

Looking at the results from the GPUs there are two main takeaways. First, the Tesla P100 and GTX 1080 Ti both outperform the Tesla V100 with a one-image batch size. It is not until a batch size of 64 images is reached that we see the Tesla V100 start to outperform the other two GPU devices. Second, we can see that the performance of the Tesla V100 quickly grows with an increase in batch size, leveraging the parallelism of the Tensor cores. We can see that the parallel nature of the GPU helps performance at much larger batch sizes.

For the CPU results, we have interesting findings. First, 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 [32] [29] [30]. 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 [49]. 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 shows much better performance in terms of CNN training than testing. Next, we observed how hyper-threading affects app performance. In this case, again, the Xeon Skylake device significantly outperforms both Xeon Phi devices. Between the two Xeon Phi devices, KNM outperforms KNL in most batch sizes. KNM outperforms KNL when comparing the maximum performance of each device in regard to hyper-threading. In terms of maximum performance between single- and hyper-threaded apps, KNL still outperforms KNM. Another fact to note is that at a batch size of 512 images, AlexNet on the KNL device using hyper-threading peaks in performance but begins to degrade in performance at higher batch sizes. 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 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, we can see 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 8 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×, we can see 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. 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 8 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 we can expect the V100 to outperform the Xeon device, which is what we observe in the next comparison. The V100 outperforms the Xeon device by an average factor of 2.6×.

## 6.2 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 better 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 in being more efficient than general-purpose architectures that lack specific accelerators for this domain.

As we can see, 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.

In terms then of cost-efficiency, or how much it costs for the performance observed, we see some interesting results. First, by far, the GTX 1080 Ti has the best cost-efficiency. This result is due to the fact that the GTX 1080 Ti has similar performance to the Tesla P100; however, it is available for a fraction of the cost. It is the lowest cost device in the study and yet the third highest in terms of performance. All GPUs in the study take up the top three spots in terms of cost-efficiency, with the V100 being second place making it the most cost-efficient out of the server-grade accelerators. Regarding the rest of the devices, the Zebra framework in combination with the Xilinx FPGA has the next best performance, outperforming all of the Intel devices in cost-efficiency. Intel OpenVINO on the PAC FPGA and the Xeon Phi KNM prove to be the lowest cost-efficient devices in the study.

Another metric, energy cost-efficiency, is interesting to look at as well. This metric shows how cost-effective the device’s efficiency is. In this case, the GTX 1080 Ti still has the highest energy cost-efficiency of all of the devices, followed by the V100, and then closely by the P100 in third. Interestingly, The Intel Xeon and Xeon Phi device perform much better in this category especially the Xeon Phi KNL. However, they are still an order of magnitude below the V100 and

P100. Zebra on the Xilinx FPGA competes with the Intel CPUs, but still lags behind. Intel OpenVINO on the PAC FPGA proves to be the worst device in terms of energy cost-efficiency, significantly behind the other devices.

# 7.0 Conclusions

In this thesis, 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 [50].

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.

Surprisingly, the GTX 1080 Ti has the highest cost-efficiency and energy cost-efficiency being the only consumer-grade product in this study. The significantly lower price allows the GTX

1080 Ti to be 9.15x more cost-efficient and 7.6× more energy cost-efficient than the highest server-grade accelerator, the V100.

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, the Arria 10, is also the largest at 20-nm. This factor can have significant implications on performance of the devices.

Overall, GPUs dominate performance, efficiency, cost-efficiency, and efficiency cost-efficiency when accelerating CNNs with Caffe. The next most efficient devices, Xilinx FPGAs, need significant work 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 [21]. 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 2019, data from Xilinx shows Versal performing at 2× over the Tesla V100 using GoogLeNet for machine-learning inference with maximum batch size [50]. This architecture, in combination with the next release of xfDNN v3 and Zebra (December-2018), has potential to make FPGAs more competitive with GPUs for machine-learning inference and significantly more efficient.

This thesis has provided insight on 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.

# 8.0 Future Work

This research will continue to compare different architectures for accelerating machine-learning apps. As new architectures and frameworks emerge it is important to understand their relative performance, efficiency, and cost-efficiency in the machine-learning inferencing domain.

The authors of this research aim to include Google's TPUs in the study. As no support currently exists for Caffe on Google's TPUs, the Caffe models will need to be converted into models that are supported with TensorFlow, the only framework available on the TPUs [28]. As FPGA frameworks begin to provide support for TensorFlow and its respective models, previously studied architectures and frameworks will also be studied using the TensorFlow-based app as well.

Additionally, NVIDIA's Deep Learning Accelerator (NVDLA) will be studied. The NVDLA is an open-source accelerator targeted at Internet-of-Things (IoT) devices based on NVIDIA's Xavier architecture [51]. With portability in mind, NVDLA is designed to be used on many different FPGA accelerators.

From here, an OpenCL-based framework for accelerating CNNs in planned to be developed on both Xilinx and Intel FPGAs. This design is intended to start accelerating tensor operations, similar to the Volta architecture, and develop additional functionality. It is intended to be a scalable design to accelerate HPC machine-learning apps.

## Appendix

**Table 3** CNN Performance Across GPU Devices with Varied Batch Sizes (Associated with Figure 7)

| GPU Performance | GTX 1080 Ti | | P100 | | V100 | |
|---|---|---|---|---|---|---|
| | GoogLeNet | Alexnet | GoogLeNet | Alexnet | GoogLeNet | Alexnet |
| **1** | 365 | 641 | 211 | 490 | 181 | 474 |
| **16** | 3709 | 5533 | | 6119 | 2498 | 5965 |
| **32** | 5239 | 8473 | | 10655 | 4471 | 10362 |
| **64** | 6590 | 12854 | | 10199 | 7174 | 17272 |
| **128** | 7307 | 14014 | | 14609 | 10076 | 25324 |
| **256** | 7755 | 15184 | | 18028 | 12789 | 30482 |
| **512** | 8054 | 16306 | | 19810 | 14154 | 34980 |
| **1024** | | 16691 | | 21159 | | 37664 |
| **2048** | | 16273 | | 21218 | | 37763 |

**Table 4** CNN Performance Across CPU Devices with Varied Batch Sizes (Associated with Figure 8)

| CPU Performnace | Xeon Skylake | | Xeon Phi KNL | | Xeon Phi KNM | |
|---|---|---|---|---|---|---|
| | GoogLeNet | Alexnet | GoogLeNet | Alexnet | GoogLeNet | Alexnet |
| **1** | 182 | 184 | 58 | 117 | 55 | 92 |
| **16** | 553 | 588 | 211 | 398 | 151 | 294 |
| **32** | 826 | 1035 | 353 | 739 | 242 | 460 |
| **64** | 1186 | 1822 | 520 | 1264 | 360 | 821 |
| **128** | 1372 | 2388 | 587 | 1409 | 406 | 946 |
| **256** | 1474 | 3172 | 602 | 1534 | 434 | 1024 |
| **512** | 1681 | 3455 | 658 | 1651 | 463 | 1135 |
| **1024** | 1694 | 4050 | 664 | 1691 | 471 | 1187 |
| **2048** | 1730 | 4358 | 689 | 1744 | 476 | 1246 |

**Table 5** CNN Performance Across CPU Devices with Varied Batch Sizes using Hyper-threading (Associated with Figure 9)

| CPU Performnace – Hyper-threading | Xeon Skylake | | Xeon Phi KNL | | Xeon Phi KNM | |
|---|---|---|---|---|---|---|
| | GoogLeNet | Alexnet | GoogLeNet | Alexnet | GoogLeNet | Alexnet |
| **1** | 28 | 130 | 14 | 26 | 13 | 29 |
| **16** | 309 | 445 | 66 | 131 | 76 | 143 |
| **32** | 542 | 624 | 128 | 206 | 136 | 238 |
| **64** | 783 | 1140 | 223 | 329 | 210 | 381 |
| **128** | 1158 | 2220 | 362 | 758 | 347 | 645 |
| **256** | 1362 | 2933 | 454 | 1002 | 454 | 1001 |
| **512** | 1598 | 3736 | 542 | 1187 | 534 | 1267 |
| **1024** | 1697 | 4582 | 444 | 1193 | 542 | 1348 |
| **2048** | 1818 | 4794 | 368 | 1248 | 574 | 1388 |

**Table 6** CNN Performance Across Intel Devices using OpenVINO with Varied Batch Sizes (Associated with Figure 11)

| OpenVINO PAC FPGA - CPU | Intel Xeon Skylake CPU | | Intel PAC FPGA | |
|---|---|---|---|---|
| | GoogLeNet | Alexnet | GoogLeNet | Alexnet |
| **1** | 119 | 76 | 19 | 23 |
| **2** | 243 | 159 | 32 | 39 |
| **4** | 460 | 249 | 49 | 80 |
| **8** | 639 | 336 | 66 | 143 |
| **16** | 778 | 523 | 85 | 213 |
| **32** | 1115 | 606 | 101 | 281 |
| **64** | 1486 | 1033 | 105 | 362 |
| **96** | 1632 | 1324 | 111 | 411 |
| **128** | 1723 | 1580 | | |
| **256** | 1832 | 2695 | | |
| **512** | 1813 | 3382 | | |
| **1024** | 1768 | 3827 | | |
| **2048** | 1730 | 4066 | | |

# Bibliography

[1] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2009.

[2] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in IEEE Computer Vision and Pattern Recognition (CVPR), 2016.

[3] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution), "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211-252, December 2015.

[4] Joseph Redmon, Ali Farhadi, "YOLO9000: Bigger, Faster, Stronger," arXiv, 25 December 2016.

[5] 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.

[6] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," in ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2015.

[7] 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].

[8] 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.

[9] 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.

[10] P. J. Werbos, "Backpropagation through time: what it does and how to do it," in IEEE, 1990.

[11]] Zhuoyao Zhong, Lianwen Jin, Zecheng 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.

[12] 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].

[13] 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.

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

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

[16] Xavier Glorot, Antoine Bordes, Yoshua Bengio, "Deep Sparse Rectifier Neural Networks," in Fourteenth International Conference on Artificial Intelligence and Statistics (PMLR), 2011.

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

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

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

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

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

[22] 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].

[23] U. o. Pittsburgh, "Center for Research Computing Resources," November 2018. [Online]. Available: https://crc.pitt.edu/resources. [Accessed November 2018].

[24] Elliot Delaye, "Integrating AI into Your Accelerated Cloud Applications," 2018.

[25] 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].

[26] NVIDIA, "NIVIDIA cuDNN," November 2018. [Online]. Available: https://developer.nvidia.com/cudnn. [Accessed November 2018].

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

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

[29] 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].

[30] 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].

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

[32] 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].

[33] Intel, "FPGA Acceleration Stack - OPAE," November 2018. [Online]. Available: https://01.org/sites/default/files/downloads/opae/open-programmable-acceleration-engine-paper.pdf. [Accessed November 2018].

[34] 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].

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

[36] 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.

[37] Songxuan Lai, Lianwen Jin, Weixin 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.

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

[39] 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].

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

[41] 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].

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

[43] John L. Hennessy, David A. Patterson, Computer Architecture: A Quantitative Approach, 6 ed., San Francisco, CA: Morgan Kaufmann, 2017.

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

[45] BittWare, "Xilinx FPGA Boards," 7 March 2018. [Online]. Available: https://www.bittware.com/fpga/xilinx/boards/. [Accessed November 2018].

[46] Brett Newman, "NVIDIA Tesla V100 Price Analysis," Microway, 8 March 2018. [Online]. Available: https://www.microway.com/hpc-tech-tips/nvidia-tesla-v100-price-analysis/. [Accessed November 2018].

[47] CPU-WORLD, "Intel Xeon Phi 7250," 20 June 2016. [Online]. Available: http://www.cpu-world.com/CPUs/Xeon_Phi/Intel-Xeon%20Phi%207250.html. [Accessed 2018 November].

[48] Dell, "Intel Programmable Acceleration Card," November 2018. [Online]. Available: https://www.dell.com/en-us/shop/intel-fpga-programmable-acceleration-card-70w-full-height/apd/403-bbvz/storage-drives-media. [Accessed November 2018].

[49] 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].

[50] 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].

[51] NVIDIA, "NVIDIA Deep Learning Accelerator (NVDLA)," 2018. [Online]. Available: http://nvdla.org/. [Accessed November 2018].