




A systematic literature review on hardware implementation of artificial intelligence algorithms

Manar Abu Talib¹ · Sohaib Majzoub¹ · Qassim Nasir¹  · Dina Jamal¹

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Artificial intelligence (AI) and machine learning (ML) tools play a significant role in the recent evolution of smart systems. AI solutions are pushing towards a significant shift in many fields such as healthcare, autonomous airplanes and vehicles, security, marketing customer profiling and other diverse areas. One of the main challenges hindering the AI potential is the demand for high-performance computation resources. Recently, hardware accelerators are developed in order to provide the needed computational power for the AI and ML tools. In the literature, hardware accelerators are built using FPGAs, GPUs and ASICs to accelerate computationally intensive tasks. These accelerators provide high-performance hardware while preserving the required accuracy. In this work, we present a systematic literature review that focuses on exploring the available hardware accelerators for the AI and ML tools. More than 169 different research papers published between the years 2009 and 2019 are studied and analysed.

Keywords Hardware accelerators · Artificial intelligence · Machine learning · AI on hardware · Real-time AI

1 Introduction

Artificial intelligence (AI) and machine learning (ML) tools gained a significant popularity in the last decade due to the advances in computational systems in terms of power, area, and performance. A wide range of applications started utilizing AI algorithms to obtain superior results compared to traditional methods. Such applications include image processing [1] such as face detection and recognition, banking and market analysis [2], robotic arms in the automated manufacturing industry [3], healthcare applications [4], efficient and smart transactions in database management [5], and security applications for face tracking and analysis [6]. Embedded vision

✉ Qassim Nasir
nasir@sharjah.ac.ae

¹ University of Sharjah, Sharjah, United Arab Emirates

systems are becoming a part of an emerging number of applications such as autonomous or semi-autonomous vehicles and planes, security and healthcare applications [7].

AI is revolutionizing areas such as autonomous vehicles where it uses deep learning algorithms in an attempt to reform personal transport [8]. AI is also used in autonomous drones for navigation and as a fighter-jet in military applications [9, 10]. Many of these applications utilize AI at its core, where efficiency and accuracy can make the difference between life and death. This trend shows the continuous interest and the high potential of AI and ML tools. These tools are increasingly becoming an inherent part of every electronic or embedded system. One of the main challenges facing this new shift is that AI algorithms are computationally expensive. This fact is pushing towards improving hardware acceleration in order to offer the required high computational power.

An optimized and specialized hardware implementation can reduce system cost by optimizing the needed resources and decreasing power requirements while improving the performance [11]. Typically, AI algorithms are developed and tested using development platforms such as Tensorflow, Keras and Caffe. Some of these platforms are popular standard neural networks (NN) such as AlexNet and VGG tools, from Visual Geometry Group. Developed NNs for specific group of applications are then realized in hardware and referred to as hardware accelerators. Hardware implementations are considered on FPGAs, GPUs, and ASICs, each of which has their own pros and cons. These hardware accelerators exploit parallelism to increase throughput and provide much higher performance compared to traditional CPUs, which can be 1/10 1/100 times slower [12].

AI algorithms, especially deep learning (DL) algorithms, require significant storage to use large databases and powerful computational processing to produce autonomous adaptive smart system for highly accurate and human-like behaviour [13]. Note that the field of deep learning (DL) only emerged in 2006 [14]. Its particular features gave it the edge to supersede traditional methods used by available systems or even humans, particularly in gaming where it won against the most proficient players in the board games of chess GO [15]. Such distinctive features provided high accuracy by eliminating the factor of human error.

In this work, we present a comprehensive survey providing a systematic literature review (SLR) to assist researchers in the area of AI hardware acceleration. The survey covers hardware implementation research of AI and ML algorithms from 2009 up until 2019. The focus of this work is the implementation of NNs as a tool for object detection in different applications. We have collected a total of 201 different research papers, out of which 169 papers addressed the hardware implementation of AI and ML algorithms. We adopted narrow exclusion criteria for this work to consider as many papers as possible in order to cover a wide perspective on the issue at hand.

The rest of the paper is organized as follows: literature review covering related work is discussed in Sect. 2. Then, Sect. 3 covers the methodology part explaining the used criteria to categorize and assess the quality of the collected research papers. In Sect. 4, we discuss the analysis and answers to the research questions set in the methodology section. Finally, the conclusion of this work is presented in Sect. 6.

2 Literature review

The research on hardware implementation of AI algorithms is gaining interest in recent years. It has been the focus of many survey papers. This interest is mainly to address computationally expensive, power-hungry AI algorithms. It is due to the need for specialized hardware with enough computing power to perform AI algorithms with large scale problems [16]. Hardware acceleration is considered the ideal solution for these algorithms [16]. The goal is to achieve faster and more efficient processing of AI algorithms. As a result, a numerous number of hardware implementations for different applications are proposed. Many studies are published over the last decade discussing hardware and software optimizations and implementations methods in this field [1, 11, 12, 14, 17, 17–19].

In this section, we explore related survey articles published between the year of 2009 and 2019 discussing hardware implementation of AI algorithms. Some surveys covered artificial neural networks implementations on hardware in general [11, 17, 18]. Other surveys focused on FPGA-based accelerators for deep learning neural network [1, 12, 14]. In [17], the authors focused on the FPGA implementation of convolutional neural networks (CNNs). In [19], the survey discussed GPU implementations. To the best of our knowledge, our SLR is the most recent survey with comprehensive coverage discussing the recent work in the area of AI hardware implementation on FPGAs, GPUs and ASIC.

Misra et al. [11] surveyed all major hardware realizations of artificial neural networks (ANNs) implementations on hardware, referred to as hardware neural networks (HNNs). HNNs have been classified according to some attributes such as on-chip/off-chip, analog/digital blocks, threshold/look-up table/computation and clock and data transfer rates [11]. A dedicated section for HNN chips covered FPGA-based and ASIC digital and hybrid neurochip implementations. Guo et al. [18] a more specialized survey concentrating on FPGA-based neural network accelerators, where they mentioned NN accelerator designs and summarized all methods used for accelerator design automation. Shawahna et al. [14] discussed the basics of deep neural network learning, with a specified section for CNN compression implementation techniques. The survey analyses FPGA-based accelerators and ASIC-based accelerators, with more focus on FPGAs. Other surveys such as [1, 17] also reviewed techniques for designing FPGA-based accelerators for CNNs. In the studied surveys, different GPU alternatives were not studied comprehensively for performance improvement of AI algorithms. We discuss the existing literature on the use of GPUs in this survey, because it is a viable option that can compete with FPGA and ASIC-based solutions. The paper provides a much needed comparative study that analyses the existing work that has been done in the last decade from FPGA, GPU, and ASIC perspectives.

This work complements the existing work and contributes towards providing the complete background on AI accelerators. The contributions of this survey can be summarized as follows:

1. The work follows the systematic literature review (SLR) approach to provide the most relevant information in this subject.
2. The survey covers the work done in the last decade, 2009 up to 2019.
3. The survey provides a comparative study of existing hardware options: FPGAs, GPUs, and ASICs.
4. The work spans papers that are at the software level, covering the framework and the algorithms used, as well as the hardware level, covering the accelerators' implementation and architecture design.

In total, eight different surveys with similar goal were done in the last decade. Table 1 presents a summary of these surveys, showing the survey's date, title, and a brief description of the survey scope.

3 Methodology

The systematic literature review (SLR) covers a wide range of hardware implementation of mainstream AI and ML algorithms. The used methodology is based on Kitchenham and Charter's methodology [20], shown in Fig. 1.

The objective of this survey is set to answer multiple research questions concerning artificial intelligence hardware accelerators, these research questions are:

- RQ1: Application perspective: What are the main applications that use AI and ML tools in hardware implementation? This question is to learn about the applications utilizing AI accelerators.
- RQ2: AI and ML perspective: What are the AI and ML algorithms and tools used for hardware implementation? This question is to analyse the most commonly used algorithms and frameworks in hardware accelerators.
- RQ3: Hardware perspective: What are the trade-offs when using different hardware platforms for AI acceleration? This question is to study and compare available hardware implementation options based on standard figures of merit.

3.1 Search strategy

The second stage of conducting this review is to identify the keywords used in searching and collecting relevant papers in order to answer the research questions. The following keywords are used:

("ASIC" OR "FPGA" OR "GPU" OR "Artificial Intelligence" OR "Neural Network") AND ("Accelerator")

The following digital libraries are used in order to get all the related articles (journals as well as conference papers) by using the search terms mentioned above:

- IEEE Explorer
- Google Scholar

Table 1 A list of all survey papers found and their corresponding overall summaries

Publication year	Survey title	Description and scope
2010	Artificial neural networks in hardware: a survey of two decades of progress	This survey paper discusses all major hardware neural network (HNN) design approaches and models throughout the years 1990–2010. They provide examples of various HNN implementations across wide range of ANN models such as CNN, spiking neural network, etc. These HNN implementations are digital, analog, hybrid, neuromorphic, FPGA or optical. Each one of them is discussed separately in different sections [11]
2013	A survey of software and hardware use in artificial neural networks	The scope of the survey was whether developers use ready-made open source software frameworks and hardware accelerators or develop their own custom versions. A table of software frameworks used for artificial neural networks (ANN) is also provided [17]
2017	A survey of FPGA-based neural network accelerator	This survey paper gives an overview of previous work on neural network inference accelerators based on FPGA and summarize the main techniques used [18]
2017	Efficient processing of deep neural networks: a tutorial and survey	This paper focuses only on the deep neural networks (DNN) type of ANN. It mentions the history, components, and application with the hardware that supports DNN type operations [1]
2018	A survey of FPGA-based accelerators for convolutional neural networks	The paper provides a survey of techniques for designing FPGA-based accelerators for CNNs. They discuss several optimization strategies used in hardware architectures for CNNs such as Loop reordering, unrolling and pipelining, fixed-point Format, etc. They also classify all implementations according to the optimization approaches for CNN and FPGA architecture or memory related optimizations [17]
2018	Evolution of the GPU device widely used in AI and massive parallel processing	The paper presents a short discussion about the history of CPUs and GPU, while going in depth with Moore's law. It also gives several important applications for AI [12]
2018	A survey of FPGA based deep learning accelerators: challenges and opportunities	The paper provides a survey about designs of FPGA-based deep learning accelerators. They classify the work according to the design, either for a specific algorithm or for a specific application or a universal accelerator framework with hardware templates [19]

Table 1 (continued)

Publication year	Survey title	Description and scope
2019	FPGA-based accelerators of deep learning networks for learning and classification: a review	The paper gives comprehensive background and history about FPGA-based accelerators. It includes the applications of deep learning and discusses GPU, ASIC (with specific examples) and FPGAs implementation examples in detailed analysis and testing [14]



Fig. 1 Systematic literature review (SLR) methodology

- ACM Digital Library
- Springer
- Elsevier

3.2 Study selection

Initially, a total of 188 papers were collected based on the search terms mentioned earlier. Then, the papers were filtered down to more focused categories. The selection and filtration process are explained below:

- Step 1: Remove duplicated or multiple versions of the same articles.
- Step 2: Apply inclusion and exclusion criteria to avoid any irrelevant papers.
- Step 3: Remove review papers from the collected papers.
- Step 4: Apply quality assessment rules to include qualified papers that best address the research questions.
- Step 5: Look for additional related papers using the reference lists of the collected papers and repeat the same process again. The applied inclusion and exclusion criteria are defined as the following:

1. *Inclusion criteria*

- Date from 2009 to 2019.
- Only journals and conference papers that discuss the optimization or implementation of AI algorithms in hardware.

2. *Exclusion criteria*

- Non-journal and non-conference articles.
- Hardware implementations that do not incorporate the use of AI or ML tools.
- Articles that discuss AI algorithms without hardware implementation.

3.3 Quality assessment rules (QARs)

Applying the quality assessment rules (QARs) is the final step used to identify the list of papers that are considered in this review. The QARs are important to ensure the proper evaluation of the research papers' quality. Therefore, 10 QARs are specified, each worth 1 mark out of 10. The score of each QAR is selected as follows: "fully answered" = 1, "above average" = 0.75, "average" = 0.5, "below average" = 0.25, "not answered" = 0. The summation of the marks obtained for the 10 QARs is the score of each article. Moreover, if the result is 5 or higher, the article is considered otherwise it is excluded. The QARs are listed below:

- QAR1: Are the research objectives identified clearly?
- QAR2: Are the techniques of the AI implementation well defined and deliberated?
- QAR3: Is a specific application of AI algorithm or accelerator design clearly defined?
- QAR4: Does the paper include practical experiments of the proposed technique?
- QAR5: Are the experiments well designed and justified?
- QAR6: Is the proposed algorithm or accelerator design fully tested using standard test cases?
- QAR7: Is the accelerator's accuracy reported?
- QAR8: Is the proposed accelerator design compared to others?
- QAR9: Are the methods used to analyse the results appropriate?
- QAR10: Does the overall study contribute significantly to the AI accelerators area of research?

3.4 Data extraction strategy

In this step, the objective is to analyse the final list of papers in order to extract the needed information that answers the given research questions. First, general details are extracted from each paper such as the paper number, paper title, publication year, and publication type. Then, more specific information is sought, such as the algorithm models and the category of hardware accelerators, whether FPGA, GPU, ASIC or a combination of them. Finally, any details directly related to the research questions are pursued. The extraction of such details is challenging due to the basic unstructured nature of the needed information. For instance, some papers discussed the compilation of software into the hardware descriptive language to be implemented on FPGAs [21]. Note that not all papers answered the three research questions.

3.5 Synthesis of extracted data

In order to synthesize the information obtained from the selected papers, we use various methods to aggregate and formulate the needed answers of the research

questions. A qualitative synthesis is used to obtain the needed information for RQ1 and RQ2. On the other hand, quantitative synthesis is used for RQ3 to gauge and compare the hardware implementations from different papers. The comparisons are shown using standard figures of merit calculations presented in a comparative tabulated format.

4 Results and discussions

In this section, findings of the predefined research questions are discussed. Based on the selected papers, we present a survey conducted on hardware implementations of AI and ML algorithms between 2009 and 2019. The first subsection gives an overview of the collected papers, followed by subsections that discuss each research question individually identifying important conclusions and remarks.

4.1 Study overview

As mentioned earlier, a total of 169, out of 201 surveyed, research papers are identified and discussed in this review. These papers have been categorized according to the hardware used for implementation e.g. (FPGA, ASIC, GPU.. etc). Figure 2 shows the overall distribution of the papers according to this categorization.

As shown in Fig. 2, the majority of the papers are FPGA-based implementations. Due to its rapid prototyping environment, significant number of research papers used FPGAs to implement complex AI algorithms. FPGA's ability to provide an easy prototyping environment with powerful computational resources made it very useful for researchers to design and test new accelerators for AI

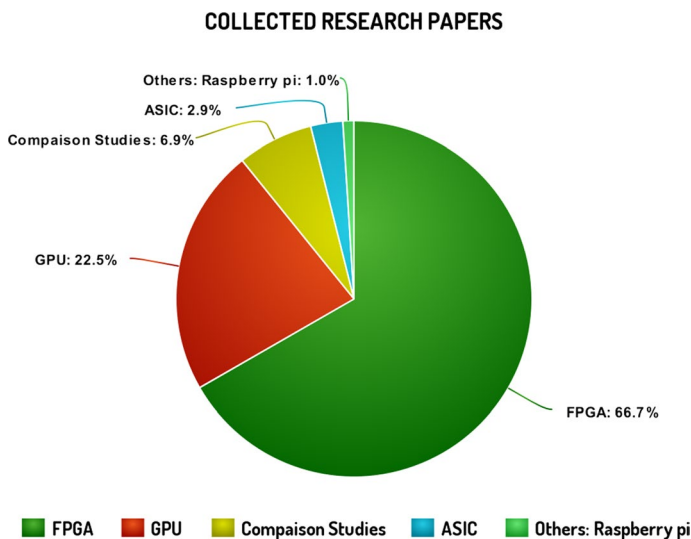


Fig. 2 Hardware implementation platform distribution in the collected research papers

algorithms [14]. As a result, the implementations of algorithms such as deep neural networks with irregular parallelism and custom data types are evolving rapidly [22]. FPGA with its high flexibility is considered to be a viable solution to implement these algorithms. Moreover, FPGA recent design trends resulted in making it more accessible and gained significant attention for deep learning research [23].

Application-specific integrated circuit (ASICs) is also among the powerful platforms used to implement AI algorithms. ASICs are customized chips designed for a specific purpose. They can save silicon area with high power and speed efficiency, which makes them suitable solutions for AI algorithms acceleration [24]. Graphical processing units (GPUs) are also used to accelerate AI algorithms where it speeds up algorithms from several times to hundreds of times [25]. GPUs are designed to perform intensive scalar and parallel computing [26]. Unlike multicore CPUs, GPUs do not rely on hidden latencies using big cache memories when accessing DRAM memories [27]. This results in aggressive explicit parallelism emphasizing throughput optimization much more than the CPU cores [27]. These characteristics made GPUs increasingly useful for AI acceleration.

Some AI algorithms are implemented using small single-board computers such as raspberry pi [28]. Single-board computers are considered a choice for AI implementations due to its small size and low power requirements. The remaining group of papers are related to comparisons between AI algorithms when implemented in various hardware accelerators. Collected research papers are analysed according to the publication year to evaluate the interest in AI accelerators for the last decade as shown in Fig. 3.

Figure 3 shows that the significant increase in AI accelerators publications since 2009 with a noticeable increase above the average is after 2015. The massive advancement in digital electronics resulted in immense computing power platforms. Such platforms provided the needed computational power for AI and its applications. AI applicability for wide range of areas is the reason behind the recent revival in the AI and ML fields [14].

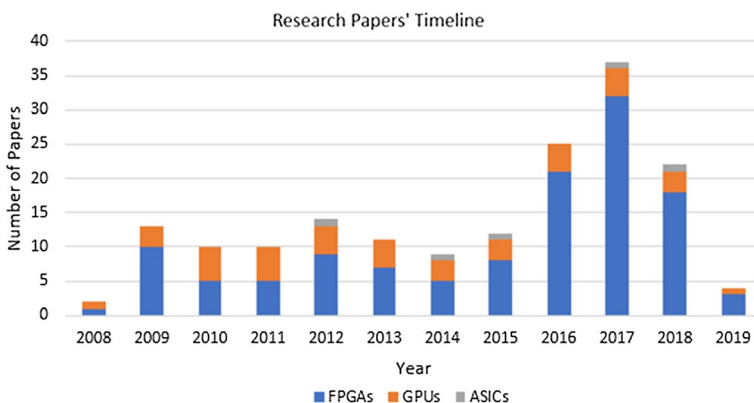


Fig. 3 Collected research papers between 2008 and 2019

4.2 Application perspective

The collected research papers are categorized according to the implemented application. Six categories are identified based on the implemented applications (Fig. 4). The list of these categories is shown below:

- Image processing
- Data mining
- Ambient intelligence
- Robotics
- Resources Gauge model
- Global navigation

As shown in Fig. 5, 93% of the papers are image processing related applications. Image processing is a rapidly growing area. A wide range of emerging applications employs image processing technique such as computer graphics, biomedical imaging, security and recognition imaging, satellite imaging, and underwater image restoration and enhancement. With the use of AI in image processing field, complex tasks such as image enhancement, object detection, face recognition and text recognition showed considerable improvements [25–27, 29].

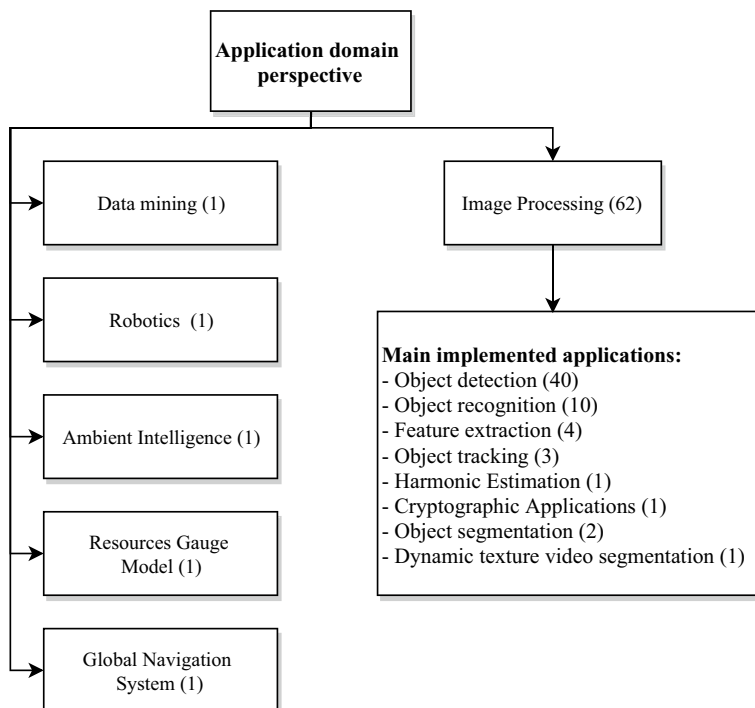


Fig. 4 Summary of application perspective section's outcomes * () indicates number of papers per application

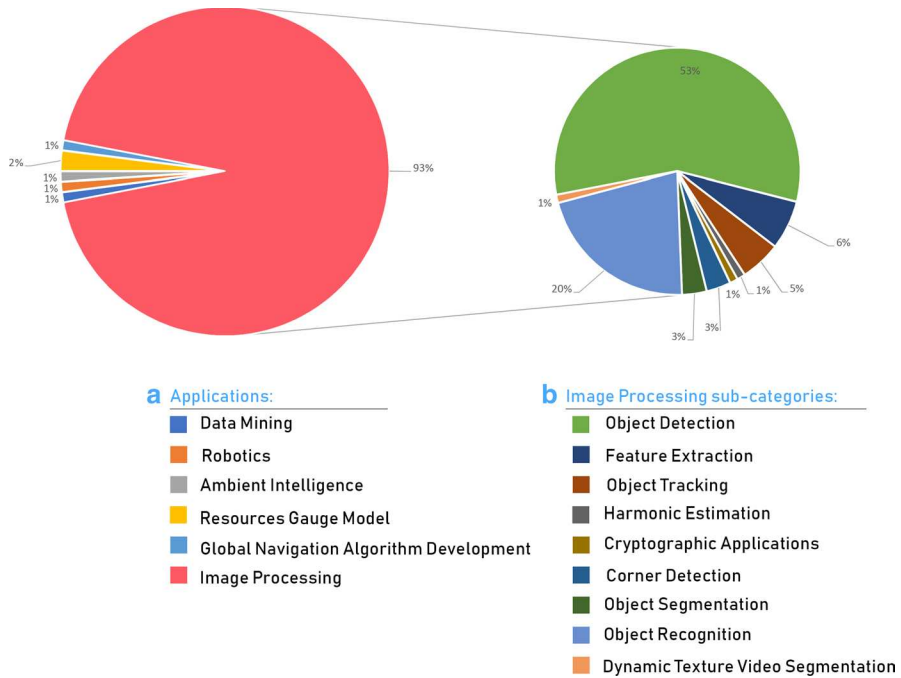


Fig. 5 **a** A comprehensive distribution of all the applications. **b** Image processing sub-category amongst all collected papers. Only 75 papers addressed application mapping

The majority of the research papers that are related to image processing are object detection. Increased interest in object detection has been due to its close relationship to computer vision and image analysis. Object detection is used in a wider range of applications especially in two main applications as shown in Table 2 which are face detection and recognition [27]. As shown in Fig. 5, most object detection's papers are about face detection, which indicates that facial detection using AI has been of high interest for researches. This is because face detection is the key component in many face-related security technologies such as surveillance and tracking of cars in traffic and individuals for criminal and security applications [27, 28, 30].

Table 2 A list of main applications of object detection using hardware accelerators

Application type	Usage description	References	Percentage
Face detection	Face detection is used in many applications such as monitoring and surveillance, human computer interfaces, smart locks, intelligent robots, and biomedical image analysis [30]	[7, 30–37]	23%
Pedestrian detection	Pedestrian detection is considered to be a key task for advanced driver assistance systems [38]	[38–41]	5%

Other image processing applications such as object tracking, and object recognition are also considered for hardware implementation in recent years. These applications gained popularity due to its future potential to automate many decision-making systems in real time.

4.3 Algorithms, tools, and platforms perspective

In this section, an analysis for the collected research papers from algorithm perspective is presented in order to state the most used AI and ML algorithms in different applications (Fig. 6). Most of the research papers implemented object detection acceleration. Table 3 shows all algorithms and tools used in object detection papers.

As shown in Table 3, researches implemented Viola–Jones framework for object detection and specially for face detection. For feature descriptors phase in object detection, researchers tend to implement histogram of oriented gradients (HOG), local binary pattern (LBP), speeded up robust features (SURF), in addition to two different extensions from HOG algorithm which are co-occurrence HOG and Binary patterned HOG features. HOG descriptors are also commonly used in object detection implementations especially pedestrian detection [49]. HOG algorithm achieves high detection accuracy and its hardware accelerators such as FPGAs are limited due to intensive floating-point computations [50]. This limitation is overcome in [50] using reduced bit-width fixed-point representation. Their solution resulted in achieving a 68.7× higher throughput than a high-end CPU, 5.1× higher than a high-end GPU, with 130× less power than CPU. Other algorithm that is also frequently used in object detection is local binary pattern (LBP). LBP is an algorithm which represents face image features by vectors that are generated from concatenation of the histograms of small regions in the image [51].

From deep learning field, convolutional neural networks (CNNs) are commonly used for object detection. CNNs are feed-forward architecture that consists of multiple linear convolution filters, interspersed with point-wise nonlinear squashing functions [7]. The CNN's popularity is due to its ability to train easily for variety of tasks such as OCR, object detection, object recognition and robot navigation [34]. Unsupervised learning algorithm such as non parametric background modelling has also been used for object detection.

In terms of classifiers, as shown in Table 3, Haar classifier is a widely used classifier. Haar classifier is considered as the core algorithm of object detection [61]. It is a viable part of the AdaBoost algorithm [31]. AdaBoost is a supervised machine learning algorithm that is used to boost classification performance of simple learning algorithms [27]. In the case of the Haar classifier, AdaBoost is used to both selecting the features and training the classifier [27]. AdaBoost algorithm uses Haar features to build weak classifier. Weak classifiers are combined to create a strong classifier that can detect more complicated objects [27]. Out of the seven research papers that uses Haar classifier algorithm, two of them are applied on FPGAs, whereas the rest used GPU for algorithm acceleration.

Table 4 presents a comparison between the two FPGA Haar classifier implementations. As shown in Table 4, paper [30] achieves better improvement over software

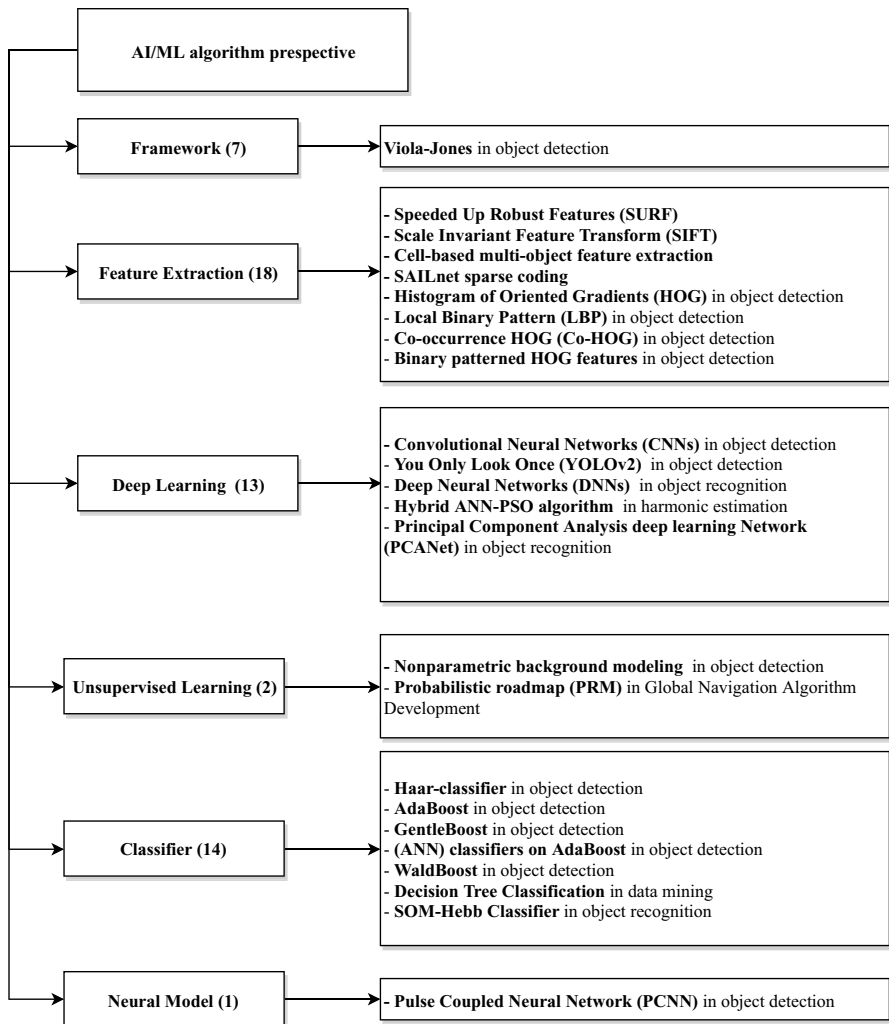


Fig. 6 Summary of algorithms, tools and platforms perspective section's outcomes

more than the other implementation of Haar classifier while using higher resolution images.

Pedestrian detection is one of emerging areas, which is an essential feature in an advanced automated video surveillance system [41]. It aims to improve road safety in real-time automated-traffic-assistance systems [48]. Other traffic related applications such as traffic sign detection was implemented due to the drivers' failure to observe traffic signs [55]. In many fields and applications real-time object tracking is an important feature, any surveillance system needs object tracking to analyse surveillance targets. Other applications, such as robots rely on object tracking to successfully perform navigation tasks [67]. More applications

Table 3 Used AI/ML-based object detection algorithms and tools

Object detection algorithms and tools	Research papers	Num. of papers
<i>Framework</i>		
Viola–Jones	[27, 42–47]	7
<i>Feature descriptors</i>		
HOG	[38, 40, 48–50]	5
LBP	[37, 41, 51–53]	5
Co-occurrence HOG (Co-HOG)	[39]	1
Binary patterned HOG features	[54]	1
SURF	[55]	1
<i>Deep learning</i>		
CNNs	[7, 33, 34, 56–58]	6
You Only Look Once (YOLOv2)	[59]	1
<i>Unsupervised learning</i>		
Nonparametric background modelling	[60]	1
<i>Classifiers</i>		
Haar-classifier	[28–31, 61, 62]	6
AdaBoost	[35, 36]	2
GentleBoost	[63, 64]	2
(ANN) classifiers on AdaBoost	[32]	1
WaldBoost	[65]	1
<i>Neural model</i>		
Pulse coupled neural network (PCNN)	[66]	1

in image analysis using AI such as corner detection, dynamic texture video segmentation, object segmentation are also addressed in the literature [68–70].

For object recognition, thirteen papers are covered in the systematic literature review. Table 5 shows all algorithms that have been used in implementing object recognition.

As shown in Table 5, deep neural networks (DNNs) are the most used algorithm for implementing object recognition. Some researchers developed a DNN algorithm for handwritten digit recognition and phoneme recognition and implemented it on FPGA [71]. The implementation showed a throughput that is about one quarter of GPU-cased system, but on the other hand it results in over 10 times of power efficiency compared to a GPU-based system. A class of DNNs which are CNNs are also mainly used in object recognition [72–74]. They are commonly used for object recognition due to its high accuracy. For example, DeepID, the face recognition method based on deep neural network, reached an accuracy rate of 99.53% for the Labelled Faces in the Wild (LFW) face recognition evaluation database [72]. CNNs, requires extensive resources and computations where the accuracy is determined by the depth of the network. They demand a very large number of arithmetic and memory access operations. So hardware

Table 4 Comparison between two FPGA Haar-classifier implementations

Paper ref	[30]		[31]		
Hardware used	Virtex-5 LX110T	Virtex-5 LX330T	Virtex-5 LX110		Virtex-5 LX155
Resolution	256 × 192		320 × 240		640 × 480
N-classifier LUT	N:1 3456	N:16 23,104	N:1 64,143	N:3 79,537	N:1 66,851 N:3 84,232
BRAM	1066	2290	41		97
DSP48E	4	42	7		7
Improvement over SW	×10 speedup	×20 speedup	×37.33 speedup		×18.81 speedup
Frames per sec	37	98	60		60

Table 5 List of AI/ML-based object recognition algorithms used in research papers

Object recognition algorithms	Research papers	Number of papers
DNNs	[71–74]	4
SOM-Hebb classifier	[75]	1
PCANet	[76]	1

implementation of CNNs on platforms such as FPGA, might be limited by the available resources [71, 73]. SOM-Hebb classifier is also used in object recognition [75]. It is a hybrid network that employs a single layer feed-forward neural network. Mapping is done by self-organizing maps (SOM) and the network is trained with a Hebbian learning algorithm. Other algorithm such as cascaded Principal Component Analysis deep learning Network (PCANet) is used [76]. PCANet is used mainly due to its simple structure compared to DNNs, which makes it more attractive to be accelerated on FPGA.

Researches also implemented general feature descriptors algorithms on hardware. According to our analysis, four methods are found and they are listed in Table 6. Speeded up robust features (SURF) is used more commonly as feature extractor on FPGAs. This is due to its low power consumption. As stated in [77], the proposed FPGA-SURF consumes less than 10 W and occupies less space than a GPU-based system [77]. Other algorithms such as scale invariant feature transform (SIFT) and cell-based labelling are also implemented for feature extraction on FPGAs [78, 79]. The [80] paper presents an ASIC implementation for feature extraction that achieves high energy efficiency and high throughput.

Object tracking implementation on FPGAs was considered by [82, 83]. In [82], an object tracking system that uses mean shift tracking algorithm was implemented. This implementation sets new position of moving object within 1.3% of frame duration. Another implementation of object tracking was done by [83] that was based on background subtraction. The speedup of this FPGA-based tracker is higher for video inputs with higher number of objects. Thus, this implementation is suitable

Table 6 Feature extraction algorithms implementation on FPGAs

Method name	Description
SURF	Extracts salient points from the image and computes descriptors of their surroundings that are invariant to scale, rotation and illumination changes [55, 77, 81]
SIFT	Invariant features to image transition, scaling, rotation, illumination and limited ranges of viewpoint changes are detected, extracted from the image and then transformed to descriptive vectors named as feature descriptors [78]
Cell-based multi-object feature extraction	Clearly binarized image are analysed to extract connected regions based on connected component labelling, after blob inspection which tracks and counts objects in moving scene [79]
SAILnet sparse coding	The sparse coding processor is built to mimic the feature extraction performed by the primary visual cortex [80]

for video input that has dense traffic environments. Two real-time face trackers were implemented on GPUs that used particle filtering algorithm [67], and fast robust tracking algorithm [42] (Table 7).

Due to the complexity of artificial neural networks, training times for a single network could take weeks or months [88]. After analyzing the collected research papers, different types of neural networks accelerators are implemented. These types are accelerators for convolutional neural networks (CNNs), recurrent neural networks, and binarized neural networks. The most implemented neural network accelerators among these types are CNN accelerators. Many AI related frameworks have been developed by researchers. Most of the frameworks are built to implement neural networks in FPGAs [89–93]. In specific, implementing CNNs is the focus of researchers. List of the frameworks is mentioned below.

- DeepBurning [89]: An energy-efficient platform used to implement neural networks in FPGA or chip design. The goal is to simplify the design flow of NN-based accelerators for ML/AI applications.
- DNNWEAVER [90]: A framework used to generate a synthesizable accelerator for a given (DNN, FPGA) pair using hand optimized templates. It uses Caffe [94], as the programming interface to provide a high-level abstraction to programmers.
- Caffeinated FPGAs [91]: This is an adaptation of the Caffe CNN framework with support for Xilinx SDAccel, that allows CNN classification to be launched on CPU-FPGA systems.
- FINN [92]: A framework used to build scalable and fast BNN inference accelerators on FPGAs. The generated accelerators can perform millions of classifications per second with sub-microsecond latency. This makes them ideal for real-time embedded applications such as augmented reality, autonomous driving and robotics.
- FPDeep [93]: A framework used to optimize the mapping of CNN training logic to multiple FPGAs and generate the resulting RTL implementation.
- fpgaConvNet [95]: A framework used for mapping convolutional neural networks on FPGAs. The proposed methodology captures ConvNets by means of an analytical Synchronous Dataflow (SDF) model. SDF is a process for networks where nodes produce and consume a fixed number of data items per firing which allows static scheduling

Table 7 Other applications AI/ML-based algorithm and tools

Application	Algorithms used
Object segmentation	Visual background extractor (VIBE) [84], mixture of Gaussians model (MOG) [70]
Data mining	Decision tree classification (DTC) [85]
Global navigation algorithm development	Probabilistic roadmap (PRM) [86]
Harmonic estimation	Hybrid ANN–PSO algorithm [87]

- Samragh et al. [96]: A framework used to customize the execution of DNNs on FPGA platforms. This is done by employing a reconfigurable clustering approach that encodes the parameters of DNNs in accordance with the application's accuracy requirement and the underlying platform constraints.
- Liu et al. [97]: A parallel framework used to exploit four levels of parallelism to implement deep CNNs on FPGA platforms.
- Field Programmable DNN (FP-DNN) [98]: A framework that has an input of TensorFlow-described DNNs, used to generate the hardware implementations on FPGA boards with RTL-HLS hybrid templates
- Wei et al. [99]: A framework used to automate CNN mapping onto the systolic arrays on FPGAs.
- TuRF [100]: CNN acceleration framework inspired by efficient CNN architectures and transfer learning, which supports domain-specific optimizations. It efficiently deploys domain-specific applications on FPGA.
- Caffe [94]: An open source deep learning framework use to provide a clear access to deep architectures. The code is written in clean, efficient C++, where CUDA is used for GPU computation.
- clCaffe [101]: An OpenCL acceleration of a well-known deep learning framework, Caffe [94], while focusing on the convolution layer which has been optimized with three different approaches. It greatly enhances the ability to leverage deep learning use cases on all types of OpenCL devices.

There are many open-source tools supported by NVIDIA and Intel to support acceleration of machine learning algorithms. For example, NVIDIA developed RAPIDS, which is a GPU-accelerated machine learning library that aims to significantly speed up the training process. RAPIDS improves the performance by accelerating the full workflow pipeline including data preparation, machine learning algorithm and visualization on GPUs. RAPIDS also enables vastly accelerated processing, training on larger dataset sizes and supports multi-GPU deployment. RAPIDS has been used in [102] to successfully decrease the needed training time for recommender systems, which is computationally considered intensive by a speed up of a factor of 15.6 times.

NVIDIA also developed TensorRT is a programmable inference accelerator. It allows people, who develop neural networks and artificial intelligence, to run those networks in production or devices with full performance that GPUs can offer. It also applies the reduced precision optimization on inference that can significantly reduce application latency and make it suitable for real-time services and embedded applications. TensorRT was used in fpga-ConvNet framework in order to obtain highly optimized mappings [95].

Intel provided Intel Math Kernel Library for Deep Neural Networks (MKL-DNN). It is a performance-oriented library that provides highly vectorized and threaded building blocks for implementing CNNs with C and C++ interfaces on Intel CPUs and GPUs. There are many applications enabled with Intel MKL-DNN such as Intel distribution of Caffe framework. It improves performance of this deep learning framework when running particularly on Intel R Xeon processors. For example, IntelCaffe was used in [103] to map CNNs into Intel Xeon Phi scalable

processors by computing the CNN with 8-bit quantization of weights and activations with the usage of hardware support and Intel MKL-DNN library. The low precision inference and the hardware support by Xeon Phi processor lead to compute more operations per second and reduce the access pressure. As a result, this achieves higher throughput with lower latency.

4.4 Hardware implementation perspective

This section discusses the hardware-implemented AI and ML accelerators based on the standard figures of merit (Fig. 7). Hardware accelerators' papers are categorized from hardware perspective to provide useful comparisons. These categories are five as listed below:

- Hardware mapping
- FPGA-based accelerator design
- GPU-based accelerator design
- Reclassification of accelerators
- Xeon-Phi accelerator design

4.4.1 Hardware mapping

An emerging challenge for researchers is to find a suitable convolutional neural network algorithm for a specific application and to map it efficiently on hardware [21]. CNNs are built using feed-forward propagation networks which can be considered to be a large number of similar scalar operations communicated through different stages. This type of computations gains a significant speed up when running on FPGAs [104]. The FPGA mapping challenges include finding an efficient fitting between the computational model of the CNN and the execution model supported by the FPGA [104]. Several research papers addressed these issues of CNN mapping on FPGAs [21, 89, 104–110].

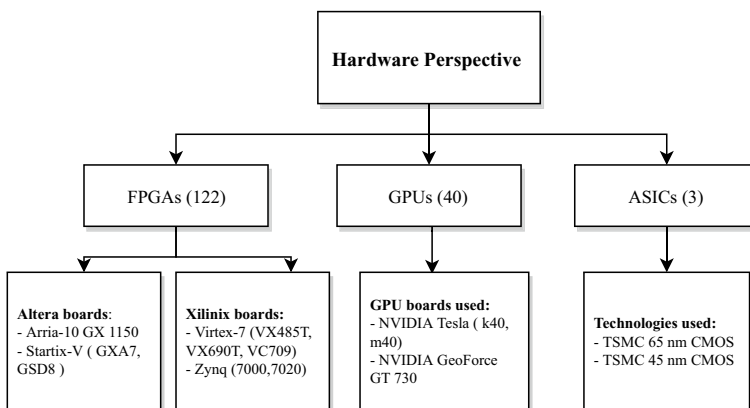


Fig. 7 Summary of hardware perspective section's outcomes

In [105], the authors developed a heuristic search algorithm and a dataflow instruction set architecture (ISA). The proposed design used DNNWEAVER to generate high-performance accelerators operating within a limited power budget and on-chip memory of the FPGA. DNNWEAVER is a framework that automatically generates a synthesizable accelerator for a given (DNN, FPGA) pair using hand-optimized templates [90]. In [89], a framework called DeepBurning was built to simplify the operation of mapping diverse neural networks into FPGAs or ASICs. In [21], the authors proposed a library-based RTL compiler to automatically generate customized FPGA-based accelerators for a given CNN algorithm. Comparing [21, 21, 89] obtained more than 6.4× speedup compared to [89]. Other library-based RTL compilers were proposed in [106, 110, 111] to accelerate CNNs on FPGA platforms.

In [104], the authors investigated the feasibility of direct hardware mapping of CNNs on FPGA. The proposed model claimed to open new opportunities for further optimization and can be extended to ASIC technologies as well as binary neural networks. A latency-driven design methodology for mapping ConvNets into FPGAs was proposed in [107]. The authors in [108] developed a python-based automation tool to generate FPGA-based CNN accelerators. The tool consistently delivered high throughput designs for various CNN models. Other papers such as [109] proposed f-CNNx, which is an automated toolflow for the optimized mapping of multiple CNNs on FPGAs.

Comparisons between these research papers according to type of neural network used is presented in the tables below. As shown in Tables 8, 10, 11 and 12, the library-based RTL compiler [110] achieved the best performance which is 732.36 GOPS, 1604.57 GOPS, 651.49 GOPS and 789.44 for NiN, VGG-16, ResNet-50 and ResNet-152 respectively on Startix 10 GX2800 FPGA. For AlexNet, the python-based automation tool [108] reported the best performance which is 274.5 GOPS as shown in Table 9.

Table 8 Comparison between hardware mapping of the CNN model NiN neural network on FPGA

	[111]	[21]	[106]	[110]
FPGA	Stratix-V GXA7	Stratix-V GXA7	Stratix-V GXA7	Intel Arria 10 GX 1150
Tech.	28 nm	28 nm	28 nm	20 nm
Clock (MHz)	100	150	100	240
Precision	Fixed 8–16 bits	16-bits	Fixed 8–16 bits	Fixed 16-bits
DSP	256 (100%)	256 (100%)	256 (100%)	3036 (100%)
Latency/image (ms)	18.75	7.79	18.75	3.01
Performance (GOPS)	117.3	282.67	117.3	732.36

Table 9 Comparison between hardware mapping of CNN model AlexNet neural network on 28 nm FPGA

	[111]	[106]	[107]	[108]
FPGA	Stratix-V GXA7	Stratix-V GXA7	Zynq XC7Z045	Stratix-V GXA7
Clock (MHz)	100	100	125	200
Precision	Fixed 8–16 bits	Fixed 8–16 bits	Fixed 16-bit	Fixed 16-bit
DSP	256 (100%)	256 (100%)	900	256 (100%)
Latency/image (ms)	9.92	9.92	8.22	12.61
Performance (GOPS)	114.5	114.5	187.80	274.5

4.4.2 FPGA-based accelerator design

Many FPGA-based accelerator architecture designs were proposed [112–152]. Most of the research papers presented FPGA-based CNN accelerators. From hardware perspective, these papers are categorized into two categories:

- Heterogeneous architecture
- FPGA

Heterogenous architecture accelerators, such as the one presented in [139], uses high-end CPU with an FPGA to implement a high-performance binarized neural networks (BNN) accelerator. The proposed accelerator is designed to take advantage of the Xeon CPU+FPGA system. The FPGA architecture is targeted for the most compute intensive parts of the BNN, while other parts of the topology can be handled by the CPU. In [128, 140], other CPU-FPGA-based CNN accelerators were proposed. The computationally intensive and universal operations (e.g. 2D convolution) were implemented in FPGA.

Other heterogenous architectures, combining ASICs with FPGAs as the one proposed in [146], focused on deep learning domain with efficient tensor matrix/vector operations. Authors in [114] proposed an accelerator architecture using two FPGAs. An implementation of CNN FPGA-based accelerator reached 82.8% accuracy in [125].

Many CNN models have been accelerated on FPGAs that differ in size. Figure 8 shows CNN models versus their size. The size of the implemented CNN is measured by either Giga Operations (GOP) in fixed point implementations and Giga Floating Point Operations (GFLOP) in floating point implementation.

Xilinx FPGA-based CNN accelerators are compared in Figs. 9 and 10. Figure 9 contain the fixed-point implementations where papers are grouped according to the precision used in the implementation, whereas Fig. 10 illustrates the difference between 32-floating-point FPGA-based CNN accelerators. As noticed from Fig. 9, [137] has the highest performance among the other implementations. Their implementation achieves state-of-art performance. This is due to the cross-layer scheduling that have been used which reduced data transfer from 6.6MB to 0.2MB with a 97% reduction. Figure 10 which refers to 32-floating-point

Table 10 Comparison between hardware mapping of CNN model VGG-16 neural network on FPGA

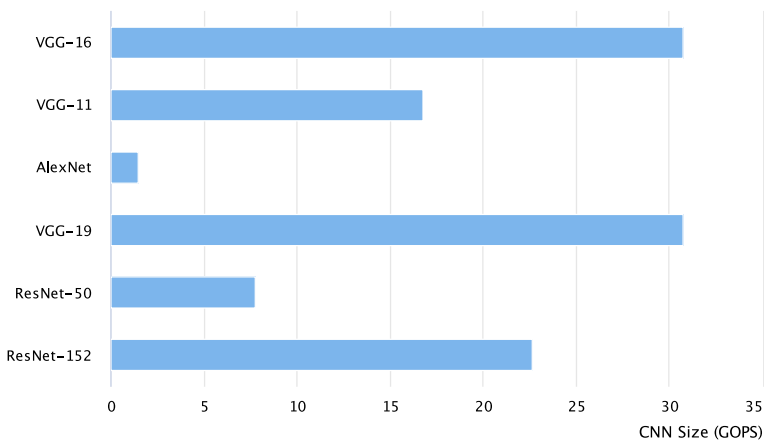
	[22]		[110]		[107]	[108]
FPGA	Stratix V GXA7	Arria-10 GX 1150	Arria 10 GX 1150	Stratix 10 GX 2800	Zynq XC7Z045	Stratix-V GXA7
Tech.	28 nm	20 nm	20 nm	14 nm	28 nm	28 nm
Clock (MHz)	150	200	240	300	125	200
Precision	Fixed 16-bit		Fixed 8/16-bit		Fixed 16-bit	Fixed 16-bit
DSP	256 (100%)	1518 (100%)	3036 (100%)	8216 (71%)	900 (100%)	256 (100%)
Latency/image (ms)	87.87	42.98	31.97	19.29	249.50	16.10
Performance (GOPS)	352.24	720.15	968.03	1604.57	123.12	660.9

Table 11 Comparison between hardware mapping of CNN model ResNet-50 neural network using fixed 16-bit on FPGA

	[22]		[110]	
FPGA	Stratix V GXA7	Arria-10 GX1150	Arria 10 GX1150	Stratix 10 GX2800
Tech.	28 nm	20 nm	20 nm	14 nm
Clock (MHz)	150	200	240	300
DSP	256 (100%)	1518 (100%)	3036 (100%)	6304 (100%)
Latency/image (ms)	30.88	12.51	12.87	11.85
Performance (GOPS)	250.75	619.13	599.61	651.49

Table 12 Comparison between hardware mapping of CNN model ResNet-152 neural network using fixed 16-bit on FPGA

	[22]		[110]	
FPGA	Stratix V GXA7	Arria-10 GX1150	Arria 10 GX1150	Stratix 10 GX2800
Tech.	28 nm	20 nm	20 nm	14 nm
Clock (MHz)	150	200	240	300
DSP	256 (100%)	1518 (100%)	3036 (100%)	6304 (100%)
Latency/image (ms)	81.19	31.85	32.37	28.59
Performance (GOPS)	278.67	710.30	697.09	789.44

**Fig. 8** CNN size in GOPS based on model used. (Considering only model sizes that were reported by research papers)

FPGA-based CNN accelerators shows that [117] reported the highest performance. This work achieves high performance due to the several sources of parallelism integrated into the implementation of CNNs which aim The purpose of them is to achieve the best possible speedup. These parallelism techniques are:

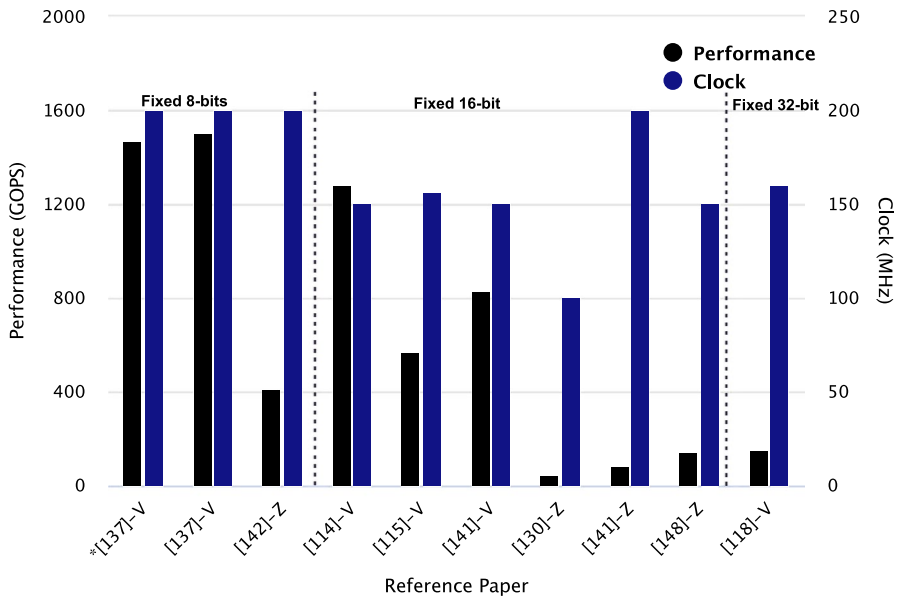


Fig. 9 Performance comparison of fixed point FPGA-based CNN accelerators—Xilinx boards. *V: Implemented on Virtex-7, *Z: Implemented in Zynq

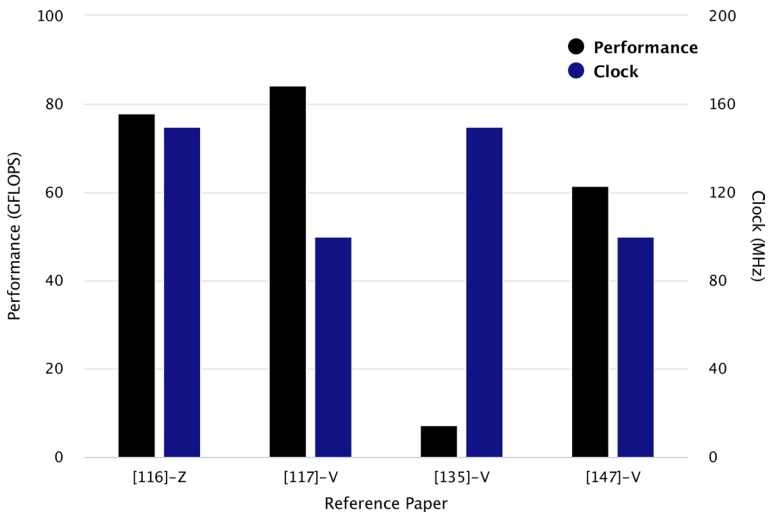


Fig. 10 Performance comparison of 32-floating point FPGA-based CNN accelerators—Xilinx boards. Note: V refers to Implementation on Virtex-7 and Z refers to Implementation in Zynq. *: Two implementation of the same paper using different clocks

- Inter layer parallelism: Different layers that does not have data dependency cannot be executed in parallel.
- Inter output parallelism: Different output feature maps are totally independent of each other. Therefore, all of them can be computed in parallel.
- Inter kernel parallelism: Since the convolutions are independent for different output pixels parallelism can be exploited.
- Intra kernel parallelism: A convolution is a set of multiplications and additions. As each multiplication between a weight in a kernel and a pixel in an input feature map is independent from another multiplication, they can be performed concurrently.

The Remaining FPGA-based CNN accelerators were implemented using altera boards. These implementations are compared in Figs. 11 and 12. For Fixed-point implementation, [119] reported the highest performance among the other implementations. They addressed the performance limitation that is caused by small on-chip memory bandwidth by a two-dimensional interconnection between processing elements (PEs) and local memory. This effectively increases the effective on-chip memory bandwidth. Whereas for 32 floating-point implementations on altera boards, [124] reported the highest performance. The techniques that were used to achieve this high performance was usage of an on-chip stream buffer that efficiently stores input and output feature maps. Also, they used a vectorization approach that

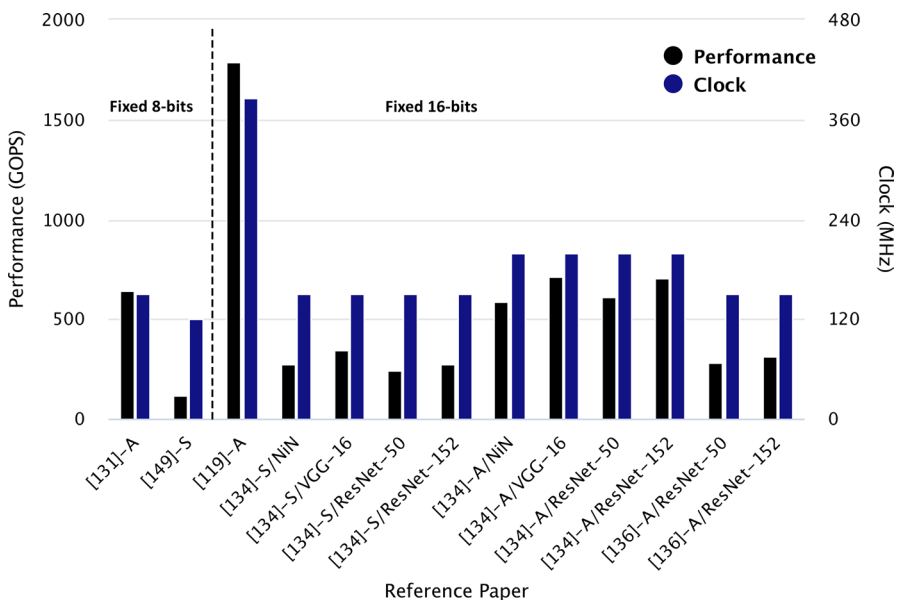


Fig. 11 Performance comparison of fixed point FPGA-based CNN accelerators—Altera boards. *A: implemented on Arria 10, *S: implemented on Startix-V

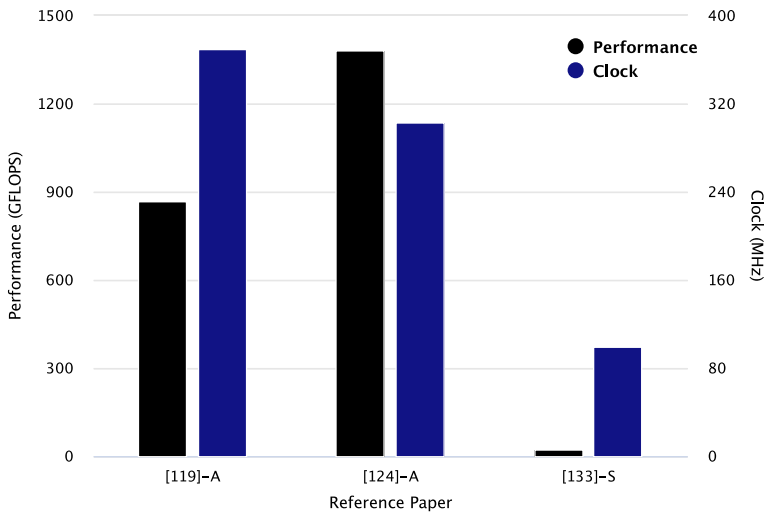


Fig. 12 Performance comparison of 32-floating point FPGA-based CNN accelerators—Altera boards. *A: implemented on Arria 10, *S: implemented on Startix-V

achieves over 60% DSP efficiency. This approach uses the Winograd transform to significantly reduce the DSPs required to perform the convolution layers in CNNs.

Other papers used FPGA-embedded processors such as NIOS-II, [113], and ARM processors in SoC-based FPGAs, [138], to implement the target accelerators. In [104], a pre-trained feed-forward ANN is executed on a NIOS-II processor where a custom instruction set is used to approximate the hyperbolic tangent function. In [128], the acceleration is achieved by utilizing the parallelized programmable logic to implement the convolution, pooling and padding, along with the on-board ARM processor to operate and execute the remaining tasks.

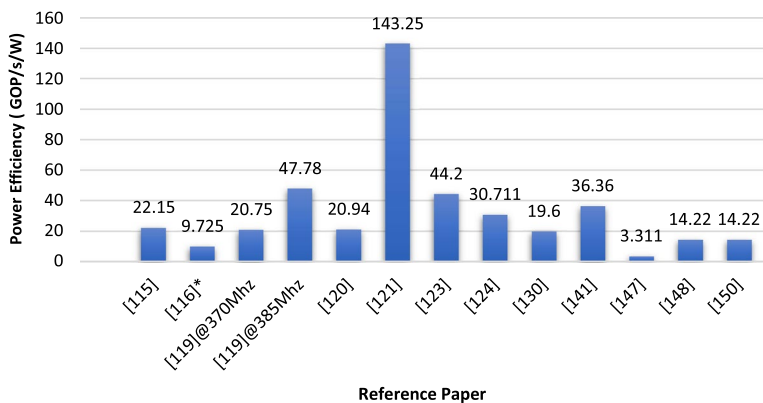


Fig. 13 Power efficiency reported of FPGA-based CNN accelerators (only papers that reported power efficiency is mentioned in the figure). *The power consumption of only the two-FPGA chips not the whole board

Most of the implementation reported power efficiency. Therefore, the implementations are also compared from the power efficiency perspective as shown in Fig. 13. The most efficient implementation from this perspective is the work done by Nakahara et al. This work implemented a binarized CNN. They realized the high-performance with less hardware by proposing many techniques. These techniques are:

- Shared XNOR-MAC circuit to support a streaming operation
- Usage of the shift register to make a streaming data flow from the memory for the feature map
- Sharing the different size of XNOR-MAC circuit into a single bitwise XNOR circuit followed by adder-trees, bias adder, and a write controller.

To further analyse the trend of these FPGA-based CNN accelerators. Figure 14 shows the number of papers that used each CNN model. As can be noticed from the figure, AlexNet and VGG-16 are the most CNN models implemented by researchers.

Also, comparisons between the precision chosen for implementation versus the CNN size are illustrated in Fig. 15. Most FPGA-based CNN accelerators implementations used fixed point representation either 8, 16 bit. The bottleneck of FPGA-based CNN implementation is the high-precision weights. This means that high-precision multipliers are needed, which costs lots of FPGA resources especially for large and deep CNNs [150]. Precision choice greatly affects the energy efficiency as the implementations that uses 48-bit fixed-point data or 32-bit floating point data achieves much lower energy efficiency [140]. This is due to the large memory access that needs much energy consumption to load high-precision weights [150].

FPGAs supports the flexibility of choosing any precision that is sufficient for the target CNN implementation. However, FPGA mappings typically rely on DSP blocks for highest efficiency. DSP blocks come with a fixed precision only. As

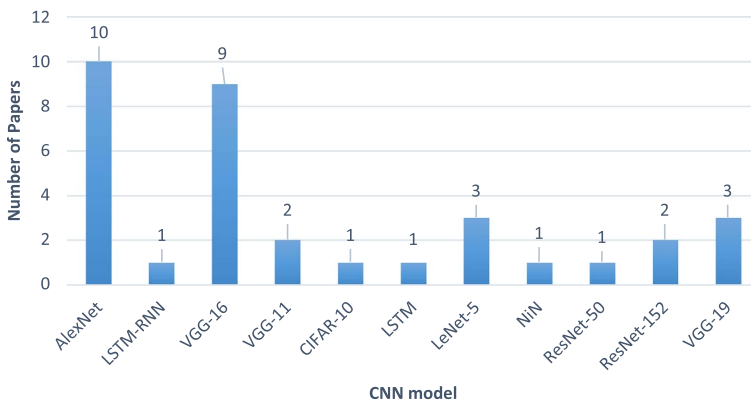


Fig. 14 Most used CNN models in FPGA-based CNN accelerators implementations

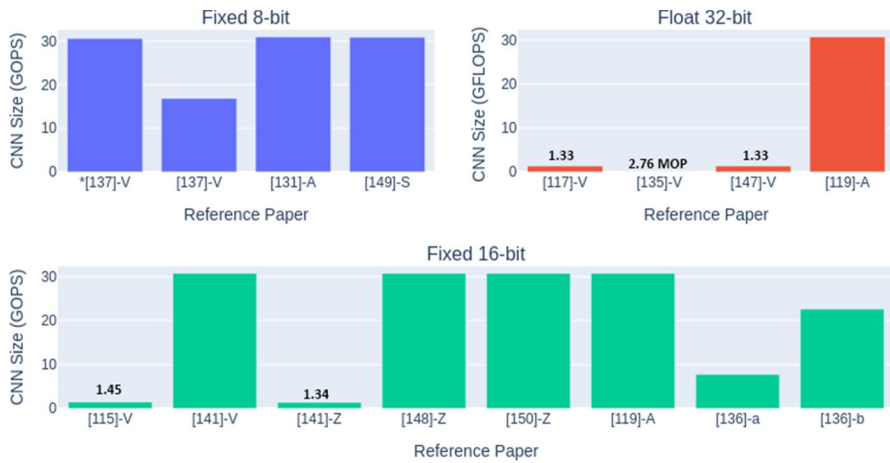


Fig. 15 Precision chosen VS CNN size

a result, using fixed-point quantization can improve the performance and save hardware resources as well [132, 151]. 16-bit fixed point is preferable over the 32-floating point because it can significantly reduce the resource requirements of each processing elements (PE) [124].

Despite the large size of CNNs, they are resilient up to 8-bit precision [132]. Therefore, some implementation used 8-bit fixed point that achieved more than 50% performance improvement over the 16-bit fixed implementation [140]. Recently, a binarized CNN (BCNN) whose weights and activation are restricted to 2-values (+1/−1) is proposed. BCNN can lead to dramatic improvements in performance and power efficiency in well-optimized software on the CPU and GPU. Some research papers focused on accelerating these BCNN in FPGA [24, 120, 121, 123,

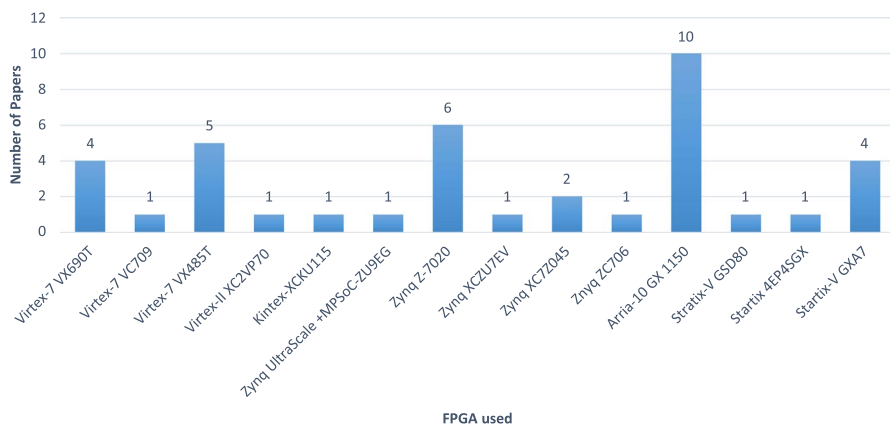


Fig. 16 Most used FPGAs in FPGA-based CNN accelerators implementations

[143](#)]. Also FPGA-based CNN accelerators are analysed from the FPGA board used perspective as shown in Fig. 16. The most used FPGA is Intel Arria 10 GX 1150.

4.4.3 GPU-based accelerator design

GPUs have been becoming significant in accelerating deep learning. This section will cover GPU-based accelerator designs that have been proposed by researchers in the covered period of this systematic literature review. Table 13 summarizes the work done on GPU-based accelerators that focuses on general proposed methods. These methods aim at accelerating algorithms on GPUs.

4.4.4 Reclassification of accelerators

In [\[161\]](#), authors proposed a new classification approach for accelerators. This classification takes into account the evolution of the design technology of integrated circuits and the design techniques.

4.4.5 Xeon-Phi accelerator design

Throughout the last decade, Intel Xeon Phi co-processor has paved the way of success in implementing deep learning algorithms. The Intel Xeon Phi is a shared-memory. This means many-core coprocessor that comprises up to 61, 1.2 GHz cores and each core can switch among 4 hardware threads in a round-robin manner. Xeon Phi makes heavy use of Single Instruction Multiple Data (SIMD) technology, which allows performing the same operation on multiple data segments simultaneously. Thus, Xeon Phi is highly suitable for deep learning application, which usually makes use of simple operations over large tensors.

Researchers proved that the Intel Xeon Phi can offer an efficient, but more general purposed way to parallelize the deep learning algorithm compared to GPUs [\[162\]](#). They demonstrated that deep learning can be optimized and scaled effectively on many-core, HPC systems [\[163\]](#). Fast-direct convolution kernels for training CNNs on Xeon and Xeon Phi systems is proposed in [\[164\]](#). These convolution kernels are implemented via a dynamic compilation approach. Moreover, researchers accelerated the training process for CNNs by proposing parallelization schemes such as Controlled Hogwild with Arbitrary Order of Synchronization (CHOAS) in [\[165\]](#). CHOAS exploits both the thread- and SIMD-level parallelism and achieves speedup of up to 103× compared to the execution on one thread of the Xeon Phi. Also, CHOAS achieves speedup of up to 58× compared to the sequential execution on Intel Core i5.

Authors in [\[166\]](#) proposed CosmoFlow which is the first large-scale science application of the TensorFlow framework at supercomputer scale with fully synchronous. They optimized the full software stack, which includes network design, I/O processing pipeline, communication, TensorFlow framework and 3D CNN primitives in MKLDNN for our 3D convolutional neural network. Some researchers focused on BNNs where a novel approach for optimizing BNNs on CPUs by a framework named BitFlow was proposed. BitFlow obtains 1.8× speedup over

Table 13 Summary of work done on GPU-based accelerator

References	Scope	Results
[153]	Neural network acceleration (NNA)	2.4× average speedup and a 2.8× average energy reduction compared with the baseline GPU architecture while the trade-off showed a 10% quality loss margin across a diverse set of benchmarks
[154]	Convolutional neural networks (CNNs) acceleration	2–24 times faster than the CPU
[155]	Standard back-propagation algorithm for training multiple perceptrons simultaneously on GPUs	50× speed increase compared to a highly optimized CPU-based computer program and more than 150× speed up compared to a CPU-based software (NeuroShell 2)
[156]	Accelerating training process of large scale recurrent neural networks	2–11× speed-up compared with the basic CPU implementation with the Intel Math Kernel Library
[157]	Memory utilization in multi-GPU systems for deep learning application acceleration	Increase the mini-batch size up to 60% and improve the training throughput up to 46.6% in a multi-GPU system
[158]	Large scale spiking neural networks	Speedups greater than 110× on the GPU, 80× on Xeon, 70× on PS3, and 60× on AMD over the serial implementation on a 2.66 GHz Intel Core 2 Quad
[159]	Recurrent neural networks language model	3.4× speedup on 4 GPUs than the single one, without any performance loss in language model perplexity
[160]	Cellular neural network (CNN)	The inherent massive parallelism of CNN along with GPUs make it an advantage for high-performance computing platform and the design is portable on all the available graphics processing devices and multi core processors

unoptimized BNN implementations on a single core of Intel Xeon Phi and 11.5× speedup over counterpart full precision DNNs. Whereas over 64 cores of Intel Xeon Phi, it obtains 10% over its counterpart full-precision DNNs on GPU (GTX 1080) [167].

5 SLR discussions and recommendations

In this section, we discuss the best implementation approach, FPGAs, ASICs or GPUs, with respect to standard figures of merit. Table 14 shows a comparison between the three approaches.

Since ASIC design cannot be altered after fabrication, it is not suitable for application areas where the design needs to be updated after the implementation. ASIC is best used when the target is low power and area. In the literature, ASICs are used for accelerators handling images and videos in mobile devices [80]. Unlike ASICs, FPGAs are flexible for changes and updates after implementation. Also, FPGAs' implementation time is considered less than ASICs. It is worth noting that FPGAs are often used for prototyping and validation, even if the final target is ASIC implementation.

Most FPGA boards improve speed through on board supporting blocks for real-time implementation. Example prices of FPGA boards used for acceleration in the surveyed literature: Xilinx virtex-5: 899\$, Altera Stratix-V GXA7: \$6995, Intel Arria 10 GX: \$4495 [168–170]. In these examples, FPGAs are mainly used to implement image processing applications such as real-time object detection and recognition. FPGAs is considered an easy and suitable solution for a real-time processing system. The power consumption of the surveyed implementations ranged from 1.7 W [76] to 20 W [81]. On the other hand, the power consumption of ASIC implementations

Table 14 Platform recommendation for AI acceleration

	FPGA	ASIC	GPU
Required technical skills	VHDL/Verilog, FPGA development environment	VHDL/Verilog, CAD tools for chip fabrication	GPU high-level programming skills
Implementation time/Time to market	Medium	High	Medium
Implementation level	Hardware	Hardware	Software
Implementation flexibility	High	High	Low
Flexibility for changes after implementation	High	Low	High
Cost	Medium	High	Medium to low
Energy efficiency	Medium	High	Low
Area efficiency	Low	High	Low
Performance	Medium	High	Medium to low

ranged from 6.67 mW [80] to 15.97 W [171]. Note that ASIC implementation yields less power consumption than that of the FPGA of the same system [35].

GPUs provide a software level solution, whereas FPGAs and ASICs provide a hardware level solution. Thus, FPGAs and ASICs provide higher flexibility during the implementation phase compared to GPUs. As a result, GPU implementation is restricted by the existing underlying hardware, while FPGAs and ASICs depend entirely on the design optimization carried out during the design phase. Thus, FPGAs can be faster than GPUs in some cases.

One key feature in GPUs is that they are designed with a fast memory hierarchy with direct memory access to resolve memory bandwidth issues. This allows higher transfer rates while taking less time. Also, GPUs stand out in floating-point operations such as signal and image processing applications thanks to the native floating-point processors. The implementation cost for both GPU and FPGA is considered medium compared to ASIC. This is because of the high fabrication cost of ASICs. The following are example prices of GPUs used for acceleration: Tesla k40: \$841.00, m40: \$769.98, GeForce GT 730: \$53 [172–174].

The usage of hardware accelerators whether it is based on FPGAs, ASICs or GPUs is expanding day by day. It can be especially seen in the implementations of AI and ML algorithms. The algorithmic superiority of these algorithms demands extremely high computational power and memory usage, which can be achieved by hardware accelerators. AI and ML are essential technologies that support daily social life, economic activities, and even medical purposes. Hence, the continuous development of these algorithms will create new applications with higher resources' demands. For instance, speech recognition systems will reach considerably higher levels of performance enabling it to communicate with humans, using both text and voice. Moreover, hardware based-expert systems are emerging in all aspects of health care, in both clinical and administrative areas. Accordingly, this improves patient care as well as the allocation of financial, social, and other resources. Implementing these algorithms on hardware will result in more rapid, efficient and accurate AI processing which can reflect positively in all fields of industry. Finally, Table 15 summarizes the main outcomes of the research study.

6 Conclusion and future work

In this paper, we present a systematic literature review (SLR) study comparing existing AI and ML accelerators. The paper covers hardware implementation research over the period between 2009 and 2019. The main objective of the SLR is to answer three research questions addressing the subject from three angles, namely application perspective, AI algorithms and tools perspective, and hardware platform perspective. The work is restricted to journal and conference papers proposing hardware acceleration implementations using FPGAs, GPUs and ASICs. A careful filtration strategy has been applied in order to select the research papers that can answer the research questions. A comparative analysis of the collected research papers is presented to study the emerging AI/ML hardware accelerators. The work presents a thorough analysis of different implementation platforms, tools, and approaches used

Table 15 Platform recommendation for AI acceleration

Research question	Research question description	Survey results
RQ1	Main applications that use AI and ML tools in hardware implementation	Image processing is the most application that uses AI and ML in hardware implementation with a percentage of 93%. Other applications are Data mining, Ambient Intelligence, Robotics, Resources Gauge Model and Global navigation
RQ2	Most commonly used AI and ML algorithms in hardware implementation	Analysis is done from application perspective. Results were as listed below: Object detection: Haar-classifier algorithm Object recognition: Deep neural networks Feature extraction: SURF Object segmentation: VIBE and MOG Data mining: DTC Global navigation algorithm development: PRM Harmonic estimation: hybrid ANN–PSO algorithm List of frameworks that have been developed is listed such as DeepBurning, Caffe, fpgaConv, DNNWEAVER, etc. The list includes a description that describe the task that can be achieved using the framework
RQ3	What is the best hardware platform used for a given application	Comparison between hardware mapping of NiN, AlexNet, VGG-16, ResNet-50 and ResNet-152 on FPGAs is provided. The comparison is made based on type of FPGA used, type of neural network, clock, precision, DSP usage, Latency/image and performance in GOPS. Also, comparison between all FPGA-based CNN accelerators is provided based on type of neural network, problem complexity, performance and power

in the last decade. Finally, the paper put forward necessary recommendations for hardware selection in terms of cost, development time, and performance results.

Acknowledgements We would like to thank the University of Sharjah and OpenUAE Research and Development Group for funding this research study. We are also grateful to our research assistants who helped in collecting, summarizing and analysing the 169 research papers used in this SLR study.

References

1. Sze V, Chen Y-H, Yang T-J, Emer JS (2017) Efficient processing of deep neural networks: a tutorial and survey. *Proc IEEE* 105(12):2295–2329
2. Pau LF (1991) Artificial intelligence and financial services. *IEEE Trans Knowl Data Eng* 3(2):137–148
3. Yao X, Zhou J, Zhang J, Boer CR (2017) From intelligent manufacturing to smart manufacturing for industry 4.0 driven by next generation artificial intelligence and further on. In: 5th International Conference on Enterprise Systems (ES)
4. Bishnoi L, Narayan Singh S (2018) Artificial intelligence techniques used in medical sciences: a review. In: 8th International Conference on Cloud Computing, Data Science and Engineering (Confluence), pp 106–113
5. Parker DS (1989) Integrating AI and DBMS through stream processing. In: Proceedings of Fifth International Conference on Data Engineering
6. Fraley JB, Cannady J (2017) The promise of machine learning in cybersecurity. SoutheastCon
7. Farabet C, Poulet C, Han JY, LeCun Y (2009). CNP: an FPGA-based processor for convolutional networks. Presented at the 2009 International Conference on Field Programmable Logic and Applications (FPL)
8. Rao Q, Frtunikj J (2018) Deep learning for self-driving cars. In: Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems—SEFAIS '18
9. Duffany JL (2010) Artificial intelligence in GPS navigation systems. Presented at the 2010 2nd International Conference on Software Technology and Engineering (ICSTE 2010)
10. Schutzer D (1983) Applications of artificial intelligence to military communications. In: IEEE Military Communications Conference, pp 786–790
11. Misra J, Saha I (2010) Artificial neural networks in hardware: a survey of two decades of progress. *Neurocomputing* 74(1–3):239–255
12. Baji T (2018) Evolution of the GPU device widely used in AI and massive parallel processing. In: IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM)
13. Jawandhiya P (2018) Hardware design for machine learning. *Int J Artif Intell Appl (IJAAI)* 9(1):63–84
14. Shawahna A, Sait SM, El-Maleh A (2019) FPGA-based accelerators of deep learning networks for learning and classification: a review. *IEEE Access* 7:7823–7859
15. Lucas SM (2009) Computational intelligence and AI in games: a new IEEE transaction. *IEEE Trans Comput Intell AI Games* 1(1):1–3
16. Rigos S (2012) A hardware acceleration unit for face detection. In: Mediterranean Conference on Embedded Computing (MECO), Bar, pp 17–21
17. Mittal S (2018) A survey of FPGA-based accelerators for convolutional neural networks. *Neural Comput Appl* 32(4):1109–1139
18. Guo K, Zeng S, Yu J, Wang Y, Yang H (2019) [DL] A survey of FPGA-based neural network inference accelerators. *ACM Trans Reconfig Technol Syst* 12(1):1–26
19. Wang T, Wang C, Zhou X, Chen H (2018) A survey of FPGA based deep learning accelerators: challenges and opportunities. *arXiv preprint arXiv:1901.04988*
20. Budgen D, Brereton P (2006) Performing systematic literature reviews in software engineering. In: Proceeding of the 28th International Conference on Software Engineering—ICSE '06
21. Ma Y, Cao Y, Vrudhula S, Seo J (2017) An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks. In: 27th International Conference on Field Programmable Logic and Applications (FPL)

22. Nurvitadhi E, Venkatesh G, Sim J, Marr D, Huang R, Ong Gee Hock J, Liew YT, Srivatsan K, Moss D, Subhaschandra S, Boudoukh G (2017) Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays—FPGA '17
23. Lacey G, Taylor G, Areibi S (2016) Deep learning on FPGAs: past, present, and future, pp 1–8. [arXiv: 1602.04283](https://arxiv.org/abs/1602.04283)
24. Faraone J, Gambardella G, Boland D, Fraser N, Blott M, Leong PHW (2018) Customizing low-precision deep neural networks for FPGAs. In: 28th International Conference on Field Programmable Logic and Applications (FPL)
25. Cheng Kwang-Ting, Wang Yi-Chu (2011) Using mobile GPU for general-purpose computing; a case study of face recognition on smartphones. In: Proceedings of 2011 International Symposium on VLSI Design, Automation and Test
26. Ouerhani Y, Jridi M, AlFalou A (2010) Fast face recognition approach using a graphical processing unit “GPU”. In: IEEE International Conference on Imaging Systems and Techniques
27. Li E, Wang B, Yang L, Peng Y, Du Y, Zhang Y, Chiu Y-J (2012) GPU and CPU cooperative acceleration for face detection on modern processors. Presented at the 2012 IEEE International Conference on Multimedia and Expo (ICME)
28. Shah AA, Zaidi ZA, Chowdhry BS, Daudpoto J (2016) Real time face detection/monitor using raspberry pi and MATLAB. In: IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)
29. Oro D, Fernandez C, Saeta JR, Martorell X, Hernando J (2011) Real-time GPU-based face detection in HD video sequences. In: IEEE International Conference on Computer Vision Workshops (ICCV Workshops)
30. Gao C, Lu SL (2008) Novel FPGA based Haar classifier face detection algorithm acceleration. Presented at the 2008 International Conference on Field Programmable Logic and Applications (FPL)
31. Cho J, Mirzaei S, Oberg J, Kastner R (2009) FPGA-based face detection system using Haar classifiers. In: Proceeding of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays—FPGA '09
32. He C, Papakonstantinou A, Chen D (2009) A novel SoC architecture on FPGA for ultra fast face detection. Presented at the 2009 IEEE International Conference on Computer Design (ICCD 2009)
33. Farrugia N, Mamalet F, Roux S, Fan Yang, Paindavoine M (2009) Fast and robust face detection on a parallel optimized architecture implemented on FPGA. *IEEE Trans Circuits Syst Video Technol* 19(4):597–602
34. Farabet C, Poulet C, LeCun Y (2009) An FPGA-based stream processor for embedded real-time vision with convolutional networks. In: IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
35. Kyrkou C, Theodorides T (2011) A flexible parallel hardware architecture for AdaBoost-based real-time object detection. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 19(6):1034–1047
36. Zhou W, Zou Y, Dai L, Zeng X (2011) A high speed reconfigurable face detection architecture. Presented at the 2011 IEEE 9th International Conference on ASIC (ASICON 2011)
37. Wang N-J, Chang S-C, Chou P-J (2012) A real-time multi-face detection system implemented on FPGA. Presented at the 2012 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS 2012)
38. Bauer S, Brunsmann U, Schlotterbeck-Macht S (2009) FPGA implementation of a HOG-based pedestrian recognition system. In: MPC Workshop, pp 49–58
39. Hiromoto M, Miyamoto R (2009) Hardware architecture for high-accuracy real-time pedestrian detection with CoHOG features. In: IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
40. Bauer S, Kohler S, Doll K, Brunsmann U (2010) FPGA-GPU architecture for kernel SVM pedestrian detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition—Workshops
41. Kryjak T, Komorkiewicz M, Gorgon M (2012) FPGA implementation of real-time headshoulder detection using local binary patterns, SVM and foreground object detection. In: Conference on Design and Architectures for Signal and Image Processing (DASIP), pp 1–8
42. Sharma B, Thota R, Vydyanathan N, Kale A (2009) Towards a robust, real-time face processing system using CUDA-enabled GPUs. In: International Conference on High Performance Computing (HiPC)

43. Kong J, Deng Y (2010) GPU accelerated face detection. In: International Conference on Intelligent Control and Information Processing
44. Hefenbrock D, Oberg J, Thanh NTN, Kastner R, Baden SB (2010) Accelerating Viola-Jones face detection to FPGA-level using GPUs. In: 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines
45. Masek J, Burget R, Uher V, Guney S (2013) Speeding up Viola-Jones algorithm using multi-Core GPU implementation. Presented at the 2013 36th International Conference on Telecommunications and Signal Processing (TSP)
46. Jain V, Patel D (2016) A GPU based implementation of robust face detection system. *Procedia Comput Sci* 87:156–163
47. Lescano G, Santana P, Costaguta R (2017) Analysis of a GPU implementation of Viola-Jones' algorithm for features selection. *J Comput Sci Technol* 17(1):68–73
48. Hahnle M, Saxen F, Hisung M, Brunsmann U, Doll K (2013) FPGA-based real-time pedestrian detection on high-resolution images. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 629–635
49. Komorkiewicz M, Kluczewski M, Gorgon M (2012) Floating point HOG implementation for real-time multiple object detection. Presented at the 2012 22nd International Conference on Field Programmable Logic and Applications (FPL)
50. Ma X, Najjar WA, Roy-Chowdhury AK (2015) Evaluation and acceleration of high-throughput fixed-point object detection on FPGAs. *IEEE Trans Circuits Syst Video Technol* 25(6):1051–1062
51. Dwith CYN, Rathna GN (2012) Parallel implementation of LBP based face recognition on GPU using OpenCL. In: The International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp 755–760
52. Oh C, Yi S, Yi Y (2015) Real-time face detection in full HD images exploiting both embedded CPU and GPU. Presented at the 2015 IEEE International Conference on Multimedia and Expo (ICME)
53. Oh C, Yi S, Yi Y (2018) Real-time and energy-efficient face detection on CPU-GPU heterogeneous embedded platforms. *IEICE Trans Inf Syst E* 101(12):2878–2888
54. Negi K, Dohi K, Shibata Y, Oguri K (2011) Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm. In: International Conference on Field-Programmable Technology
55. Zhao J, Zhu S, Huang X (2013) Real-time traffic sign detection using SURF features on FPGA. In: IEEE High Performance Extreme Computing Conference (HPEC)
56. Nasse F, Thureau C, Fink GA (2009) Face detection using GPU-based convolutional neural networks. In Proceedings of the 13th international conference on computer analysis of images and patterns. Springer, Berlin, pp 83–90
57. Li H, Lin Z, Shen X, Brandt J, Hua G (2015) A convolutional neural network cascade for face detection. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 5325–5334
58. Cengil E, Cinar A, Guler Z (2017) A GPU-based convolutional neural network approach for image classification. Presented at the 2017 International Artificial Intelligence and Data Processing Symposium (IDAP)
59. Tijtgaat N, Ranst WV, Volckaert B, Goedeme T, Turck FD (2017) Embedded real-time object detection for a UAV warning system. *ICCVW. Venice, Italy*, pp 2110–2118
60. Berjon D, Cuevas C, Moran F, Garcia N (2013) GPU-based implementation of an optimized non-parametric background modeling for real-time moving object detection. *IEEE Trans Consum Electron* 59(2):361–369
61. Obukhov A (2011) Haar classifiers for object detection with CUDA. In: GPU computing gems, Emerald Edition. Elsevier, pp 517–544
62. Pertsau D, Uvarov A (2013) Face detection algorithm using Haar-like feature for GPU architecture. In: IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)
63. Coates A, Baumstarck P, Le Q, Ng AY (2009) Scalable learning for object detection with GPU hardware. In: IEEE/RSJ International Conference on Intelligent Robots and Systems
64. Oro D, Fern'ndez C, Segura C, Martorell X, Hernando J (2012) Accelerating boosting-based face detection on GPUs. In: 41st International Conference on Parallel Processing
65. Herout A, Jořth R, Jur'nek R, Havel J, Hradř M, Zemřk P (2010) Real-time object detection on CUDA. *J Real-Time Image Proc* 6(3):159–170

66. Zhuang H, Low K-S, Yau W-Y (2012) Multichannel pulse-coupled-neural-network-based color image segmentation for object detection. *IEEE Trans Ind Electron* 59(8):3299–3308
67. Lozano OM, Otsuka K (2008) Simultaneous and fast 3D tracking of multiple faces in video by GPU-based stream processing. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, p 2008
68. Possa PR, Mahmoudi SA, Harb N, Valderrama C, Manneback P (2014) A multi-resolution FPGA-based architecture for real-time edge and corner detection. *IEEE Trans Comput* 63(10):2376–2388
69. Barbosa JPF, Ferreira APA, Rocha RCF, Albuquerque ES, Reis JR, Albuquerque DS, Barros ENS (2015) A high performance hardware accelerator for dynamic texture segmentation. *J Syst Archit* 61(10):639–645
70. Kryjak T, Komorkiewicz M, Gorgon M (2012) Real-time background generation and foreground object segmentation for high-definition colour video stream in FPGA device. *J Real-Time Image Proc* 9(1):61–77
71. Park J, Sung W (2016) FPGA based implementation of deep neural networks using on-chip memory only. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*
72. Zhao M, Hu C, Wei F, Wang K, Wang C, Jiang Y (2019) Real-time underwater image recognition with FPGA embedded system for convolutional neural network. *Sensors* 19(2):350
73. Zhang T, Zhou W, Jiang X, Liu Y (2018) FPGA-based implementation of hand gesture recognition using convolutional neural network. Presented at the 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)
74. Reyes E, Gómez C, Norambuena E, Ruiz-del-Solar J (2019) Near real-time object recognition for pepper based on deep neural networks running on a backpack. In: *RoboCup 2018: Robot World Cup XXII*. Springer, pp 287–298
75. Zhou Y, Wang W, Huang X (2015) FPGA design for PCANet deep learning network. In: *IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*
76. Hikawa H, Kaida K (2015) Novel FPGA implementation of hand sign recognition system with SOM-Hebb classifier. *IEEE Trans Circuits Syst Video Technol* 25(1):153–166
77. Svab J, Krajník T, Faigl J, Preucil L (2009) FPGA based speeded up robust features. Presented at the 2009 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)
78. Yao L, Feng H, Zhu Y, Jiang Z, Zhao D, Feng W (2009) An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher. In: *International Conference on Field-Programmable Technology*
79. Gu Q, Takaki T, Ishii I (2013) Fast FPGA-based multiobject feature extraction. *IEEE Trans Circuits Syst Video Technol* 23(1):30–45
80. Knag P, Kim JK, Chen T, Zhang Z (2015) A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding. *IEEE J Solid-State Circuits* 50(4):1070–1079
81. Bouris D, Nikitakis A, Papaefstathiou I (2010) Fast and efficient FPGA-based feature detection employing the SURF algorithm. Presented at the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines
82. Ali U, Malik MB, Munawar K (2009) FPGA/soft-processor based real-time object tracking system. In: *5th Southern Conference on Programmable Logic (SPL)*
83. Liu S, Papakonstantinou A, Wang H, Chen D (2011) Real-time object tracking system on FPGAs. Presented at the 2011 Symposium on Application Accelerators in High-Performance Computing (SAAHPC 2011)
84. Kryjak T, Gorgon M (2013) Real-time implementation of the ViBe foreground object segmentation algorithm. In: *Federated Conference on Computer Science and Information Systems (FedC-SIS)*, pp 591–596
85. Saqib F, Dutta A, Plusquellic J, Ortiz P, Pattichis MS (2015) Pipelined decision tree classification accelerator implementation in FPGA (DT-CAIF). *IEEE Trans Comput* 64(1):280–285
86. Pan J, Lauterbach C, Manocha D (2010) g-Planner: real-time motion planning and global navigation using GPUs. In: *Proceedings of AAAI Conference on Artificial Intelligence* 1245–1251
87. Vasumathi B, Moorthi S (2012) Implementation of hybrid ANN-PSO algorithm on FPGA for harmonic estimation. *Eng Appl Artif Intell* 25(3):476–483
88. Appleyard J, Kocisky T, Blunsom P (2016) Optimizing performance of recurrent neural networks on gpus. *arXiv preprint [arXiv:1604.01946](https://arxiv.org/abs/1604.01946)*

89. Wang Y, Xu J, Han Y, Li H, Li X (2016) DeepBurning: automatic generation of FPGA-based learning accelerators for the neural network family, pp 1–6
90. Sharma H, Park J, Amaro E, Thwaites B, Kotha P, Gupta A, Kim Joon K, Mishra A, Esmailzadeh H (2016) DNNWeaver: from high-level deep network models to FPGA acceleration. In: Workshop on Cognitive Architectures
91. DiCecco R, Lacey G, Vasiljevic J, Chow P, Taylor G, Areibi S (2016) Caffeinated FPGAs: FPGA framework for convolutional neural networks. In: International Conference on Field-Programmable Technology (FPT)
92. Umuroglu Y, Fraser NJ, Gambardella G, Blott M, Leong P, Jahre M, Vissers K (2017) FINN: a framework for fast, scalable binarized neural network inference. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays—FPGA '17
93. Geng T, Wang T, Sanaullah A, Yang C, Patel R, Herbordt M (2018) A framework for acceleration of CNN training on deeply-pipelined FPGA clusters with work and weight load balancing. Presented at the 2018 28th International Conference on Field Programmable Logic and Applications (FPL)
94. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia—MM '14
95. Venieris SI, Bouganis C-S (2016) FPAGConvNet: a framework for mapping convolutional neural networks on FPGAs. In: IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)
96. Samragh M, Ghasemzadeh M, Koushanfar F (2017) Customizing neural networks for efficient FPGA implementation. Presented at the 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)
97. Liu Z, Dou Y, Jiang J, Xu J, Li S, Zhou Y, Xu Y (2017) Throughput-optimized FPGA accelerator for deep convolutional neural networks. *ACM Trans Reconfig Technol Syst* 10(3):1–23
98. Guan Y, Liang H, Xu N, Wang W, Shi S, Chen X, Sun G, Zhang W, Cong J (2017) FP-DNN: an automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates. In: IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)
99. Wei X, Yu CH, Zhang P, Chen Y, Wang Y, Hu H, Cong J (2017) Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. Presented at the 54th Annual Design Automation Conference 2017
100. Zhao R, Ng H-C, Luk W, Niu X (2018) Towards efficient convolutional neural network for domain-specific applications on FPGA. In: 28th International Conference on Field Programmable Logic and Applications (FPL)
101. Bottleson J, Kim S, Andrews J, Bindu P, Murthy DN, Jin J (2016) clCaffe: OpenCL accelerated Caffe for convolutional neural networks. In: IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)
102. Rabbi S, Sun W, Perez J, Kristensen MRB, Liu J, Oldridge E (2019) Accelerating recommender system training 15x with RAPIDS. In: Proceedings of the Workshop on ACM Recommender Systems Challenge. RecSys Challenge '19: ACM Recommender Systems Challenge 2019 Workshop
103. Gong J, Shen H, Zhang G, Liu X, Li S, Jin G, Maheshwari N, Fomenko E, Segal E (2018) Highly efficient 8-bit low precision inference of convolutional neural networks with IntelCaffe. In: Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning (ReQuEST '18). Association for Computing Machinery, New York, NY, USA, Article 2, 1
104. Abdelouahab K, Pelcat M, Serot J, Bourrasset C, Berry F (2017) Tactics to directly map CNN graphs on embedded FPGAs. *IEEE Embed Syst Lett* 9(4):113–116
105. Sharma H et al (2016) From High-level deep neural models to FPGAs. In: 49th Annual IEEE/ACM International Symposium on Microarchitecture, pp 1–12
106. Ma Y, Suda N, Cao Y, Vrudhula S, Seo JS (2018) ALAMO: FPGA acceleration of deep learning algorithms with a modularized RTL compiler. *Integration* 62:14–23
107. Venieris SI (2017) Latency-driven design for FPGA-based convolutional neural networks
108. Zeng H, Zhang C, Prasanna V (2018) Fast generation of high throughput customized deep learning accelerators on FPGAs. In: International Conference on Reconfigurable Computing FPGAs, ReConFig 2017, vol 2018-Janua, pp 1–8

109. Venieris SI (2018) f-CNN x : a toolflow for mapping multiple convolutional neural networks on FPGAs
110. Ma Y, Cao Y, Vrudhula S, Seo JS (2020) Automatic compilation of diverse CNNs onto high-performance FPGA accelerators. *IEEE Trans Comput Des Integr Circuits Syst*. <https://doi.org/10.1109/TCAD.2018.2884972>
111. Ma Y, Suda N, Cao Y, Seo JS, Vrudhula S (2016) Scalable and modularized RTL compilation of convolutional neural networks onto FPGA. In: 26th International Conference on Field-Programmable Logic and Applications (FPL)
112. Cadambi S, Graf HP (2010) A programmable parallel accelerator for learning and classification, pp 273–283
113. Art P (2011) Artificial neural network acceleration on FPGA using custom instruction, pp 450–455
114. Luo G, Zhang C, Cong J, Sun J, Sun G, Wu D (2016) Energy-efficient CNN implementation on a deeply pipelined FPGA cluster, pp 326–331
115. Sun F et al (2018) A high-performance accelerator for large-scale convolutional neural networks. In: Proceedings of the 15th IEEE International Symposium on International Parallel and Distributed Processing with Application. 16th IEEE International Conference on Ubiquitous Computing and Communications, ISPA/IUCC 2017, pp 622–629
116. Qiao Y (2011) FPGA-accelerated deep convolutional neural networks for high throughput and energy efficiency. *Seismol Res Lett* 82(2):2010–2011
117. Motamedi M, Gysel P, Akella V, Ghiasi S (2016) Design space exploration of FPGA-based deep convolutional neural networks. In: Proceeding of Asia and South Pacific Design Automation Conference, ASP-DAC, vol 25–28 Jan, pp 575–580
118. Rahman A, Lee J, Choi K (2016) Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array, pp 1393–1398
119. Zhang J, Li J (2017) Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network, pp 25–34
120. Yonekawa H, Nakahara H (2017) On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an FPGA. In: Proceedings of IEEE 31st International Parallel and Distributed Processing Symposium Work, IPDPSW, pp 98–105
121. Nakahara H, Fujii T, Sato S (2017) A fully connected layer elimination for a binarized convolutional neural network on an FPGA. In: 27th International Conference on Field-Programmable Logic and Applications (FPL), pp 1–4
122. Kim L (2017) DeepX: deep learning accelerator for restricted Boltzmann machine artificial neural networks, pp 1–13
123. Zhao R et al (2017) Accelerating binarized convolutional neural networks with software-programmable FPGAs, pp 15–24
124. Aydonat U, O'Connell S, Capalija D, Ling AC, Chiu GR (2017) An OpenCL(TM) deep learning accelerator on Arria 10, pp 55–64
125. Shimoda M, Sato S, Nakahara H (2018) All binarized convolutional neural network and its implementation on an FPGA. In: International Conference on Field-Programmable Technology, ICFPT, vol 2018-Janua, pp 291–294
126. Xian A, Chang M, Culurciello E (2017) Hardware accelerators for recurrent neural networks on FPGA, pp 0–3
127. Guo J, Yin S, Ouyang P, Liu L, Wei S (2017) Bit-width based resource partitioning for CNN acceleration on FPGA. In: Proceedings of IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines. FCCM 2017, p 31
128. Zhang C, Prasanna V (2017) Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system, pp 35–44
129. Yan S, Lu L, Liang Y, Xiao Q, Tai Y-W (2017) Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs, pp 1–6
130. Gong L, Wang C, Li X, Chen X, Zhou X (2017) Work-in-progress: a power-efficient and high performance FPGA accelerator for convolutional neural networks
131. Ma Y, Cao Y, Vrudhula S, Seo J (2017) Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks, pp 45–54
132. Nguyen D, Kim D, Lee J (2017) Double MAC: doubling the performance of convolutional neural networks on modern FPGAs. In: Proceedings of 2017 Design, Automation and Test in Europe Conference and Exhibition, pp 890–893

133. Hwang WJ, Jhang YJ, Tai TM (2017) An efficient FPGA-based architecture for convolutional neural networks. In: 40th International Conference on Telecommunications and Signal Processing, TSP, vol 2017-Janua, pp 582–588
134. Ma Y, Cao Y, Vrudhula S, Seo JS (2018) Optimizing the convolution operation to accelerate deep neural networks on FPGA. *IEEE Trans Very Large Scale Integr Syst* 26(7):1354–1367
135. Guan Y, Yuan Z, Sun G, Cong J (2017) FPGA-based accelerator for long short-term memory recurrent neural networks. In: *Proceedings of Asia and South Pacific Design Automation Conference*, ASP-DAC, pp 629–634
136. Ma Y, Kim M, Cao Y, Vrudhula S, Seo JS (2017) End-to-end scalable FPGA accelerator for deep residual networks. In: *Proceedings of IEEE International Symposium on Circuits and Systems*, pp 0–3
137. Yu J et al (2018) Instruction driven cross-layer CNN accelerator with winograd transformation on FPGA. In: *International Conference on Field-Programmable Technology*, ICFPT 2017, vol 2018-Janua, pp 227–230
138. Kim JH, Grady B, Lian B, Brothers J, Anderson JH (2017) FPGA-based CNN inference accelerator synthesized from multi-threaded C software, pp 268–273
139. Moss DJM et al (2017) High performance binary neural networks on the Xeon+FPGATM platform. In: *27th International Conference on Field-Programmable Logic and Applications (FPL)*
140. Guo K et al (2018) Angel-Eye: a complete design flow for mapping CNN onto embedded FPGA. *IEEE Trans Comput Des Integr Circuits Syst* 37(1):35–47
141. Gong L, Wang C, Li X, Chen H, Zhou X (2018) MALOC: a fully pipelined FPGA accelerator for convolutional neural networks with all layers mapped on chip. *IEEE Trans Comput Des Integr Circuits Syst* 37(11):2601–2612
142. Duarte RP (2018) Lite-CNN: a high-performance architecture to execute CNNs in low density FPGAs
143. Rybalkin V, Pappalardo A, Ghaffar MM, Gambardella G, Wehn N, Blott M (2018) FINN-L: Library extensions and design trade-off analysis for variable precision LSTM networks on FPGAs. In: *Proceedings of 2018 International Conference on Field-Programmable Logic and Applications (FPL)*, pp 89–96
144. Yu Q, Wang C, Ma X, Li X, Zhou X, (2015) A deep learning prediction process accelerator based FPGA. In: *Proceedings of 2015 IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid 2015, no 500, pp 1159–1162
145. Abdelfattah MS et al (2018) DLA: compiler and FPGA overlay for neural network inference acceleration
146. Nurvitadhi E et al (2018) In-package domain-specific ASICs for Intel® Stratix® 10 FPGAs: a case study of accelerating deep learning using TensorTile ASIC, pp 106–110
147. Zhang C (2015) Optimizing FPGA-based accelerator design for deep convolutional neural networks, pp 161–170
148. Qiu J et al (2016) Going deeper with embedded FPGA platform for convolutional neural network, pp 26–35
149. Vrudhula S et al (2016) Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks, pp 16–25
150. Wang Y et al (2016) Low power convolutional neural networks on a chip. In: *Proceedings of IEEE International Symposium on Circuits and Systems*, vol 2016-July, no 1, pp 129–132
151. Feng G, Hu Z, Chen S, Wu F (2016) Energy-efficient and high-throughput FPGA-based accelerator for convolutional neural networks, pp 4–6
152. Wang C, Gong L, Yu Q, Li X, Xie Y, Zhou X (2017) DLAU: a scalable deep learning accelerator unit on FPGA. *IEEE Trans Comput Des Integr Circuits Syst* 36(3):513–517
153. Park J, Lotfi-Kamran P, Sharma H, Esmailzadeh H, Yazdanbakhsh A (2016) Neural acceleration for GPU throughput processors, pp 482–493
154. Strigl D, Kofler K, Podlipnig S (2010) Performance and scalability of GPU-based convolutional neural networks. In: *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, PDP 2010, pp 317–324
155. Guzhva A, Dolenko S, Persiantsev I (2009) Multifold acceleration of neural network computations using GPU. In: *Artificial Neural Networks—ICANN 2009*, pp 373–380
156. Li B, Zhou E, Huang B, Duan J, Wang Y, Xu N, Zhang J, Yang H (2014) Large scale recurrent neural network on GPU. In: *International Joint Conference on Neural Networks (IJCNN)*

157. Kim Y, Lee J, Kim J-S, Jei H, Roh H (2018) Efficient multi-GPU memory management for deep learning acceleration. In: IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)
158. Bhuiyan MA, Pallipuram VK, Smith MC (2010) Acceleration of spiking neural networks in emerging multi-core and GPU architectures. In: IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW)
159. Zhang X, Gu N, Ye H (2016) Multi-GPU based recurrent neural networks language model training. In: Communications in computer and information science, pp 484–493
160. Potluri S, Fasih A, Vutukuru LK, Machot FA, Kyamakya K (2011) CNN based high performance computing for real time image processing on GPU. Presented at the 16th Int'l Symposium on Theoretical Electrical Engineering (ISTET)
161. Farah NICLA (2014) A new classification approach for neural networks hardware: from standards chips to embedded systems on chip, pp 491–534
162. Jin L, Wang Z, Gu R, Yuan C, Huang Y (2014) Training large scale deep neural networks on the Intel Xeon Phi many-core coprocessor. In: IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)
163. Kurth T, Zhang J, Satish N, Racah E, Mitliagkas I, Patwary MMA, Malas T, Sundaram N, Bhimji W, Smorkalov M et al (2017) Deep learning at 15PF: supervised and semi-supervised classification for scientific data. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, pp 7
164. Georganas E, Avancha S, Banerjee K, Kalamkar D, Henry G, Pabst H, Heinecke A (2018) Anatomy of high-performance deep learning convolutions on SIMD architectures. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC '18, Piscataway, NJ, USA. IEEE Press, pp 66:1–66:12
165. Viebke A, Memeti S, Pillana S, Abraham A (2017) CHAOS: a parallelization scheme for training convolutional neural networks on Intel Xeon Phi. *J Supercomput* 75(1):197–227
166. Mathuriya A, Bard D, Mendygral P, Meadows L, Arnemann J, Shao L, He S, Karna T, Moise D, Pennycook SJ, Maschhoff K, Sewall J, Kumar N, Ho S, Ringenburt MF, Prabhat P, Lee V (2018) CosmoFlow: using deep learning to learn the universe at scale. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis
167. Hu Y, Zhai J, Li D, Gong Y, Zhu Y, Liu W, Su L, Jin J (2018) BitFlow: exploiting vector parallelism for binary neural networks on CPU. Presented at the 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)
168. “Virtex-5”, Xilinx.com (2019). <https://www.xilinx.com/products/boards-and-kits/device-family/nav-virtex-5.html>. Accessed 16 Oct 2019
169. “Stratix V GX FPGA Development Kit”, Intel.com (2019). <https://intel.ly/31pCBMI>. Accessed 16 Oct 2019
170. “Arria 10 GX FPGA Development Kit”, Intel.com (2019). <https://intel.ly/2ITEPW0>. Accessed 16 Oct 2019
171. Chen Y et al (2014) DaDianNao: a machine-learning supercomputer
172. Amazon.com (2019). <https://www.amazon.com/NVIDIA-Computing-Processor-Graphic-900-22081-2250-000/dp/B00KDRRTB8>. Accessed: 16 Oct 2019
173. Amazon.com (2019). <https://www.amazon.com/Nvidia-TESLA-Accelerator-Processing-900-2G600-0000-000/dp/B01MDNO5BK>. Accessed 16 Oct 2019
174. “NVIDIA GeForce GT 730 Review”, Benchmarks.ul.com (2019). <https://benchmarks.ul.com/hardware/gpu/NVIDIA+GeForce+GT+730+review>. Accessed 16 Oct 2019