



**IMPROVED FIELD PROGRAMMABLE GATE ARRAYBASED
ACCELERATOR OF DEEP NEURAL NETWORK USING
OPENCL**



MASTER OF SCIENCE IN ELECTRONIC ENGINEERING

2022



Faculty of Electronics and Computer Engineering

A faded version of the UTeM logo and university name is visible in the background behind the title text.

IMPROVED FIELD PROGRAMMABLE GATEARRAYBASED ACCELERATOR OF DEEP NEURAL NETWORK USING OPENCL

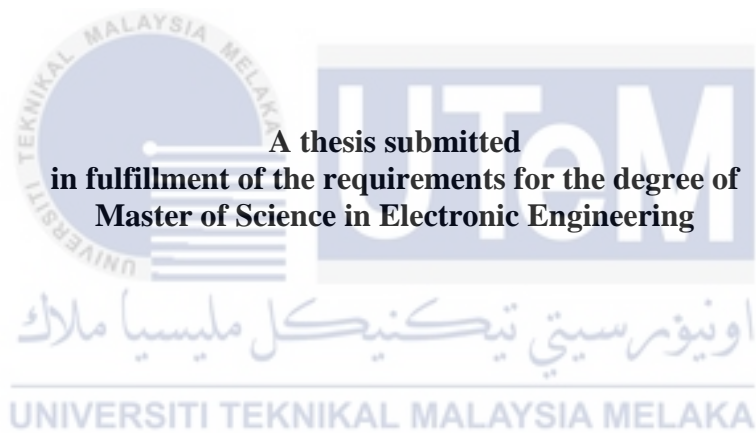
Yap June Wai

Master of Science in Electronic Engineering

2022

**IMPROVED FIELD PROGRAMMABLE GATEARRAYBASED ACCELERATOR
OF DEEP NEURAL NETWORK USING OPENCL**

YAP JUNE WAI



Faculty of Electronics and Computer Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2022

DECLARATION

I declare that this thesis entitled “Improved Field Programmable Gate Array (FPGA)Based Accelerator of Deep Neural Network Using OpenCL” is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature

:



Name

:

YAP JUNE WAI

Date

:

15 AUGUST 2022



APPROVAL

hereby declare that I have read this thesis and in my opinion, this thesis is sufficient in terms of scope and quality for the award of the degree of Master of Science in Electronic Engineering.

Signature

:



Supervisor Name

:

Professor Dr Zulkalnain bin Mohd Yussof

Date

:

25 AUGUST 2022



DEDICATION

Specially dedicated to my beloved family



ABSTRACT

Being compute-intensive and memory expensive, it is hard to deploy Deep Neural Network (DNN) based models into the embedded devices. Despite recent studies that have shown the efforts to explore the Field Programmable Gate Array (FPGA) as an alternative to deploy DNN-based models such as AlexNet and VGG, there is still a lot of challenges to implement DNN-based object detection model on Field Programmable Gate Array (FPGA). Hence, in this research, the design of a scalable parameterised DNN-based object detection model: Tiny YOLOv2 targeting on FPGA: Cyclone V PCIE Development Kit using High-Level-Synthesis (HLS) tool is explored. Considering the hardware resource limitations in term of computational resources and memory bandwidth, data quantization is proposed to convert the floating point (32-bit) of Tiny YOLOv2 into fixed-point (8-bit) design. To achieve the good performance, an in-depth analysis on the computation complexity and memory footprint of the Tiny YOLOv2 is also studied to find the best quantization scheme for Tiny YOLOv2. The proposed quantization scheme improves the memory requirements to store the parameter from 60 MB to 15 MB, which is around $\times 4$ times improvement compared to the original floating-point design. Finally, the proposed implementation achieves a peak performance density of 0.29 Giga-Operation Per Second (GOPS)/Digital Signal Processing Block (DSP) with only 0.4% loss in the accuracy, which the performance is comparable to all other previous works.

PENAMBAHBAIKAN PEMECUT BERASASKAN TATASUSUNAN BOLEH ATUR GET MEDAN UNTUK JARINGAN NEURAL DALAM MENGGUNAKAN OPENCL

ABSTRAK

Pengiraan dan penggunaan ingatan yang intensif telah memberi cabaran untuk pelaksanaan algoritma Rangkaian Neural Dalam (DNN) dalam sistem terbenam. Walaupun kajian baru-baru ini telah menunjukkan usaha untuk menerokai Tatasusunan Logik Boleh Aturcara (FPGA) sebagai alternatif untuk melaksanakan pengklasifikasian objek seperti “*AlexNet*” dan “*VGG*”, namun masih terdapat cabaran untuk melaksanakan algoritma pengesanan objek berasaskan rangkaian neural dalam dengan menggunakan tatasusunan logik boleh aturcara. Oleh itu, dalam penyelidikan ini, pengesanan objek: “*Tiny YOLOv2*” yang dapat diskalakan ke atas tatasusunan logik boleh aturcara bernama “*Cyclone V PCIE Development Kit*” dengan menggunakan peralatan Tahap Tinggi-Sintesis (HLS) akan dieksploitasikan. Memandangkan sumber yang berhad dari segi sumber pengiraan dan kapasiti memori, pengkuantuman data telah dicadangkan untuk menukar titik terapung ke titik tetap. Demi mencapai pretasi yang baik untuk skim pengkuantuman, analisis terhadap kerumitan pengiraan dan memori *Tiny YOLOv2* telah dikajikan. Skim pengkuantuman yang dicadangkan dapat mengurangkan keperluan memori untuk menyimpan data sebanyak 60 MB sehingga 15 MB, iaitu kira-kira $\times 4$ kali peningkatan berbanding dengan aritmetika titik mengambang. Akhir sekali, kajian yang dicadangkan mencapai 0.29 GOPS / DSP dengan hanya 0.4% pengorbanan dalam pretasi ketepatan pengesanan objek. Pretasi yang dicapai dapat dibandingkan dengan semua karya sebelumnya

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Professor Dr Zulkalnain bin Mohd Yussof from Faculty of Electronic and Computer Engineering (FKEKK), Universiti Teknikal Malaysia Melaka (UTeM) for his guidance and assistance throughout the completion of this thesis. I would also like to express my sincere acknowledgement to my co-supervisor, Dr Sani Irwan bin Salim from Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka (UTeM) for his advice throughout the project. It is a genuine pleasure to express my deep sense of appreciation to UTeM Zamalah Scheme for the financial support throughout this project

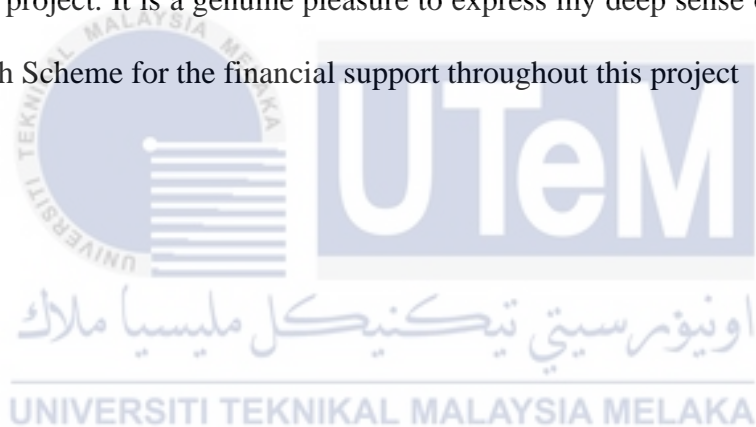


TABLE OF CONTENTS

| | PAGE |
|--|-------------|
| DECLARATION | |
| APPROVAL | |
| DEDICATION | |
| ABSTRACT | i |
| ABSTRAK | ii |
| ACKNOWLEDGEMENTS | iii |
| TABLE OF CONTENTS | iv |
| LIST OF TABLES | vii |
| LIST OF FIGURES | viii |
| LIST OF SYMBOLS AND ABBREVIATIONS | xii |
| LIST OF APPENDICES | xv |
| LIST OF PUBLICATIONS | xvi |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Background | 1 |
| 1.2 Problem Statement | 3 |
| 1.3 Research Question | 4 |
| 1.4 Hypothesis | 5 |
| 1.5 Research Objective | 5 |
| 1.6 Research Scope | 5 |
| 1.7 Contribution of Research | 6 |
| 1.8 Thesis Outline | 7 |
| CHAPTER 2 LITERATURE REVIEW | 9 |
| 2.1 Machine Learning | 9 |
| 2.2 Neural Network | 9 |
| 2.3 Deep Convolutional Neural Network | 11 |
| 2.4 Common Layers used to Build Deep Convolutional Neural Networks | 12 |
| 2.4.1 Convolutional Layer | 13 |
| 2.4.2 Pooling Layer | 14 |
| 2.4.3 Activation Layer | 16 |
| 2.5 Common Architecture in Deep Neural Networks | 18 |
| 2.5.1 Deep Neural Networks Image Classification | 19 |

| | | |
|---|---|-----------|
| 2.5.1.1 | AlexNet | 20 |
| 2.5.1.2 | Visual Geometry Group (VGG) | 20 |
| 2.5.2 | Deep Neural Networks Object Detection | 21 |
| 2.5.2.1 | Region-Based Convolutional Neural Network (R-CNN) | 22 |
| 2.5.2.2 | Fast Region Based Convolutional Neural Network (Fast RCNN) | 23 |
| 2.5.2.3 | Faster Region Based Convolutional Neural Network (Faster RCNN) | 24 |
| 2.5.2.4 | You Only Look Once (YOLO) | 24 |
| 2.5.2.5 | You Only Look Once version 2 (YOLOv2) | 26 |
| 2.5.3 | Summary of Modern Deep Neural Networks (DNNs) | 26 |
| 2.6 | Field Programmable Gate Array (FPGA) | 27 |
| 2.7 | Board Selection Study | 28 |
| 2.8 | High-Level Synthesis (HLS) | 30 |
| 2.8.1 | Open Computing Language (OpenCL) | 30 |
| 2.8.2 | Intel Field Programmable Gate Array Software Development Kit for OpenCL | 35 |
| 2.9 | Data Quantization | 36 |
| 2.10 | Related Work on Field Programmable Gate Array | 39 |
| CHAPTER 3 METHODOLOGY | | 44 |
| 3.1 | Project Implementation Description | 44 |
| 3.2 | Experimental Setup | 44 |
| 3.2.1 | Board Specification | 45 |
| 3.2.2 | Hardware and Software Installation | 46 |
| 3.3 | Design and Development | 51 |
| 3.3.1 | Design Flow Using Intel FPGA SDK for OpenCL | 51 |
| 3.3.2 | Overall System Design Overview | 54 |
| 3.3.3 | Tiny YOLOv2 Architecture | 57 |
| 3.3.4 | Analysis of Computational Complexity and Space Complexity | 59 |
| 3.3.5 | Optimization on Convolution Operation | 61 |
| 3.3.5.1 | General Matrix to Matrix Multiplication (GeMM) Based Convolution | 61 |
| 3.3.5.2 | Data Rearrangement On-The-Fly | 62 |
| 3.3.5.3 | Tiled Matrix Multiplication | 63 |
| 3.3.6 | Optimization on Hardware Synthesis using OpenCL Specification | 64 |
| 3.3.6.1 | Loop Unrolling | 65 |
| 3.3.6.2 | Kernel Vectorization | 65 |
| 3.3.6.3 | Local Memory Caching | 66 |
| 3.3.7 | Optimization on Hardware Resources | 67 |
| 3.3.7.1 | Merging Batch Normalization into Convolution | 67 |
| 3.3.7.2 | Data Quantization | 70 |
| 3.4 | Performance Evaluation | 75 |
| 3.4.1 | The Profiling Stage for Identifying the Exact Resources Consumption | 75 |
| 3.4.2 | Mean Average Precision (mAP) | 76 |

| | | |
|-------------------|---|------------|
| 3.4.3 | Performance Density | 79 |
| CHAPTER 4 | RESULTS AND DISCUSSION | 81 |
| 4.1 | In-Depth Analysis on the Computation Complexity and Space Complexity | 81 |
| 4.2 | In-Depth Analysis on the Biases and Weights | 85 |
| 4.3 | The Hardware Resource Utilization and Performance | 93 |
| 4.4 | Evaluation on the Accuracy of the Proposed Accelerator | 98 |
| 4.5 | Comparison of the Performance between Software Implementation and Hardware Implementation | 105 |
| 4.6 | Performance Comparison with Previous Research with FPGA-based Design | 107 |
| CHAPTER 5 | CONCLUSION AND RECOMMENDATIONS | 109 |
| 5.1 | Conclusion | 109 |
| 5.2 | Future Work and Improvements | 111 |
| REFERENCES | | 113 |
| APPENDICES | | 121 |



LIST OF TABLES

| TABLE | TITLE | PAGE |
|--------------|--|-------------|
| Table 2.1 | The Summary of Modern Deep Neural Networks | 27 |
| Table 2.2 | The Overview of Four Low-End FPGAs in Market | 29 |
| Table 2.3 | The Overview of High-Level Synthesis Tools | 30 |
| Table 2.4 | The Summary of Resources in Context | 33 |
| Table 2.5 | The Summary of Previous Related Works | 43 |
| Table 3.1 | The Specification of Cyclone V PCIE Development Kit | 46 |
| Table 3.2 | The Summary of Specification Used in This Work | 48 |
| Table 3.3 | The System Environment Variable Configuration | 48 |
| Table 3.4 | The Configuration of Tiny YOLOv2 on Pascal VOC 2007 Dataset | 58 |
| Table 4.1 | The Precision Study for the Parameters of Tiny YOLOv2 in 8-bit fixed point | 90 |
| Table 4.2 | The Summary of Per Class Average Precision between the Original Tiny YOLOv2 and Proposed Work | 104 |
| Table 4.3 | The Summary of the Comparison of the Performance between Software Implementation and FPGA Implementation | 106 |
| Table 4.4 | The Summary of Comparison with Previous Works on FPGA | 108 |

LIST OF FIGURES

| FIGURE | TITLE | PAGE |
|-------------|--|------|
| Figure 2.1 | The Perceptron Model (Haykin et al., 1994) | 10 |
| Figure 2.2 | Simple Neural Network | 11 |
| Figure 2.3 | The Overview of Architecture of Deep Neural Network (LeCun et al., 1998) | 12 |
| Figure 2.4 | Architecture of LeNet-5, a Convolutional Neural Network for digit recognition(LeCun et al., 1998). | 13 |
| Figure 2.5 | The Overview of Convolutional Layer (Zhang et al., 2015) | 13 |
| Figure 2.6 | Example of Max pool with a 2×2 filter and a stride of 2 | 15 |
| Figure 2.7 | Example Average Pool with a 2×2 filter and a stride of 2 | 16 |
| Figure 2.8 | Sigmoid Function (Han et al., 1995) | 17 |
| Figure 2.9 | Hyperbolic Tangent (Zamanlooy et al., 2013) | 17 |
| Figure 2.10 | Rectified Linear Unit (Krizhevsky et al., 2012) | 18 |
| Figure 2.11 | Leaky Rectified Linear Unit (Krizhevsky et al., 2012) | 18 |
| Figure 2.12 | The Difference between classification (a) and object detection (b) | 19 |
| Figure 2.13 | The Architecture of AlexNet (Krizhevsky, 2012) | 20 |
| Figure 2.14 | The Architecture of VGG-16 (Simonyan et al., 2014) | 21 |
| Figure 2.15 | The Overview of Region-based CNN (R. Girshick et al., 2014) | 23 |
| Figure 2.16 | The Overview of Fast Region-based CNN (R. Girshick et al., 2015) | 23 |
| Figure 2.17 | The Overview of Faster RCNN (R. Girshick et al., 2015) | 24 |
| Figure 2.18 | The Overview of You Only Look Once (YOLO) (Redmon et al., 2016) | 25 |
| Figure 2.19 | The Architecture of You Only Look Once (YOLO) (Redmon et al., 2016) | 26 |

| | | |
|-------------|---|----|
| Figure 2.20 | The Architecture of Intel Arria 10 FPGA (Intel Altera, 2016) | 28 |
| Figure 2.21 | OpenCL Platform Model (Munshi et al, 2011) | 31 |
| Figure 2.22 | OpenCL Execution Model | 32 |
| Figure 2.23 | OpenCL Memory Model (Munshi et al, 2011) | 34 |
| Figure 2.24 | The OpenCL Platform Model for FPGAs (Intel, 2017) | 36 |
| Figure 2.25 | The Overview of Data Quantization Techniques | 37 |
| Figure 2.26 | The Architecture of PipeCNN | 41 |
| Figure 2.27 | The Comparison of Data Pipelining (a) and Without Pipelining (b) | 41 |
| Figure 2.28 | The Decomposition of Matrix Multiplication, Singular Value Decomposition | 42 |
| Figure 3.1 | The Summary of Workflow Progress | 44 |
| Figure 3.2 | The Layout Diagram of Cyclone V PCIE Development Kit | 45 |
| Figure 3.3 | Hardware and Software Installation Workflow | 47 |
| Figure 3.4 | Device Manager of the Workstation | 49 |
| Figure 3.5 | The Board Support Package of Cyclone V PCIE Development Kit | 50 |
| Figure 3.6 | Design and Development Flow of OpenCL | 53 |
| Figure 3.7 | The Overall Top-Down System Design Overview | 55 |
| Figure 3.8 | Memory Architecture of The Proposed Accelerator | 56 |
| Figure 3.9 | The Architecture of Tiny YOLOv2 with Pascal VOC 2007 dataset | 57 |
| Figure 3.10 | Mapping 3D Convolution to GeMM Based Convolution | 62 |
| Figure 3.11 | Pseudo Code for Data Rearrangement on-the-fly | 63 |
| Figure 3.12 | Tiled Matrix Multiplication | 64 |
| Figure 3.13 | Kernel Vectorization | 66 |
| Figure 3.14 | The Workflow of Designing a Quantizer | 72 |
| Figure 3.15 | The Proposed Quantization Scheme | 74 |

| | | |
|-------------|---|----|
| Figure 3.16 | The workflow of Profiling Process | 76 |
| Figure 3.17 | The Confusion Matrix | 77 |
| Figure 3.18 | Intersection of Union (IoU) | 78 |
| Figure 4.1 | The Computation Complexity and Space Complexity of Tiny YOLOv2 running on Pascal VOC 2007 Datasets | 82 |
| Figure 4.2 | The Segmentation of Operations in Tiny YOLOv2 | 83 |
| Figure 4.3 | The Segmentation of Parameters in Tiny YOLOv2 | 83 |
| Figure 4.4 | Segmentation of Number of Operations in Each Layer of Tiny YOLOv2 | 85 |
| Figure 4.5 | Segmentation of Number of Parameters in Each Layer of Tiny YOLOv2 | 85 |
| Figure 4.6 | The Data Distribution of Bias in Each Layer of Tiny YOLOv2 | 87 |
| Figure 4.7 | The Data Distribution of the Weights in Each Layer of Tiny YOLOv2 | 88 |
| Figure 4.8 | The Visualised SNQR Test Using MATLAB on the Weights Convolutional Layer No 5 | 89 |
| Figure 4.9 | The Data Distribution of Quantized Bias in Each Layer of Tiny YOLOv2 | 91 |
| Figure 4.10 | The Data Distribution of Quantized Weight in Each Layer of Tiny YOLOv2 | 92 |
| Figure 4.11 | ALUTs Utilization on Cyclone V PCIE Development Kit with Different Configurations of <i>BLOCK_SIZE</i> and <i>SIMD</i> | 94 |
| Figure 4.12 | FFs Utilization on Cyclone V PCIE Development Kit with Different Configurations of <i>BLOCK_SIZE</i> and <i>SIMD</i> | 94 |
| Figure 4.13 | RAMs Utilization on Cyclone V PCIE Development Kit with Different Configurations of <i>BLOCK_SIZE</i> and <i>SIMD</i> | 95 |
| Figure 4.14 | DSPs Utilization on Cyclone V PCIE Development Kit with Different Configurations of <i>BLOCK_SIZE</i> and <i>SIMD</i> | 95 |

| | | |
|-------------|---|-----|
| Figure 4.15 | Throughput Achieved on Cyclone V PCIE Development Kit with Different Configurations of BLOCK_SIZE and SIMD | 98 |
| Figure 4.16 | The Comparison of Detection Results Between Ground Truth, Original TinyYOLOv2 and Proposed Fixed-Point Design Using Pascal VOC 2007 Test Datasets | 100 |
| Figure 4.17 | The Comparison of Detection Results between Floating Point Design and Fixed Point Design Using Random Images | 101 |
| Figure 4.18 | The Wrong Samples of Ground Truth, Original Tiny YOLOv2 and Proposed Fixed-Point Design Using Pascal VOC 2007 Test Datasets | 102 |
| Figure 4.19 | The Comparison of Proposed Accelerator on FPGA with Software Implementation | 107 |



LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|------|--------------------------------------|
| 2D | 2 Dimensional |
| 3D | 3 Dimensional |
| AI | Artificial Intelligence |
| ALM | Adaptive Logic Module |
| ALUT | Adaptive Look Up Table |
| AP | Average Precision |
| API | Application Programming Interface |
| BN | Batch Normalization |
| BSP | Board Support Package |
| CNN | Convolutional Neural Network |
| COCO | Common Object in Context |
| CPU | Central Processing Unit |
| CU | Compute-Unit |
| CV | Computer Vision |
| DL | Deep Learning |
| DNN | Deep Neural Network |
| DSP | Digital Signal Processing |
| FF | Flip-Flop |
| FPGA | Field Programmable Gate Array |
| GEMM | General Matrix-Matrix Multiplication |

| | |
|--------|---|
| GFLOPS | Giga Floating Point Operations Per Second |
| GMACS | Giga Multiplication-Accumulates per Second |
| GOPS | Giga Operations Per Second |
| GPIO | General Purpose Input Output |
| GPU | Graphic Processing Unit |
| GUI | Graphic User Interface |
| HDL | Hardware Description Language |
| HLS | High Level Synthesis |
| ILSVRC | ImageNet Large Scale Visual Recognition Challenge |
| IoT | Internet of Things |
| IOU | Intersection Over Union |
| MAP | Mean Average Precision |
| ML | Machine Learning |
| NN | Neural Network |
| OpenCL | Open Computing Language |
| PE | Processing Element |
| RAM | Random Memory Access |
| RCNN | Region Based Convolutional Neural Network |
| ReLU | Rectifier Linear Unit |
| ROI | Region of Interest |
| RTL | Register-Transfer Level |
| SDK | Software Development Kit |
| SIMD | Single Instruction Multiple Data |



| | |
|------|------------------------------|
| SVD | Singular Value Decomposition |
| SVM | Support Vector Machine |
| VOC | Visual Object Classification |
| YOLO | You Only Look Once |



LIST OF APPENDICES

| APPENDIX | TITLE | PAGE |
|------------|---|------|
| APPENDIX A | AMATLAB Analysis on the Weights and Biases of Tiny Yolo-v2 Layers | 121 |
| APPENDIX B | Maximum Value Analysis on Tiny-Yolo-v2 layers | 130 |



LIST OF PUBLICATIONS

Indexed Journal

Yap, J.W., Yussof, Z.M., Salim, S.I. and Lim, K.C., 2018. Fixed Point Implementation of Tiny-Yolo—v2 using OpenCL on FPGA. International Journal of Advanced Computer Science and Applications (IJACSA), 9(10), pp.62.

Yap, J.W., Yussof, Z.M., and Salim, S.I., 2019. A Scalable FPGA based Accelerator for Tiny-YOLO-v2 using OpenCL. International Journal of Reconfigurable and Embedding System (IJRES), 8(10), pp.206.



CHAPTER 1

INTRODUCTION

1.1 Background

The rise of the new digital industrial revolution, also known as Industry 4.0, is a significant transformation on the road to an end-to-end value chain with Industrial IoT and decentralized artificial intelligent (AI) in manufacturing, production, and logistic. It is undoubtedly that the AI will be the highlight and the key to propelling the rise of the new era. Deep learning, a subfield in AI family, which recently acts as a catalyst to the growing of AI, is inspired by the artificial neural network algorithm. The deep learning is widely adopted in various applications, which include video surveillance (Xu et al., 2015; Liu et al., 2016; Bashbaghi et al., 2018; Shorfuzzaman et al., 2020; Perez et al., 2021), autonomous vehicle (Bojarski et al., 2016; Sallab et al., 2017; Tian et al., 2018, Kuuti et al., 2020), mobile robot vision (Zhu et al., 2017; Levine et al., 2018, Chen et al., 2020). The deep learning led to early success in 1986, notably a neural network classification model named as LENET which is able to recognize the handwritten digits. Unfortunately, the neural network did not catch the attention to solve large scale problem. It is mainly due to the limitations in data availability and computing power in early 1990. With a large database which is known as ImageNet containing millions of labelled images was created in 2010, which led to the biggest breakthrough in Artificial Intelligence (AI) history. With the rapid growth of computer processing power and larger database, the first large scale deep neural network (DNN), namely AlexNet was introduced in 2012. AlexNet achieved a top-5 error of 15.3%, more than 10.8% error rate lower than the runner up in the ImageNet Large Scale Visual

Recognition Challenge (ILSVRC). The results significantly outperformed all the prior competitors and any other traditional classification approach. This led to the rapid evolution of Deep Neural Network (DNN) model, consequently the emerging of DNN-based classification model such as VGG16 and the emerging of DNN- based object detection models such as Recurrent Neural Network (RNN), Region-Based Convolutional Neural Network (RCNN), and You-Only-Look-Once (YOLO). The accuracy of these models has increased tremendously since 2012, with an increase in the complexity and number of layers in the network. The increase in the model size of deep learning model greatly increases the number of parameters, consequently leading to a significant increase in the computational requirements, memory bandwidth and storage required to store the parameters. For example, AlexNet requires around 244 MB of parameters and over 1.4 billion of operations to perform classification on a single input image, while VGG16 requires around 500 MB of parameters and over 30 billion of operations to perform classification on a single input image.

This gives a challenge to the deployment of DNN model in consumer devices that use embedded, such as household appliances, mobile phones, digital camera, and so on. Compared to stationary workstation such as desktop, embedded systems are more limited in terms of computational power, memory bandwidth, and power consumption. Hence, recent researches have explored the hardware implementation of deep learning models as an alternative to accelerate the DNN model. Field Gate Programmable Array (FPGA) can provide a relatively high performance, more energy-efficient, flexibility and fast development cycle especially with the introduction of High-Level synthesis (HLS) tool such as Open Computing Language (OpenCL), enabling the auto-compilation from high- level programming such as C/C++ to Hardware Description Language (HDL) such as Verilog or VHDL. It greatly reduces the effort on hardware development where the development