Power Analysis of 8-bit and Binary Neural Networks for Image Classification on FPGAs

A thesis submitted in partial fulfilment of the
requirements for the award of the degree

Bachelor of Engineering (Computing)

from

University of Wollongong

by

Julien F. Hill

School of Electrical, Computer and Telecommunications Engineering

June, 2023

Supervisor: Associate Professor Montse Ros

Abstract

When selecting the type of hardware (HW) to run neural networks (NN) for image classification the traditional choice has been between slow CPUs or power intensive GPUs. The traditional choice has been disrupted in recent years as new tools have made it cost and time efficient to develop NN onto Field Programmable Gate Arrays (FPGA). FPGAs provide a hardware acceleration option that allows for fast power efficient implementations of NNs. High level abstraction provided by tools like FINN reduce the amount of coding required to get a NN operational on a FPGA from months to days.

Whilst FINN provides a method to convert networks onto FPGA hardware much remains to be learnt about what kinds of networks work most effectively on a FPGA platform. When deploying NN to FPGAs, floating point weights and activations need to be quantised down to smaller bit widths due to the limited amount of HW resources available to the FPGA. Commonly, 8-bit quantisation is used however a more drastic 1-bit variant has been presented in works like FINN. 1-bit networks, also known as Binary Neural Networks (BNNs), allow for both the benefits of lower bit widths and binary operations.

This work demonstrates that BNNs provide a more power efficient implementation when working on the FPGA. In total, two data sets each with three NNs are tested in the work. On the MNIST data set, BNNs are on average 2.45 times more power efficient than the 8-bit reference. Along with BNNs, 2-bit (trinary) networks are tested. The inclusion of the trinary network allows for a comparison on whether the binary operations provide additional power savings. The trinary network provided 2.59 times more power efficiency than the reference 8-bit. Demonstrating that the main benefit of BNNs comes from the quantisation of weights and not the binary operations. An attempt was made to run CIFAR-10 networks however they were unable to produce reliable results. Being impeded by large reductions in their accuracy when deployed to the FPGA.

Statement of Originality

I, Julien F. Hill, declare that this thesis, submitted as part of the requirements for the award of Bachelor of Engineering, in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications or assessment at any other academic institution.

As author of this thesis, I also hereby grant, subject to any prior confidentially agreements, SECTE permission, to use, distribute, publish, exhibit, record, digitize broadcast, reproduce and archive this work for the purposes of further research and teaching.

(strike out that which does not apply)

This thesis;

~~IS~~

IS NOT

subject to a prior confidentially agreement.

Signature: Julien

Print Name: JULIEN HILL

Student ID Number: 6800506

Date: 15 - Oct. - 2023

Contents

List of Tables

List of Figures

Abbreviations and Symbols

$ASIC$	Application Specific Integrated Circuit
BNN	Binarized Neural Network
CLB	Configurable Logic Block
CNN	Convolution Neural Network
$Conv$	Convolution
DSP	Digital Signal Processing
FC	Fully Connected
FF	Flip Flop
$FPGA$	Field Programmable Gate Array
HLS	High Level Synthesis
HW	Hardware
LUT	Lookup Table
MPU	Max Pooling Unit
$MVAU$	Matrix Vector Activation Unit
NN	Neural Network
$PMBus$	Power Management Bus
$RAMA$	Resource-Aware Model Analysis
RTL	Register Transfer Level
SNN	Spike Neural Networks
SRL	Shift Register Look up table
SWU	Sliding Window Unit
XPE	Xilinx Power Estimation

List of Changes

Section	Statement of Changes	Page Number
Abstract	Complete update to make it reflect the new results and goals of the thesis.	ii
Section 2.1	General correction of facts. FINN is a data-flow architecture, hence it will not access DRAM as much as was originally implied.	3
Section 2.1.1	Increased the review of the power measurements. Added in the table for the FINN power results. Revised the review of the FINN paper to remove comparisons to external networks and focus on the PMBus and wall power differences.	4
Section 2.1.1	Removed paragraph on XPE and references to spiking NN.	4
Section 2.2	Swapped positions with Effects of Binarization. Corrected some errors in the first paragraph. Added in all the additional sub headings about how FINN works. Refined the folding and partitioning section to now be inline with FINN.	5
Section 2.3	Provided information on how binary operations work. Removed section on scaling of networks. Minor corrections for the Different Levels of Quantisation.	10
Section 2.4	Removed mention of ImageNet. Provided information about each data set. Included specific topologies for each of the networks that will be used in the work.	12
Appendix C.1	Updated the literature review tables to have combined references.	56

Chapter 1

Introduction

In 2016, Venieris et la. [1] released a paper detailing a methodology to automatically map Convolution Neural Networks (CNN) onto Field Programable Gate Arrays (FPGAs). Thus enabling FPGAs to become viable hardware accelerators for neural networks. Where it once took months for a seasoned professional to hand code the network [2] it could now take an hour or two for a program like FINN [3, 4] to convert a NN model into a working bit file.

NN have been rapidly developing over the last decade in the field of image classification. The primary method for training and running these NN are on traditional CPU and GPUs chipsets [2, 5]. However, these chipsets whilst having the backing of years of design and experience being developed, are limited in their ability to provide fast, high parrallel processing of networks at a power efficient manor. CPUs are slow with limited ability to use parallel processes and GPUs are power intensive [1, 2, 6–8].

Binary Neural Networks (BNN) are NNs which have their weights and activations quantised down to a single bit [9]. At this level of quantisation the network not only receives a significant drop in hardware (HW) footprint but also allows them to utilise binary bitwise operations [3, 8–11]. Together with their affinity towards the FPGA fabric they appear to provide a significant improvement to the current quantised networks in terms of power efficiency.

1.1 Project Aim

The aim of the project is as follows:

- *To determine if BNNs provide improved power efficiency over 8-bit quantised NNs at image classification.*

BNNs have also been presented as having a significantly decreased accuracy [10, 12]. To compensate for this drop in accuracy the NN needs to increase the amount of parameters present. Partially offsetting the improvement in HW resources [3].

For the thesis, the power efficiency of pre-trained networks will be compared with custom developed networks. The networks will be developed using Brevitas, a quantisation aware training tool for PyTorch. Then converted over to a PYNQ-Z2 FPGA for testing. For BNNs to be considered more power efficient in image classification tasks they will need to have a statistically significant, 95% confidence interval, increase in power efficiency measured in FPS/Watt. Additionally, a new metric FPS/Watt*(Top-1 Accuracy) will be compared in the same manner to ensure that the networks have not sacrificed accuracy for power efficiency.

An additional quantisation level of 2-bit, trinary, networks will be used to determine the impact of the binary operations separate from the benefits provided by the reduction in bit width.

Chapter 2

Literature Review

In this literature review four aspects will be covered: FPGA HW, effects of binary quantisation, NNs and data sets, and mapping a NN to FPGAs.

2.1 FPGA HW and Power Efficiency

To effectively compare between BNNs and NNs, knowledge of the FPGA HW is required. This section examines how FPGAs consume power and how other papers have recorded power efficiency.

For the FPGA two types of power drain are present: dynamic, and static. Static power drain results from current leakage in the system [13, 14]. Shang et al. [14] reviews the dynamic power consumption of a Virtex-II FPGA. The paper directly determines the static power drain for the FPGA to be between 5-20% of total power. Both Shang [13] and Jevtic [14] conclude that static power will increase as the FPGA's temperature rises. The dynamic power consumption refers to the power consumed as the FPGA operates. Dynamic power consumed will depend on the configuration of the modules, the length of the relays, and the type of operations performed. For the Virtex-II the majority of the dynamic power comes from the interconnections between the Configurable Logic Blocks (CLBs), approximately 50-70% [14]. Jevtic et al. [13] present a calibration model to be used on high level power estimation tools, like Xilinx Power Estimation (XPE) [15]. Their measurements on the Xilinx XUP board directly contradict Shang's claim stating that the logic operations are the most power intensive operation of the FPGA consuming between 67-77% [14]. If Shang is correct, then FINN will need to optimally route the CLBs to minimize the length of interconnections. If Jevtic's report is correct, then FINN's routing will not be as great an impact on overall power consumed.

A factor not mentioned in the above two papers is the power cost of accessing external memory. Machupalli et al. [16] implements various NN architectures on ASIC chips. During the tests the weight stationary model is determined to have the least access to external memory compared to the two other methodologies. It is this

limited access to external memory that enables the weight stationary model to have a decreased power requirement. The paper uses an ASIC chip instead of a FPGA. As such Machupalli's work only provides a guide for this thesis. Duy Thanh Nguyen et al. [12] supports the ASIC findings on their FPGA. They state that decreasing access to DRAM also has the added benefit of reducing bottlenecks and increases the streamlining of layers. The resulting improvements in speed allow the power efficiency to be boosted. FINN uses a data-flow architecture, which is a version of the weight stationary model. As a result, FINN will have very little access to external memory. Only requiring reading and writing operations when reading in the image stream and outputting the results.

2.1.1 Measuring FPGA Power

Previous papers have reported power efficiency through various units, Table 2.1. The operations per second per Watt remains out of the scope of this thesis due to the lack of tools that would be able to measure it. The second option, FPS/Watt, will be selected to be the primary method for comparing network power efficiency. Measurements using this metric will not be comparable across data sets as the frame sizes are not constant.

Paper	Power Efficiency Unit
[1],[8]	GOp/s/W
[6],[3]	FPS/W
[7]	Joules

Table 2.1: Power Efficiency Units from Literature

Each of the listed papers in Table 2.1 measure chip power directly using a Power Management Bus (PMBus) or equivalent. Without a PMBus the system requires the wall power to be recorded instead. Umuroglu et al. [3] present power recordings for both on chip and wall measurements whilst presenting their work on FINN, the selected framework for building FPGA accelerators. In their experimental design they develop three BNN prototype networks using FINN to demonstrate the tools capabilities. Of note are the convolution network inspired by BinaryNet [9] and VGG-16 [17]. The experiment uses a Xilinx Zynq-7000 All Programmable SoC ZC706 Evaluation Kit with an onboard Zynq Z7045 FPGA. The board draws a static load of 7 Watts from the wall. They note that increasing the throughput of

a network increases power efficiency. Of importance is the inclusion of both wall and PMBus measurements. The paper has three types of networks: Small Fully Connected (SFC), Large Fully Connected (LFC) and Convolution (CNV). MAX and FIX represent the scenario of operation. MAX will have the network run as fast as it can with FIX limiting the network to run only at 9000 FPS. All the networks tested in the paper are a mix of binary and trinary weights and activations. They do not compare their results to 8-bit networks. Power results are listed for the binary variants in Table 2.2, with the wall power having the idle power loss of 7 Watts removed. The relative change between the PMBus and the Wall power is not constant. As a result, the wall power measurement is not a direct stand in for the PMBus.

Name	PMBus (W)	Wall Power (W)	Relative Change
SFC-MAX	7.3	14.2	95%
LFC-MAX	8.8	15.6	77%
CNV-MAX	3.6	4.7	31%
SFC-FIX	0.4	1.1	175%
LFC-FIX	0.8	0.9	13%
CNV-FIX	2.3	3	30%

Table 2.2: FINN Power Results

2.2 Mapping Neural Networks onto FPGAs

The recent development of automatic mapping NNs to the FPGA have been a key development in enabling this work. It has significantly reduced the development time of each network; allowing for multiple NNs to be evaluated in this work. There are two key architectures for deploying NNs to FPGAs, data-flow and overlay style. Hamanaka et al. [6] review both architectures to determine whether the data-flow method provides better performance and power efficiency than the overlay style. They use an experimental research design in their paper. The design uses the CIFAR-10 data set and ResNet-8 as the baseline model. Vitis-AI was used to implement the overlay style on a Kria-K26-SOM FPGA. For the data-flow implementation, FINN was extended to FINN-S to enable the use of the same HW board as Vitis-AI. The results showed that the data-flow method excelled by providing a higher performance and power efficient implementation than Vitis-AI. An efficiency of 2,287 FPS/W for FINN-S to Vitis-AI's 695 FPS/W. Hamanaka uses a

FPGA with higher HW resources than the one used in the thesis. The advantage of a powerful FPGA allows networks to have less folding and increased parallelism, improving throughput. As a result, the efficiency is greatly increased. The same results are not guaranteed with a resource constrained FPGA, such as the one that is used in this work. The results of Hamanaka's work justify the use of FINN and its data-flow architecture in this thesis.

Three mapping tools in total were considered for the thesis, FPGAConvNet [1], DeepBurning [7], and FINN [3]. Out of the three listed here only FINN is in active development. Funded by Xilinx Research, FINN provides a tool that enables the user to convert a quantised NN into a FPGA bit file. FINN also provides users with the ability to customize its open source code and extend the program like Hamanaka did with FINN-S [6]; feature that will not be used in the thesis. The other tools mentioned have not been developed since their papers original release. As such FINN has been selected as the bridge between the computer generated NN's and the FPGA.

2.2.1 FINN Synthesis Steps

FINN implements a data-flow model. As such all the layers of the neural network will be loaded onto the FPGA at the same time. Creating a resource bottleneck for the development of neural networks in the thesis. The networks need to be small enough to fit entirely on the FPGA. Kumar [4] writes about how to decrease the time of synthesis in FINN and how to lower the amount of resources used in convolution layers. Their work focuses closely on how FINN's synthesis stage can be sped up by employing fine grain control over the implementation of the Matrix Vector Activation (MVA) blocks and their re-usability during synthesizing similar blocks. Kroes [18] writes about optimizing the memory mapping for FINN. Their work focuses on how to improve BRAM utilization without impacting system throughput. These two papers will be used to explain how FINN works in the following paragraphs. Figure 2.1 presents a diagram of the overall work flow.

FINN works by consuming a pre-trained quantised NN from a supported file format and converting it into a bit file for FPGA deployment. For the thesis, Brevitas will be used to train quantized networks. FINN will convert the Brevitas file into

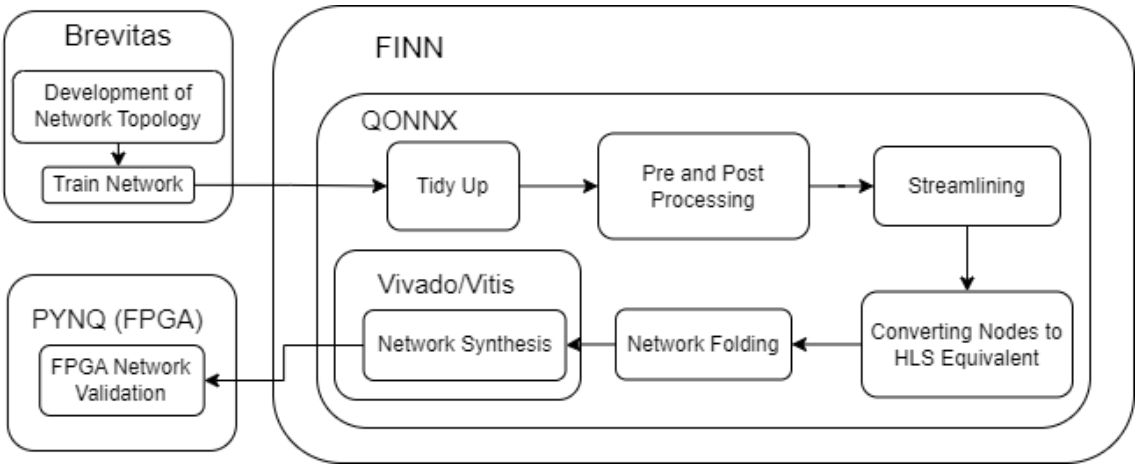

Figure 2.1: FINN Work Flow

a directed acyclic graph represented by Qonnx. The graph is then transformed by calling FINN transformation commands until the generic graph nodes contain FINN specific nodes. These nodes are then passed to Vivado HLS, Vivado and Vitias to be converted into the bit file.

FINN provides worked examples on their GitHub page [19]. Work from this will also be included alongside information from Kumar on how FINN works. The first step applied to all models are the tidy-up transformations. In this stage the network is prepared for the following stages by defining nodes inputs, outputs, and tensors, providing them with human readable names and cleaning up the graph with constant folding. Pre and post processing steps are also included in the FINN example, however, are omitted from Kumar's description. Pre-processing allows the raw data to be directly inserted into the accelerator without needing the host CPU to process it into the data type that was used at training. Postprocessing adds a final top-K node to the graph that will select the inferred classification. The next step is streamlining the network. Umuroglu [20] developed a method to convert floating point networks into quantized equivalents. FINN uses this method to quantize any remaining unquantized values that have been left in the network. This step also moves nodes around the graph in order to position them in a way that other transformation methods are able to collapse them into multi-thresholding layers. Each network will have to have its own tailored streamlining implementation as the move transformations are highly network specific. The next step involves converting the compatible nodes into FINN nodes that are specified in the FINN_HLS library.

At this stage the High Level Synthesis (HLS) nodes will all need to be in sequential order. *Create_dataflow_partition* will split the graph up into the FINN_HLS nodes and other remaining nodes. This step will fail if non-HLS nodes are distributed within the FINN_HLS nodes.

The next step sets the folding parameters for the network. The code generation step converts the FINN_HLS nodes into C++ code. This code is used to inform the IP Generation step about how to create a RTL description of each node using Vivado HLS. The next step uses Vivado IP Integrator to stitch all the nodes together using AXI-Stream protocol to link each of the RTL nodes together. Finally, the last step runs the synthesis of the entire network placing and routing the network onto the described FPGA. It is at this stage resource requirements are checked. If the network over utilizes the available FPGA resources the synthesis will fail. If it does not fail, then the bit file will be generated for the network to run on the FPGA.

2.2.2 FINN HLS Nodes

For the thesis, three of the HLS implementations from FINN's HLS library will be implemented, the Matrix Vector Activate Unit (MVAU), Sliding Window Unit (SWU), and Max Pooling Unit (MPU). The MVAU is the primary means of computation within FINN. It is used within FC and Conv layers. It multiplies the weight matrix and input activation vector together. Accumulating the outputs and applies the activation function on the accumulated result. FINN supports two types of memory modes for the weight matrix. First is constant memory, in which the memory is stored within the MVAU node. Secondly, the decoupled memory mode creates a separate block that holds the weights. These are then streamed into the MVAU input [18]. This creates additional overhead in the system.

The SWU leads the MVAU in all Conv layers. It will store the input stream in line buffers that will then be passed into the MVAU as dictated by the SWU parameters. Breaking up the input feature maps into MVAU compatible inputs. The MVU is implemented as a line buffer that when full will apply the maxpool operation to determine the output value.

2.2.3 Folding and Parallelism

Folding and partitioning a network allows large networks to be deployed onto FPGAs. Venieris et al. [1] explains the concepts of folding and partitioning a NN in their work on FPGAConvNet. This work is later referenced in FINN as the source for folding. Partitioning is the act of splitting up a NN into sections. Two levels of folding exist, coarse and fine grain. Fine grain folding only works on convolution and pooling layers. The dot products in these layers are time multiplexed onto the same hardware section. If no folding were to take place, one dot product would take place on a single clock cycle. Coarse grain folding works on time multiplexing a layers operations. Umuroglu [3] goes into detail explaining that for a network with no folding each neuron in a network will have its own hardware implementation. In FPGAConvNet these three parameters allow designers control over how the network is implemented.

When viewed from a hardware perspective each MVAU node has two parameters, Single Input Multiple Data (SIMD) and Processing Elements (PE). Each PE handles the same input image stream but multiplies it by a different part of the weight matrix [3, 4]. SIMD represents the synapses, increasing the number of SIMD increases the parallelism that each of the different inputs can handle. If the SIMD equals the number of input streams then each input will be computed at the same time. This is demonstrated in Figure 2.2. The first half displays SIMD when it is two, allowing for both the input streams to be processed at the same time. The second half PE is equal to two, allowing for both neurons to process the single input at the same time.

The computer folding method uses FINN's *setFolding* transformation, which aims to set the *number_of_cycles_per_frame* to a default value of 1,000. It achieves this by increasing each layers PE and SIMD values up until it reaches the required value. First *setFolding* will increase the number of SIMD channels until the target cycles are achieved. If the number of cycles is greater than the weight block width it stops increasing SIMD channels. It will then increase the number of PE's from one until it either reaches the target or the number of PE's reaches the number of channels.

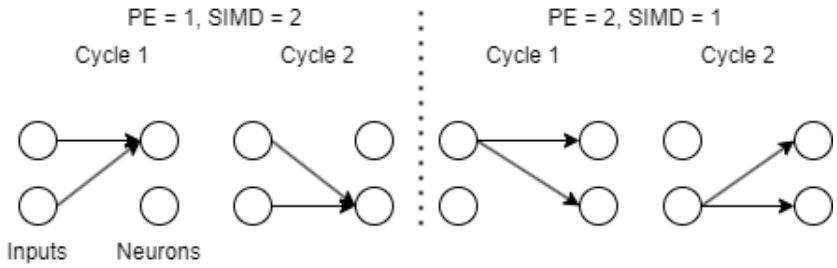

Figure 2.2: FINN HW Parallelism

2.3 Effects of Binary Quantisation on Networks

Binary Quantisation of a network involves quantising a network's weights and activations to a single bit each. A BNN will be able to use bit-wise operations instead of more resource intensive multi-bit operations. The first BNN was submitted by Courbariaux et al. in 2016 [9]. In their paper they outline the methodology used and showed that the results of the network are on par in terms of accuracy for CIFAR-10, however, suffer on ImageNet when compared to the then state of the art NNs. Specialised topologies need to be adopted for BNNs to be effective. Following the release of Courbariaux's work, additional papers focused on developing the topologies of the BNN to allow for more accurate inferences on complex datasets. Liang [8] implemented the original BinaryNet [9] on the CIFAR10 and MNIST dataset and XNOR-NET [21] on ImageNet. They deploy both BNNs onto a FPGA using their Resource-Aware Model Analysis (RAMA) tool. They report that a loss of accuracy of 0.46%, 2.75% and 13.7% on the MNIST, CIFAR-10 and ImageNet datasets, respectively. These losses are contrasted by a weight reduction of 32x, 32x, and 21.86x, respectively [8]. Allowing for more of the network to be loaded onto FPGA memory than the original NN. The reduction in HW footprint and the ability to reduce costly matrix operations into bit-wise operations is the primary appeal for BNNs. However, the decrease in weights will lead to loss of accuracy. This loss of accuracy will be mitigated if the system is scaled up in size [3, 10]. As such, raising the question of whether the gains of reducing HW will be undermined by the need to scale up the BNN.

The binary operations that BNN's use are expressed as providing 32x speed up in

GPU implementations by Courbariaux. Due to the ability to concatenate 32 binary variables into a single 32-bit register on the GPU providing the stated speed up for binary operations. FPGAs are flexible in their implementation and binary operations naturally lend themselves to the FPGA fabric. The nature of binary weights and activations lead to three optimizations that can be made in the networks, binary accumulation, batchnorm-activitation and max-pooling. For binary accumulation only two values, 0 or 1, are viable. As a result, only the set bits need to be counted with the others being inferable. This allows for the use of *popcount* to count the number of set bits. FINN shows that compared to the alternative signed accumulation counts the binary *popcount* method utilizes approximately half the amount of Look Up Tables (LUTs) and Flip Flops (FFs) for HW implementation [3]. For all layers in BNNs batch normalization's are applied to the outputs before the activations are determined by the selected sign function [3, 9]. Batch normalization presented by Ioffe [22] presents a method to convert the outputs of a layer to have zero means, unit variances and decorrelate them. These batch normalisation's provide an alternative to dropout. Additionally, the batch normalisation's when combined with the signed operations consume a larger degree of HW when compared to the described methodology of thresholding [3]. Max-pooling can be replaced with the Boolean OR operations. It is observed by Umuroglu [3] that the max pooling can be applied after the threshold layer. As such given the nature of MAX the output will be high if any of the inputs to the threshold layer are greater than the threshold. As such MAX can be replaced with a binary OR operation after the thresholding operations on all the inputs.

2.3.1 Different Levels of Quantization

Different levels of quantisation will lower the amount of HW requirements whilst not degrading accuracy of the network. However, weights and activations that are not binary will result in the NN not benefiting from the previously mentioned use of bit-wise operations. Nguyen et al. [12] tests various quantization levels of YOLOv2 on a FPGA. The experimental design of the paper tests the impact of various levels of quantization of the activation values. They use YOLO-v2 on various years of PASCAL VOC datasets as a test set. Their results show that the quantised networks still suffer accuracy losses as the activation values are increasingly quantised. For

the YOLO-v2 comparison the system lost 2.5% accuracy and gained a 30x and 5.4x reduction in weight and activation sizes, respectively [12]. Nguyen leverages these HW reductions by implementing a streaming architecture which is normally prohibitively expensive to install onto a FPGA. Nguyen uses a Virtex-7 VC707, a significantly larger FPGA than the selected Z-7020. Nguyen's experiment fails to deploy a fully BNN to test the quantized model against citing the lack of accuracy BNNs have for large complex data sets. In the thesis quantised NNs will need to be employed to test against BNNs. As FINN is a data flow system a quantised NN will need to be used to enable it to fit onto the FPGA. Supporting the quantisation of the NNs are Shang et al. [23] and Fu et al. [24]. Shang's work [23] on a GPU demonstrates that 8-bit quantisation improves the speed of results by 27x when compared to 32-bits as it allows for greater data parallelism. Fu [24] lists the use of INT8 as the optimal precision for DSP48E2 present in Xilinx boards. As a result, the thesis will use 8-bit NNs to compare the BNN models to.

2.4 NN and Data Sets

It is important to select data sets that are common across the literature to enable effective comparisons. As a result, multiple papers where compiled to present a holistic view of the literature. Works like Blaiech et al. [5] provide an excellent secondary source of information used throughout the literature review. Data sets across the literature are standardized, the HW that networks are tested on are not. To mitigate the issue Blaiech [5] presents the listed improvements as percentages generated from the baseline provided in the original paper. They state that it is preferable to use a GPU during the training stage. The use of a framework, like FINN, is recommended for the implementation of the system onto the FPGA. Importantly, it is from Blaiech's work that an overview of all the research done can be assessed. The data sets and NNs that are common throughout the reviewed papers are listed in Table C.1 and Table C.2 respectively. Only the reviewed papers in the thesis are listed.

The two most common data sets used throughout the literature are CIFAR-10 and MNIST, Table C.1. CIFAR-10 was developed by Krizhevsky [25] and contains ten

image categories. CIFAR-10 is a low-resolution data set with images having a resolution of 32x32 pixels. The data set has 5,000 images for the training and 1,000 for the test set for each category. MNIST by Lecun [26] contains 60,000 training images and 10,000 test images. The data set contains a series of handwritten characters that are used as a bench marking tool for computer vision [26–28]. A review paper of MNIST completed by Baldominos et al. [28] published in 2019 details all cutting edge implementations of NNs on this data set. It is notable that the review does not mention any BNNs. However, for the thesis only FC networks will be tested on MNIST with convolution networks to be tested on CIFAR-10.

2.4.1 MNIST Networks

For the MNIST data set three networks have been selected. The 8-bit and trinary networks will be using a single hidden FC layer as described by Lecun [26]. The original networks were not quantised to 8-bits nor trinary. Lecun did not state that activation's were used in their work. Two types of single hidden networks were presented in the work. One with 300 neurons in the hidden layer and one with 1000. Lecun stated that the hidden network with 300 had an error rate of 4.7%, 95.3% accuracy, and the 1000 network had an accuracy of 95.5%.

For the BNN and trinary networks FINNs own trained networks [29] will be deployed to the FPGA. The TFC network for 1-bit and 2-bit weight and activation's will be used. The accuracy of these networks are 93.17% and 96.60% respectively. The topology of the TFC networks includes three FC layers of widths 64 each. Drop out and batch normalisation is included between all FC layers.

2.4.2 CIFAR-10 Networks

For the CIFAR-10 data set the quantized network will use Lenet5 described by Lecun [26]. Lenet5 was originally developed for the MNIST data set. As a result, it suffers from poor accuracy on the more complex CIFAR-10 images. However, its small topology will better fit onto the FPGA. An adapted version of the network developed by Raschka [30] will be used in the thesis. Raschka increases the the number of kernels by a factor of three to offset the difference in CIFAR-10 three colour channels when compared to the original single channel in MNIST. The network has a reported accuracy of 67.30%. The network consists of three convolution

layers with a stride of one and 5x5 kernels followed by three fully connected layers. The signmod activations have been replaced with ReLu, which makes this network compatible with FINN.

To provide better accuracy a medium sized network developed by Calik [31] will be considered. The network was developed for CIFAR-10 and aims to be as small as possible. Calik states that the network only requires 2GB of memory using full precision. The goal of Calik's network is to enable it to be used on embedded devices. Although the work still uses a GPU to run the network. This network topology has 4 convolution layers with pooling layers in between each convolution layer and finishes with two fully connected layers. The network has a listed accuracy of 85.9%.

Finally, FINN uses a variation of VGG-16 developed by Simonyan [17]. VGG-16 is the largest network considered in the thesis. FINN's VGG-16 comprises of six convolution layers and three fully connected layers at the end. The convolution layers have a kernel size of three with a stride of one. Additionally, the pooling layers have a size of two. For the BNN and trinary networks FINN's CONV networks have an accuracy of 84.22% for the Binary variant and 89.03% for the trinary [32].

Chapter 3

Research Design

The goal of the thesis is to determine whether BNNs are more power efficient than 8-bit quantised counter parts. To test the hypothesis two data sets have been selected, MNIST and CIFAR-10. Both have been selected as they are standard data sets used throughout the literature, Table C.1. Additionally, both where selected due to their low complexity. The low complexity allows for simpler NN works to be used on them. Smaller networks allow for a decrease in both training time and HW resource footprint. Both of which are limited resources in the thesis.

3.1 Developing the NNs and FINN

When developing the networks for the thesis three constraints need to be met. Firstly, FINN only allows for a very limited number of supported layers to be implemented. Secondly, FINN needs the layers to be ordered in a way that can be translated into HLS code. Finally, FINN requires the entire network be loaded onto the FPGA at the same time. As stated before, this limits the ability of networks to be scaled in size and reduces the number of acceptable network topologies. FINN does support modifying a network's topology during its *qonnx* stage. However, the changes that can be made are limited. As a result, it is best to develop networks to be compatible with FINN to begin with.

The selected network topologies are presented in Section 2.4. The selection criterion for the networks was firstly based on size. Large networks are excluded from consideration as they will not fit onto the FPGA. The second selection criterion was complexity of the networks. As FINN only has a limited number of supported layers more complex networks like ResNet with its residual functions would not be supported. As a result, the selected networks are the simplest types currently available. Finally, the last selection criterion was accuracy. Accuracy whilst important was not the focus of the thesis. The higher accuracy networks were selected from those that could run on the FPGA.

To develop the networks a starting network topology will be selected and implemented using quantised equivalent blocks. The network will be tested for compatibility by running through all but the synthesis steps of FINN. Using the inbuilt *showInNetron* various transformation steps will be run to convert the network into the HLS compatible nodes. Whilst many of the transformation steps are standard between each network changes in their topology will still require the transformation steps to be reviewed. Both synthesis and training tests will be run in parallel. The synthesis tests will ensure that the network is FINN compatible. The training will test to see if the network is able to be trained. If it can be trained then it is left to gain some accuracy whilst the synthesis tests are completed. Synthesis of a network can take between 20 minutes to an hour. If the network is able to be fully synthesized then the training will be suspended and the network saved. The final test will be synthesizing the partially trained network. If the network retains its accuracy when running on the FPGA then the topology will be considered compatible; the network will receive the full ten hours of training. A diagram of the development model is presented in the Figure 3.1.

The original structure for the MNIST 8-bit quantised networks will be the smaller of Lecun's single hidden FC network, 300 nodes. The TFC network from FINN will be used for both the binary and trinary networks. A trinary network version of Lecun's single hidden FC network will also be adopted.

For the CIFAR-10 data set the binary and the trinary networks from FINN will be used for their respective categories. Both Lenet5 and Calik will be tested for the 8-bit network. Lenet5 will be tested first as it is a smaller network. The version of Lenet5 that will be tested is the one from Rasbt [30] which increases the number of channels in the Conv. Layers by three and the inputs and outputs of each FC layer by the same factor. This is done to offset the three input colour channels instead of the single monotone of the MNIST data set. Both Lenet5 and Calik will be tested to see if they fit on the FPGA. As Calik is the larger of the networks by design, only Lenet5 will be shrunk to fit onto the FPGA. As it is originally the smaller network.

Once the network topologies have been checked for complete compatibility each network will need to be fully trained. Ideally each network would be given an

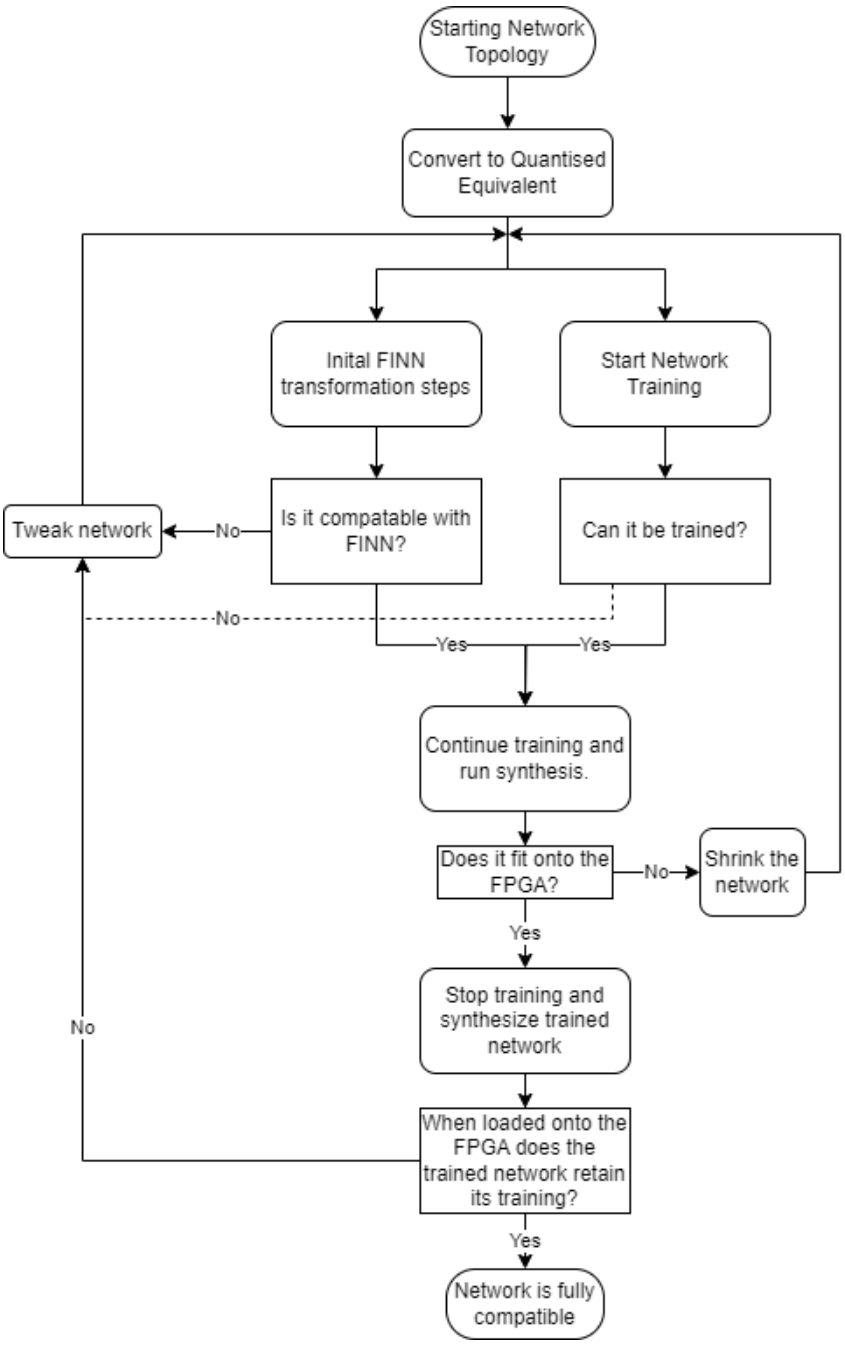

Figure 3.1: Network Development Cycle

unlimited amount of training time to reach their maximum accuracy. However, time constraints, the tendency of networks to reach a saturation point and the unreliability of generated networks has limited the amount of time each network has to train to ten hours. The learning rates of each network will decrease by 50% every 40 epochs, as shown in FINN's examples.

3.1.1 Implementing NN's with FINN

A large portion of time was spent working to develop an understanding of FINN and how to use it. Many combinations of nodes and network topologies have been tested during the course of the thesis. FINN's work flow consists of 3 key programs.

For FINN to convert layers into HLS equivalents the NN needs to be constructed with FINN at the forefront of design choices. The final design methodologies developed for the thesis are presented in the next paragraphs.

The implementation method was developed from reviewing the code from the FINN Examples [19]. All networks need to start with a quantization identity layer. The initial quantization parameters for the thesis have been selected to be identical to all the following quantization layers.

For the first method a FC layer will be known as a FC Stack due to the requirement for multiple different layers to make up the equivalent PyTorch linear layer. A FC Stack will use a *QuantLinear* layer from brevitas with Bias set to false. Following the *QuantLinear* will be a *BatchNorm1d* layer, from torch, and a *QuantIdentity* layer from brevitas. *QuantLinear* implements pytorch's linear layer. However, the inputs, outputs, bias and weights fields are all able to be quantised [33]. For this work only the weights will to be quantised. The outputs of each layer will not be quantised. Instead they will be quantised using the *QuantIdenity* layer. This *QuantIdenity* layer is a simple layer that only quantises its inputs. *Tanh* and *ReLu* are supported by Brevitas and would act in place of *QuantIdentity* however both demonstrated incompatibility with FINN when used. For the CONV Stack a similar process is followed as the FC stack. However, the *QuantLinear* and *BatchNorm1d* were replaced with a *QuantConv2d* and *BatchNorm2d*, respectively.

MAX Pool is supported by FINN with a few important caveats. When in between CONV stacks the MAX Pool will be absorbed into the FINN logic. However, if

placed between a CONV stack and the reshape node, found between the CONV and FC stacks, it will not be converted into its HLS equivalent. This can be worked around by placing a QuantIdentity after the MAX Pool layer. However, since the CONV networks struggled to maintain their accuracy when converted to the FPGA with and without the fix in place it cannot be determined if this fix provides a solution to the problem or whether MAX Pool layers need to be avoided at the end of the convolution layers.

To set the folding parameters FINN's *SetFolding* transformation will be used. Custom folding will not be attempted in the work. A script titled *findBestFolding* uses binary search to determine the smallest *target_cycles_per_frame* that can be achieved. The smaller the number the larger more parallel the network implementation will be.

3.2 Measurements and Validation Methodology

The FPGA lacks an internal PMBus hence only the wall power of the system will be measurable. To measure power in the thesis two multi-meters will be connected across the power input of the FPGA. One will be set up to record the current and one to record the voltage. The meters will be video recorded with the intent to capture both readings at the same time. The power will be recorded as the multiple of both the current and voltage with an average power taken over the entire run.

To ensure that the system does not become effected by an increase in the static power each test will record the starting and ending power used before the network is run. These readings will be averaged and subtracted from the average power of each run. Additionally, each network will be run five times.

To record the throughput of each network the time it takes to run through the test set will need to be recorded. To achieve this the code running the validation tests will be modified to include a *perf_counter*. The timer will start at the beginning of the for loop that iterates over the test set. It will not include the initialization times needed to download and load training data. This method allows time measurements to be handled internally by the PYNQ Board. Power efficiency will be computed for all runs of the five tests using equation 3.1. And power efficiency scaled by accuracy will be computed using the average of the five runs, equation 3.2. Scaling

the network by accuracy will be a guide for the types of networks to consider. It is present as a means to emphasize the need for networks to be accurate and not just power efficient.

$$Power\ Efficiency = \frac{FPS}{Power} \tag{3.1}$$

$$Power\ Efficiency\ Scaled = \frac{FPS}{Power} * Accuracy \tag{3.2}$$

To validate whether the BNN is more power efficient for each data set the unpooled t-test will be taken with a 95% confidence interval. The hypothesis to be tested is the average sample mean for the power efficiency will be greater for the BNN than the quantised network.

The computer generated folding method provided by FINN will be used for all networks. FINN provides its own custom folding for the binary networks and this will also be recorded. To determine if BNNs are more power efficient only the computer generated version will be tested against. The custom folding is presented as a means to compare the effects of manually setting the folding parameters when compared to the computer's version. The custom folding method enables users to also select a mix of memory types for each layer. For the computer version only the decoupled *mem_ mode* will be used. The *findBestFolding* will be used to ensure the networks are as parallel as possible. Exploiting any saving in HW resources by converting unused resources into adding additional parallelism.

3.3 Developed Networks

The developed stack methodology was designed after reviewing the networks provided in the brevitas examples [29]. However, after many trials it was determined that the stack design was ineffective. A big contributor to the problems faced in the work was the requirement to get multiple different software applications to work with each other. A problem only compounded by the long synthesis times needed to check if the network synthesised correctly. The final FPGA accuracies, alongside the brevitas (computer) accuracies and the listed accuracies from the sources are presented for each network in Table 3.1. Key points of contention in the network development were as follows: what are the allowable brevitas layers that can be

Network	Dataset	Listed Acc.	Computer Acc.	FPGA Acc.
TFC BNN[1]	MNIST	93.17%[32]	93.17%	92.96%
TFC BNN[2]	MNIST	93.17%[32]	93.17%	90.90%
TFC (2-bit)	MNIST	96.60%[32]	96.60%	94.71%
SHL (2-bit)	MNIST	95.5% [26]	96%	86.97%
SHL (8-bit)	MNIST	95.3% [26]	96%	84.51%
FINN VGG-16[1]	CIFAR-10	84.22 % [32]	84.22%	84.19%
FINN VGG-16[2]	CIFAR-10	84.22 % [32]	84.22%	24.33%
Lenet5 (2-bit)	CIFAR-10	67.30% [30]	38%	10.04%
Lenet5 (8-bit)	CIFAR-10	67.30% [30]	58%	10.07%

[1] Custom Folding, [2] Computer Folding

Table 3.1: Network Accuracy

used? what arrangement are they able to be used in? which forms of quantisation allow for the network to be both trained and synthesised?

3.3.1 MNIST

For the MNIST data set both the TFC BNN and TFC (2-bit) worked out of the box. With a performance degradation of 2.27% and 1.89% between the brevitas model to the FPGA respectively. When comparing the reference TFC model provided by FINN to the altered one presented as computer folded the only difference between the two topologies is the folding. The FINN network used a manually defined folding scheme that defined both the PE and SIMD parameters but also the type of *mem_mode* that each layer would use individually. The custom folding method only resulted in a performance degradation of 0.21%. As a result it is concluded that the folding method used by the networks can have a measurable impact on the final accuracy of the network on the FPGA. The biggest impact is the *mem_mode* selected. The decoupled memory needs the weights to be streamed into the layer compared to the constant memory which has the memory bundled with the layer. Future work needs to test the impact of different folding schemes to examine the effect of each *mem_mode* and the degree of folding.

The alternate trinary network was developed from the Single Hidden Layer model developed by Lecun [26]. The trinary network had a higher then expected accuracy when compared to the original Lecun model on the computer but suffered a loss of 9.03% when brought over onto the FPGA. For the network to be able to reach

a high percentage accuracy on the computer it had to be increased to a width of 1024 in the hidden layer. The stack design functioned for this specific network. The network utilised *CommonActQuant* and *CommonWidthQuant* for its quantisation specifications. These quantisation specifications have been difficult to work with. From the experimentation they often appear to be the most FINN friendly specifications. However, the 8-bit network was unable effectively use the common quantisation specifications. Instead, needing use the predefined *Int8ActPerTensorFloat* and *Int8WeightPerTensorFloat*, the default quantisation for the *QuantIdentity* and *QuantLinear* layers respectively. For the 8-bit quantisation the stack method was abandoned in favour of enabling the layers bias and removing the initial quantisation layer. These actions increased the recorded accuracy on the FPGA from 41.43% to the listed 84.51%. A mixture of the folding, quantisation and network topology are predicted to be the cause of the 8-bit network loses 11.49%. The quantisation processes will not have completely impacted the trinary network as it used the more compatible common quantise definitions. Another important distinction between the 8-bit and the trinary was the width of the hidden layer. The 8-bit system was able to achieve its 96% brevitas accuracy only with a width of thirty two compared to the trinary's width of 1024. For the final network typologies are presented in Figure 3.2.

3.3.2 CIFAR-10

Only the FINN Binary network was fully compatible with FINN. The network was able to be loaded onto the FPGA and retain its accuracy score. This network was a derivative of the VGG-16 network topology, Figure 3.3. The trinary version for FINN was unable to be loaded onto the FPGA as it was too large. Calik's network was also to large for the FPGA. Lenet5 similarly was too large in its original state to be loaded. As a result the network was trimmed down significantly. The original contained a max pool between the last Conv. Layer and the Reshape. It was removed as it potentially posed a point of failure for the synthesis. To compensate for this removal the remaining max pool had its kernel size increased to four. The FC layers where cut from three to only one. For the trinary network the FC layers were reduced to two to improve the networks ability to classify the images. For the final CIFAR-10 topologies refer to Figure 3.3 . These reductions in the networks complexity limited

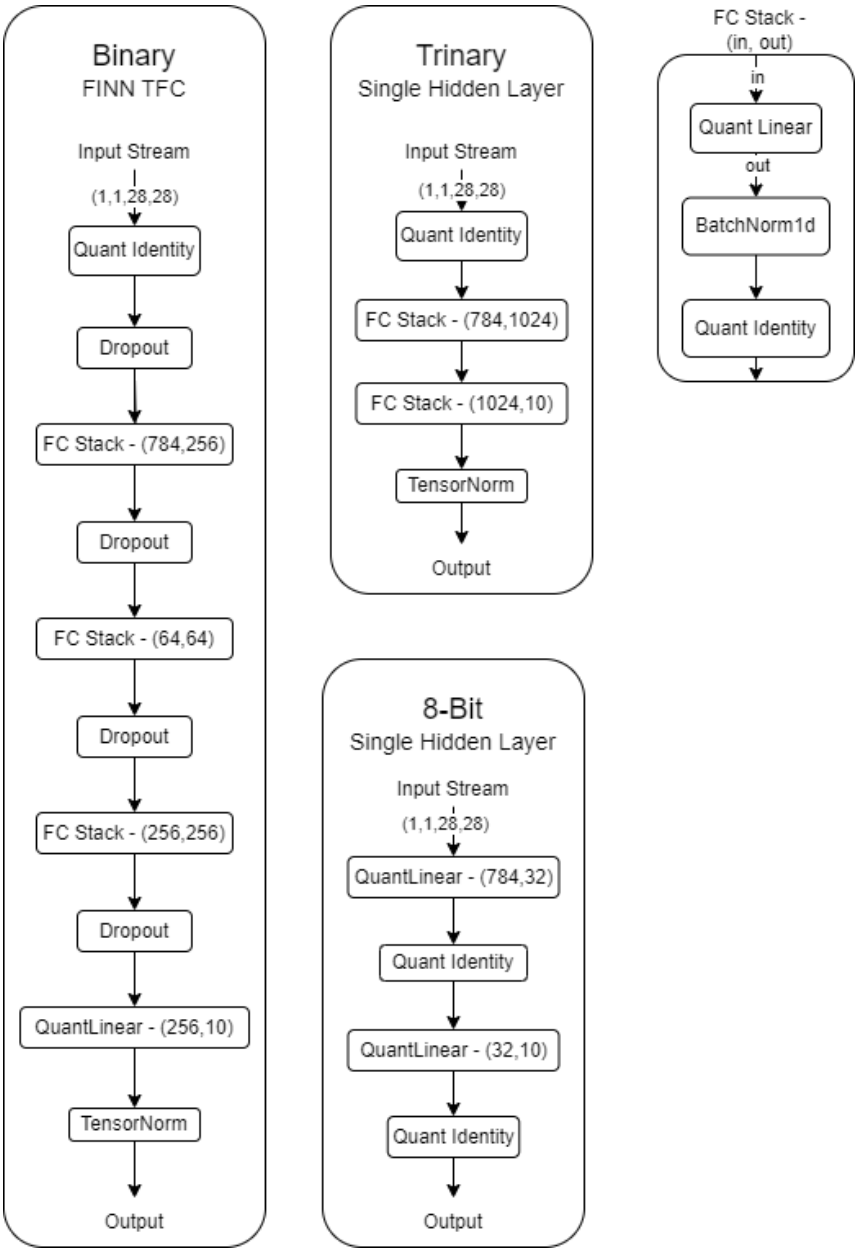

Figure 3.2: MNIST Network Topology

their accuracy significantly. The listed 67.30% accuracy for Lenet5 was developed from a extended version of the original Lenet5. This extended version increased all its parameters by a factor of three. For the thesis the networks had to decrease their parameters below the original Lenet5 version. This is a design limitation imposed by the FPGA. For the eight bit variant the impact of this reduction resulted in a reduction of 9.3%, Table 3.1. The increase afforded by the quantisation of the 2-bit network allowed an additional FC stack to be installed. Without the additional FC Stack the trinary network would not have been able to train and improve its accuracy above 10%. Since there are ten categories in CIFAR-10 an accuracy of 10% will be statistically achieved if the network is guessing the categories. The increase in the network size allowed it to start making predictions that were better then guesses however the network lost 29.3% of its accuracy due to the parameter reductions. Another factor that contributed to the networks decrease in accuracy is the quantisation types that are implemented in the network. If the quantisation was not set correctly then the network will not achieve any learning. The networks topology would also impact its accuracy. However, continued work on these aspects was overshadowed by the need to get the networks to be converted over to the FPGA.

Both quantised and trinary networks failed to match the computer accuracy when translated onto the FPGA, Table 3.1. Both the networks had a silent error(s) somewhere in the FINN work flow, resulting in an accuracy of approximately 10% each, equivalent to guessing. No error messages where generated by FINN that could have been used to diagnosed the problem. The root of the problem is considered to be in the convolution layers. This was not able to be tested as removing the convolution layers and replacing them with a FC layer had resulted in a network too large to fit onto the FPGA. Only a limited amount of attempts could be made to rectify the silent error as each attempt required 30 minutes to an hour of synthesis time to test the tweak.

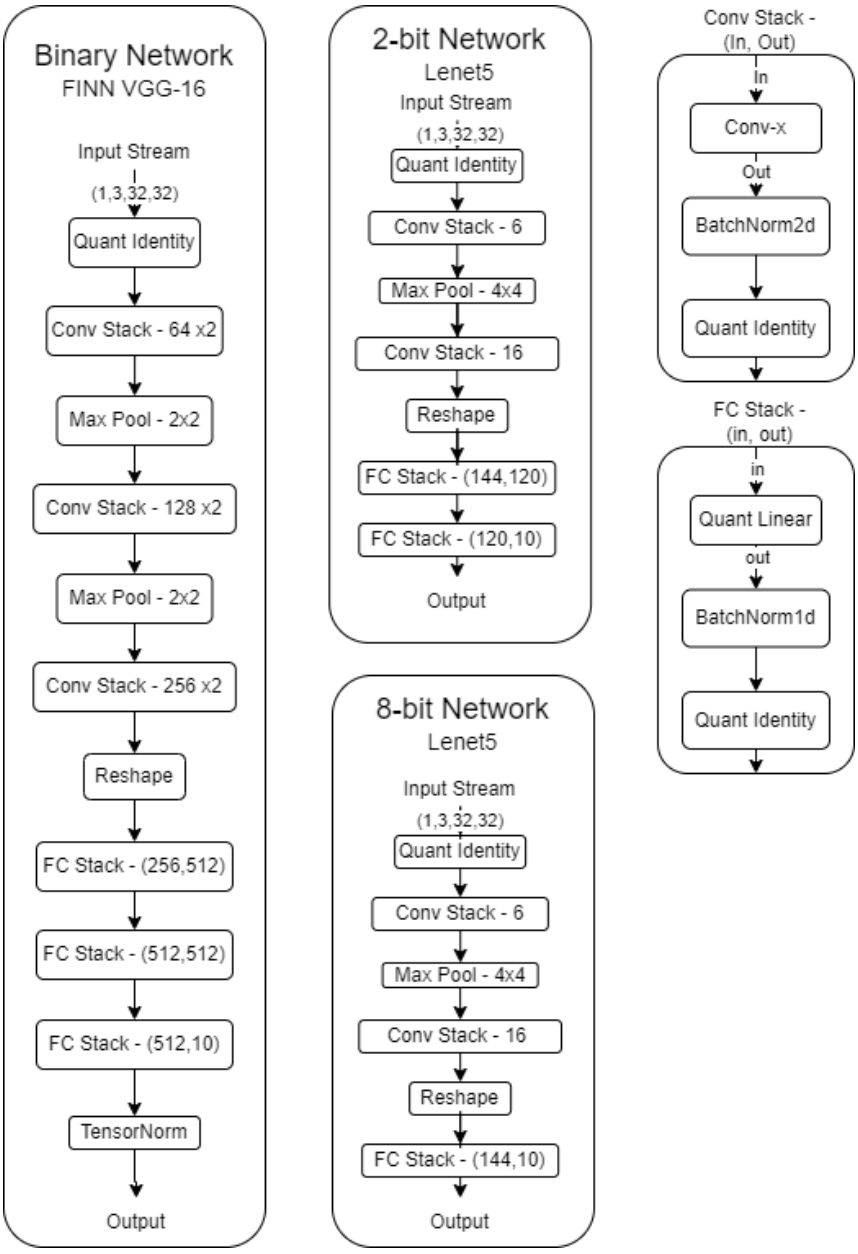

Figure 3.3: CIFAR-10 Network Topology

3.4 Power Efficiency of the Networks

When recording the power efficiency for the MNIST data set the two multi-meters struggled to provide more than a single reading before the data set was complete. With the TFC (2-bit) completing a run of the MNIST data set within 0.077 seconds, Table 3.2. On average the trinary network was more power efficient for both data sets, Table 3.2. The power used by the custom folded TFC BNN exceed the computer folded variant for the equivalent TFC BNN by 0.098 W.

For the CIFAR-10 data set the BNN with custom folding was half as fast as the other networks but was able to achieve a higher $FPS/W * Acc.$ than any of the other networks due to its accuracy not degrading when converted onto the FPGA, Table 3.1. Only the custom folded BNN was able to achieve a high accuracy with all the other networks failing the synthesis stage. For the raw data refer to Appendix C.2.

3.4.1 MNIST

For the MNIST data set both the TFC 1-bit and 2-bit networks were, with a 95% confidence interval, statistically more power efficient than all the other networks in the data set. This is clearly seen in the power efficiency box plot, Figure 3.4. Both the TFC networks were on par with each other in terms of power efficiency. When the power efficiency is scaled by the accuracy of the respective network, Figure 3.5, the results show on average it is the trinary network that is more power efficient than the others.

Network	Data set	Power (W)	Time (sec)	FPS/W	FPS/W*Acc.
TFC BNN[1]	MNIST	0.385	0.078	345,029	320,739
TFC BNN[2]	MNIST	0.287	0.078	454,909	413,513
TFC (2-bit)	MNIST	0.271	0.077	481,650	456,170
SHL (2-bit)	MNIST	0.294	0.139	261,879	227,756
SHL (8-bit)	MNIST	0.370	0.147	185,648	156,891
FINN VGG-16[1]	CIFAR-10	0.580	5.199	1,657	1,395
FINN VGG-16[2]	CIFAR-10	0.947	2.521	2,096	510
Lenet5 (2-bit)	CIFAR-10	0.177	2.727	10,700	1,074
Lenet5 (8-bit)	CIFAR-10	0.481	2.732	3,811	384

[1]Custom Folding, [2]Computer Folding

Table 3.2: Network Average Results

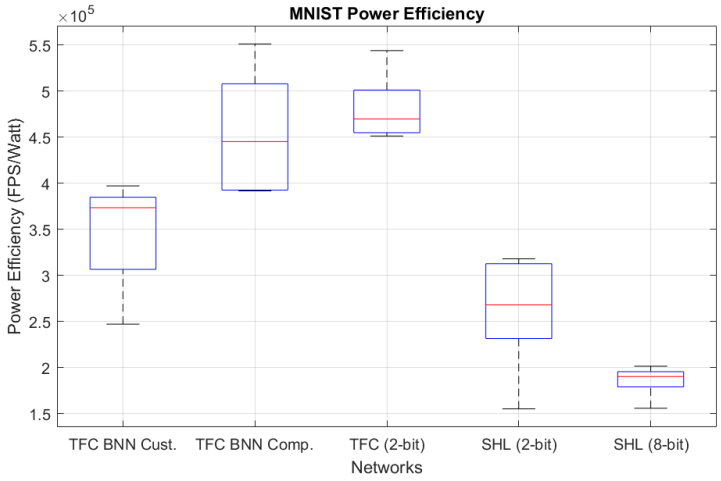

Figure 3.4: MNIST Power Efficiency

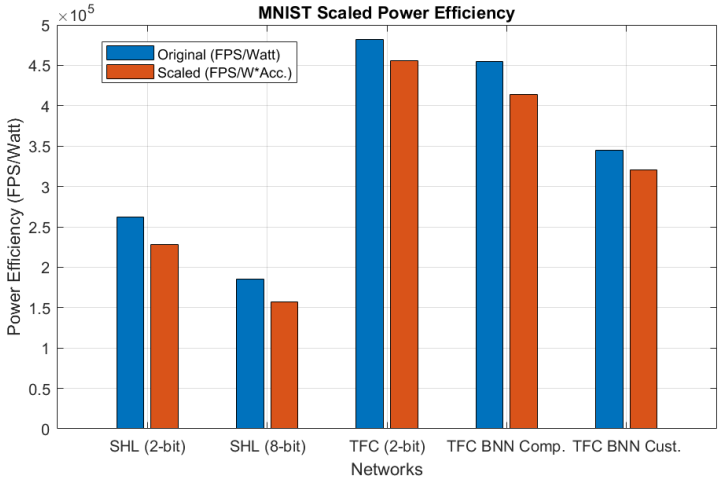

Figure 3.5: MNIST Power Efficiency and Scale Power Efficiency

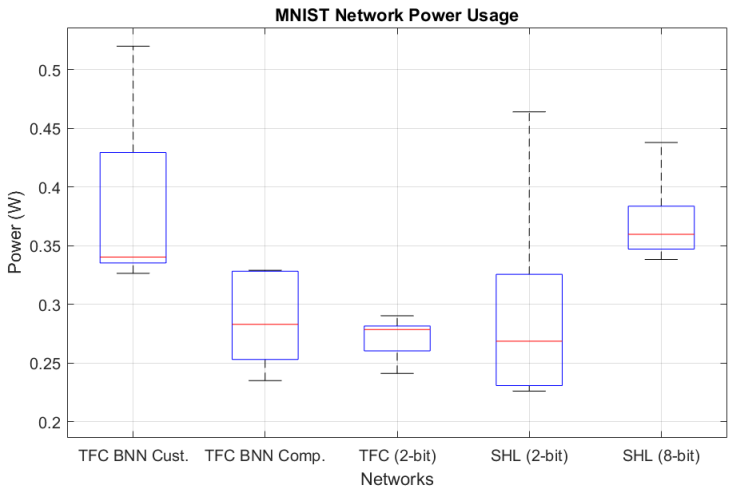

Figure 3.6: MNIST Power Usage

Comparisons	TFC BNN[1]	TFC BNN[2]	TFC (2-bit)	SHL (2-bit)	SHL (8-bit)
TFC BNN[1]	-	Greater	Greater	Null	Null
TFC BNN[2]	Null	-	Null	Null	Null
TFC (2-bit)	Null	Null	-	Null	Null
SHL (2-bit)	Null	Null	Null	-	Null
SHL (8-bit)	Null	Greater	Greater	Null	-

[1]Custom Folding, [2]Computer Folding

Table 3.3: Power Statistical Comparisons, 95% Confidence Interval

Figure 3.6 displays the average power each network used for the five test runs. When the statistical unpooled t-test is applied to all combinations of the power readings only the custom folded and 8-bit network use more power when compared to the computer folded TFC's, Table 3.3. The method to record power was ineffective as the power results are subject to external elements that are not the FPGA. The multimeters did not lend themselves to the high speed nature of the MNIST networks. Resulting in poor measurements due to low refresh rates. Future work needs to consider using a FPGA board that has a inbuilt PMBus. External factors like temperature, which increases the static drain, and onboard LEDs where minimised due to the static power being subtracted from the power readings. However, other external factors such as the power consumed by the ARM processor, and other unknown board processes could not be accounted for. A PMBus on the board would allow for the measurement of the FPGA without these external influences.

Comparisons	TFC BNN[1]	TFC BNN[2]	TFC (2-bit)
TFC BNN[1]	-	Null	Null
TFC BNN[2]	Null	-	Null
TFC (2-bit)	Null	Null	-

[1]Custom Folding, [2]Computer Folding

Table 3.4: Time Statistical Comparisons, 95% Confidence Interval

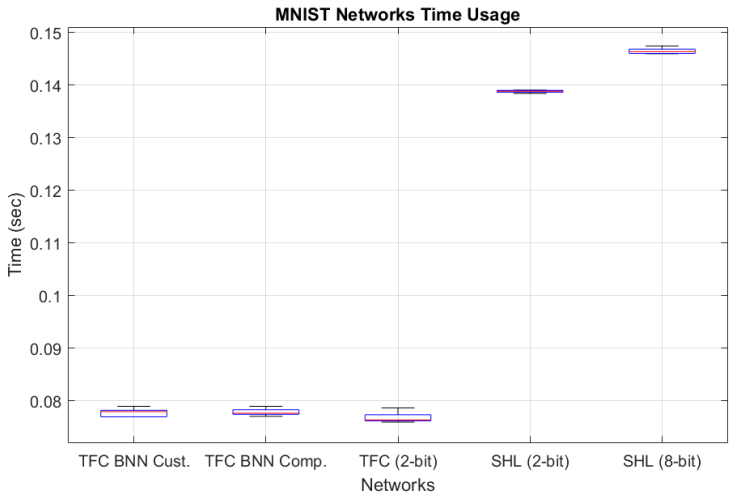

Figure 3.7: MNIST Network Time to Complete Test Set

One of the key factors in power efficiency was the throughput of a network. Increasing the throughput decreases the amount of time the network is required to run on the test data set. The amount of time needed for the FINN TFCs to complete one iteration of the test data set was significantly lower then the SHL networks over all five of the individual tests, Figure 3.7. To compare the times for the TFCs the same statistical test used in the power readings was performed. The SHL networks are excluded as they are visually distinct from each other. With the SHL (8-bit) taking the longest followed by the 2-bit variant. The results of the statistical test, Table 3.4, fail to reject the null hypothesis for all combinations. As a result none of the TFC networks out performed each other in terms of throughput in the test. The disparity in the TFC and SHL and the identical nature of the TFC times are due to the amount of folding each network was able to achieve. The lower folding leads to a higher degree of parallelism. Increasing the parallelism of the network comes at the cost of increasing the amount of HW resources needed. Hence, networks that originally had small amounts of HW resources where able to increase where networks that struggled to fit onto the FPGA would not.

Network	*target_cycles_per_frame*
TFC BNN	2
TFC (2-bit)	2
SHL (2-bit)	393
SHL (8-bit)	785

Table 3.5: SetFolding Parameters

Each of the computer folded systems ran the *findBestFolding* script to determine the smallest degree of folding that would be able to fit on the FPGA. By systematically changing the *target_cycles_per_frame* variable of the *SetFolding* method the thesis was able to determine the optimal folding parameters for each network, Table 3.5. Both the TFC models were able to have a single frame be computed within two clock cycles. Thus have a greatly increased throughput when compared to the 393 and 785 targeted cycles needed for the two and eight bit SHL, respectively. The amount of resources needed and the savings that each quantisation provides will be discussed in Section 3.5. The disparity between the different 2-bit networks is a result of the more efficient FINN TFC network topology. As the SHL had a width of 1024 the amount of resources needed to handle the large weight table is far greater than any of the TFC layers. The largest layer in the TFC required 64 PEs to reach the target cycle. But the SHL needed 128 for its 1024 wide layer. The greater amount of PEs for a single layer limits the flexibility of the system and consumes vital resources.

3.4.2 CIFAR-10

For the CIFAR-10 data set the networks behaved differently. It is important to note that the performance of all the computer folded networks is questionable due to their poor accuracy, Table 3.1. Additional work in the future will need to be done to ensure that the networks are performing correctly.

Both BNN networks failed to reject the original null hypothesis. It is visually apparent, Figure 3.8, that the most power efficient network was the trinary network with the 8-bit network following in second place. When reviewing the power used by each of the networks, Figure 3.9, the amount of power significantly differs between each of the network topologies. Unlike the MNIST data set the amount of readings for the voltage and current measurements that the multi-meters where able to record were significantly higher for CIFAR-10. As it runs for an order of magnitude longer then

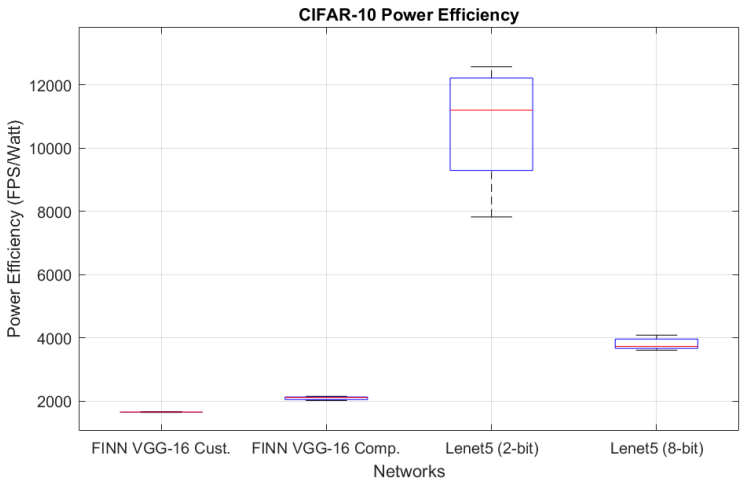

Figure 3.8: CIFAR-10 Power Efficiency

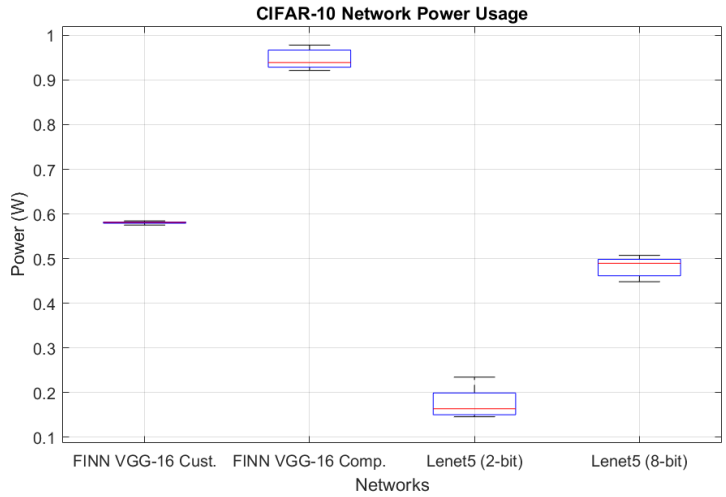

Figure 3.9: CIFAR-10 Power Usage

the slowest MNIST network on average, Table 3.2. The custom folded BNN takes on average double the length of time, Table 3.2, to process the CIFAR-10 test data set than the other two networks, Figure 3.10. Both the increase in the length of time and the amount of average power needed to run the custom folded BNN contribute to its poor power efficiency score. This is in contrast to the custom folding having higher parallelism than the computer folding method. The *findBestFolding* was not used on the CIFAR-10 networks as they failed to pass the network full synthesis stage.

The performance of the computer folded networks is undermined by their low accuracy scores. Resulting in the custom folded network having a better scaled power efficiency than the computer folded ones, Figure 3.11. When testing the scaled

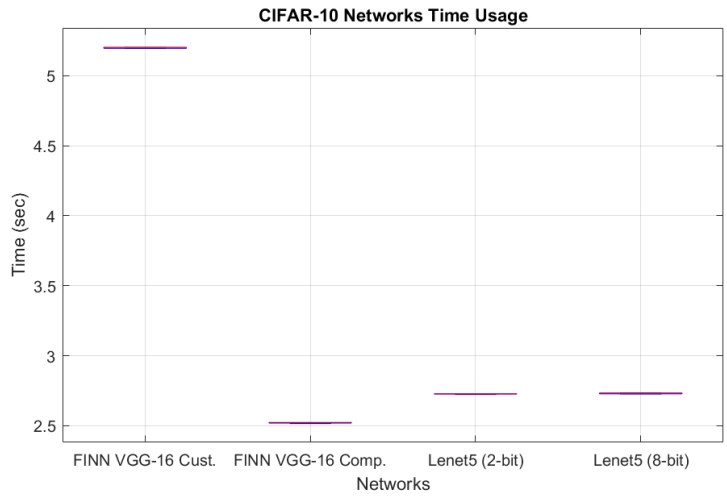

Figure 3.10: CIFAR-10 Network Time Usage

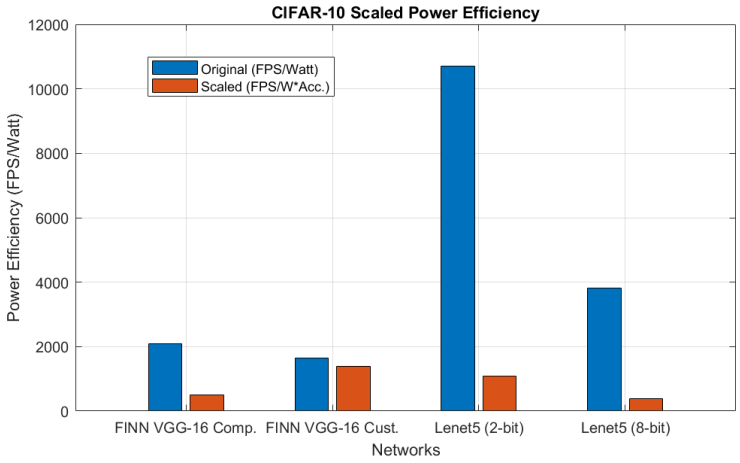

Figure 3.11: CIFAR-10 Scaled Power Efficiency

power efficiency of the custom folded BNN using the t-test with the 95% confidence interval the BNN is able to reject the null hypotheses for all the other networks. As a result the BNN is more power efficient then the other two networks when accuracy scaling is performed. No conclusion will be drawn on whether BNNs are more efficient than the alternatives for the CIFAR-10 data set. No network besides the FINN custom folding network was able to pass the development cycle.

Network	Layer	Cycles	Network	Layer	Cycles
TFC BNN[1]	MVA 0	64	FINN VGG-16[1]	MVA 6 & 7	32,768
TFC BNN[2]	MVA 0	28	FINN VGG-16[2]	MVA 1	12,544
TFC (2-bit))	MVA 0	49	-	-	-
SHL (2-bit)	Thres. 0 & MVA 0	392	Lenet5 (2-bit)	CIG 0	19,760
SHL (8-bit)	Thres. 0 & MVA 0	784	Lenet5 (8-bit)	CIG 0	19,760

[1]Custom Folding, [2]Computer Folding

Table 3.6: Bottleneck Layers

3.5 Hardware Requirements for Each Network

FINN provides methods for both obtaining the amount of resources required and estimating clock cycles needed for each layer. It is using these methods that a deeper understanding of the effects of the quantisation on the networks and the impacts of the binary operations can be assessed.

3.5.1 Network Time Review

As each of the layers in a network are run in parallel only the slowest layer will constitute a bottleneck. The listed cycles per frame in Table 3.6 only contain the layer that had the highest estimated cycles per frame and the amount of cycles listed. The FINN custom folding code was modified to return a cycles estimate. To retrieve estimated cycles per frame the netron models for the post folding stage of the synthesis were used.

To test the network on the test data set FINN provides a *validate.py* script. This script was modified to include time counters for all the various parts of the FPGA run time. Each part will accumulate its time as the script runs and provides the total time used for that part at the end of the run. The stages that are measured in the total time are image processing, buffer write, execute network, buffer read and processing of the results. For the MNIST data set a large percentage of time was spent writing the image stream onto the data buffers, Figure 3.12. The CIFAR-10 didn't experience the same percentage of time needed to write to the buffer instead it was the operation of the network that took the majority of the time. As a result when dealing with more complex networks writing to the FPGA will not be a bottle neck.

Using Equation 3.3 it is possible to further determine the impact of estimated layer

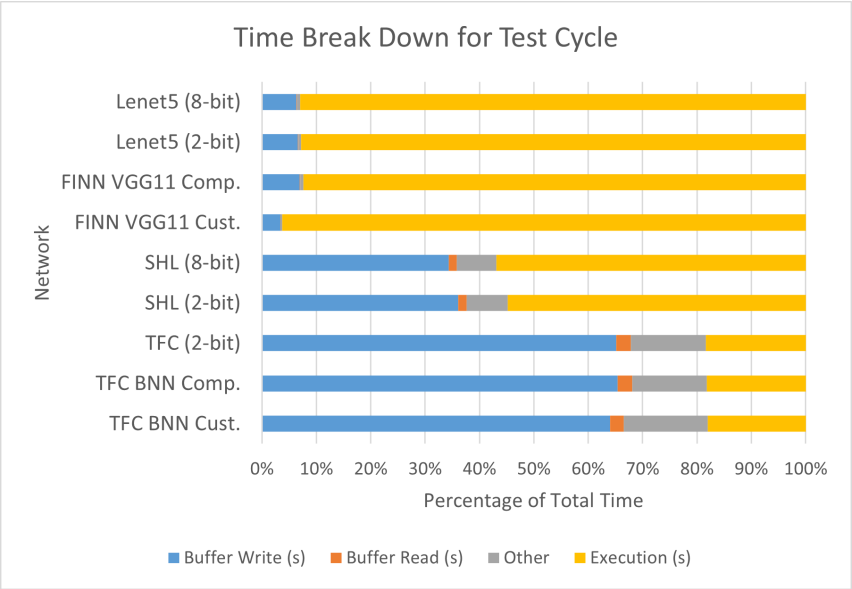

Figure 3.12: Time Break Down for Test Cycle

execution. When comparing the maximum cycle estimated time to the time it takes for the execution of the network on the FPGA on average 57% of the time is unaccounted for, listed as Other in Figure 3.13. Besides the SHL (8-bit) network the unaccounted for time is a large majority of the total execution time. Unlike the buffer write bottleneck the unaccounted for time does not become insignificant as the execution time increases. It is believed that this unaccounted for time is from the overhead of not having a continuous stream of images loaded onto the FPGA. As the verification script loads batches onto the FPGA separately parts of the network will be idle for some time. If the data stream could be continuously loaded and the FPGA executed continuously then the overhead would be minimised. The reduction of the overhead is critical as the throughput of the network is one of the key factors in improving the power efficiency.

$$Time(s) = \frac{Cycles}{Frame} * Num.\ Test\ Frames * Clock\ Period \tag{3.3}$$

3.5.2 Resource Requirements

A listed drawback of the BNN topology is the need to have an increased amount of parameters to offset the lack of information resulting from the binary weights and activation's. The amount of trainable parameters that each network has is listed in Table 3.7. The SHL (2-bit) network was constrained to the topology of the SHL (8-bit) resulting in an ineffective trinary network. The TFC BNN required

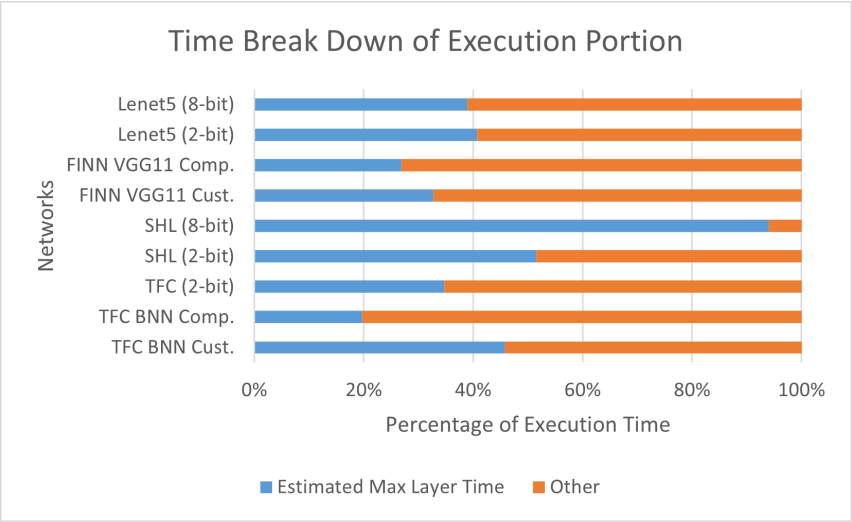

Figure 3.13: Time Break Down of Execution Portion

approximately 2.3 times the number of parameters than the 8-bit network and still suffers from a 2.83% decrease. The MNIST networks were small enough to fit onto the FPGA without compromising the stated network topology. The method of recording the resources used for each network doesn't list the total percentage of used Configurable Logic Blocks (CLBs). Instead it only lists summary information on the number of LUTs, FF, Shift Register LUT (SRL) and BRAM utilisation. As a result presented resource utilisation can only be used as a guide. Where there may appear to be left over resources in reality they may be none. Additionally, the synthesising of the network might not be able to use all the resources due to the networks topology. The layer may not be able to have its parallelism increased without overshooting the amount of resources available. Logic slices might not be fully utilised creating left over LUTs and FFs that cannot be used but are presented as free. The MNIST networks all passed the network synthesis stage. The networks then underwent the parallelism optimisation using the *findBestFolding* script to improve throughput. As a result of the increased parallelism the networks increased the amount of resources each one used. Figure 3.14 presents the percentage of HW utilisation when compared to the listed Z-7020 resources [34]. One of the stated advantages of the BNN is the amount of resources that it requires to be implemented are significantly lower than a 8-bit variant. The resource utilisation of the TFC networks all remain below the SHL variants even though they have 2.3 times the number of parameters compared to the SHL (8-bit) network. The

Network	# of Param.	Network	# of Param.
TFC BNN and 2-bit	59,394	FINN VGG-16	1,546,690
SHL (2-bit)	815,126	Lenet5 (2-bit)	21,636
SHL (8-bit)	25,452	Lenet5 (8-bit)	4,324

Table 3.7: Number of Trainable Parameters for Each Network

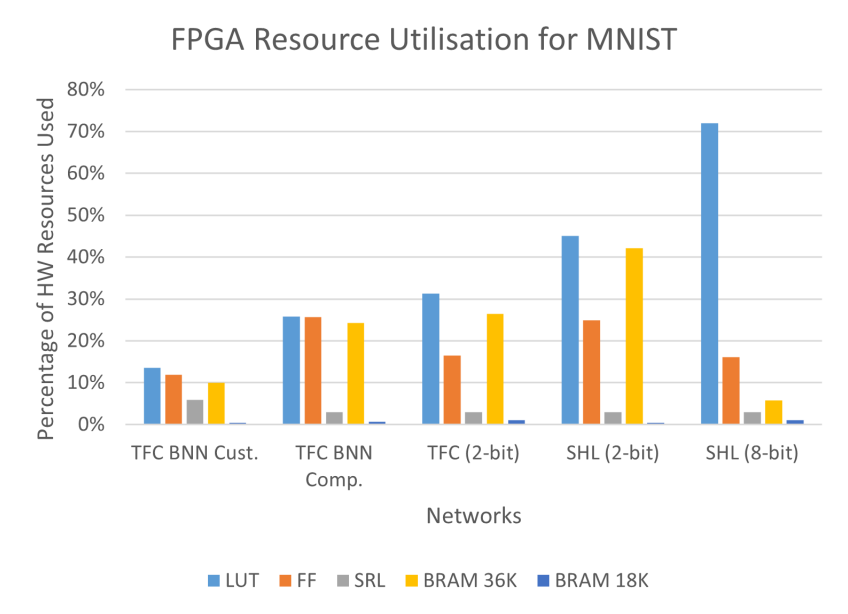

Figure 3.14: FPGA Resource Utilisation for MNIST Post *findBestFolding*

savings made in the lower HW requirements are able to be converted into higher parallelism. The relative increase in resources consumed before *findBestFolding* and after by each network, Figure 3.15 . The SHL 8-bit network did not experience any increase in HW uses as it was already at its optmial folding and could not be unfolded any further. The SHL 2-bit layer was able to increase its parallelism to a *target_cycles_per_frame* of 393, Table 3.5, roughly a third of the default 1000. This is represented by an increase of 58% and 49% for LUTs and FF. A full break down of the used resources for each network is presented in Figure C.3.

Notable is the large increase of FF and BRAM 36K used by the TFC BNN to reach the same *target_cycles_per_frame* when compared to the trinary variant. To assess the impact of the binary operations and limiting the resources savings provided by a reduction of bits two test network topologies will be synthesised. Both will contain a single FC stack with an input width of 36 for one and 784 for the other. Both will have an output width of ten and will be tested with all three types of quantisation. The results of this experiment will directly determine the impact of quantisation and

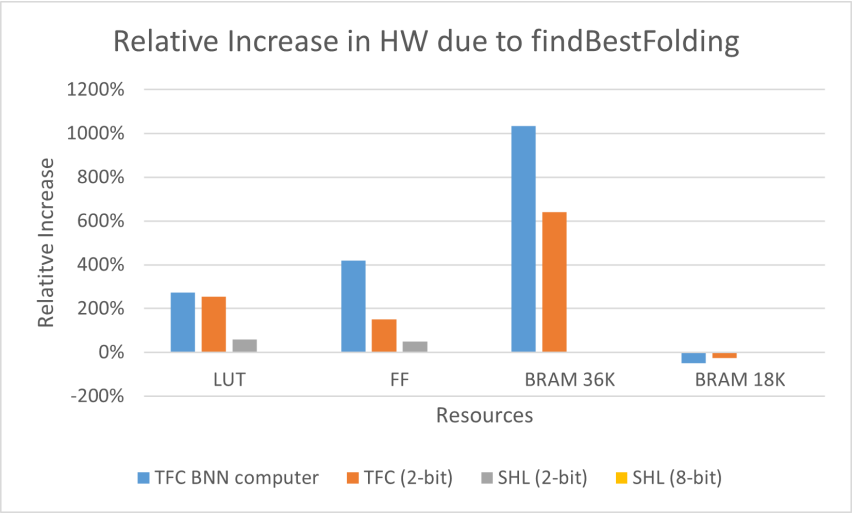

Figure 3.15: Relative Increase in HW Comparing Before and After *findBestFolding* for MNIST

binary operations. The folding will be handled using *SetFolding* with the default *target_cycles_per_frame* of 1,000. When comparing the two types of quantisation to the reference 8-bit, Table 3.8, the additional quantisation provided by lowering the amount of bits by 7 and 6 bits, respectively, results in the majority of the resource saving. Comparing the additional resources saved by the binary operations is achieved by using the relative difference between the reference of 2-bits and the binary as the new value, Table 3.9. When synthesising the network FINN creates three partitions. The first partition and last partitions, partition 0 and partition 2 respectively, use a constant amount of resources and are Representative of the input and output logic. The intermediate partition is representative of the network. Since these are small test networks the input and output logic will dilute the savings made by the BNN. As a result Table 3.9 includes the savings made by the network partition itself. It is found that binary operations save an additional 5.8% and 4.2% LUTs and FF respectively. This reduction in resources comes at the cost of binary networks not being able to perform as well in accuracy as their trinary networks. Whilst the SHL (2-bit) network required 32 times the number of parameters when compared to the 8-bit variant, Table 3.7, it was still able to achieve an accuracy on par with the 8-bit variant, Table 3.1. A SHL variant of the binary network was not developed as BNN require specialised training methods. The trinary network offering nearly the same HW benefits as the BNN but with the added benefit of being able to function similar to a 8-bit network is a big incentive to further explore

Quantisation	LUT %[1]	LUT %[2]	FF %[1]	FF %[2]
1-bit	41.57%	27.45%	81.82%	42.57%
2-bit	42.06%	27.93%	82.34%	43.08%
8-bit reference	5311	10022	3867	10294

[1] Width of 32, [2]Width of 784

Table 3.8: Resource Comparison Between 1 and 2-bit Quantisation with Reference to 8-bit

	LUT	FF
32 Bit Width Total	-1.2%	-0.6%
32 Bit Width Partition 1	-8.2%	-6.6%
784 Bit Width Total	-1.7%	-1.2%
784 Bit Width Partition 1	-5.8%	-4.2%

Table 3.9: Relative Reduction of Resources when Comparing 2-bit with 1-bit Quantisation

the differences between the trinary and binary networks.

The CIFAR-10 networks did not pass the final network synthesis test. As a result they did not have the same parallelism optimisation as the MNIST networks. Hence, they are graphed separately in Figure 3.16. These networks did not get the *findBestFolding* treatment. However, since they are already over the 1000 default *target_cycles_per_frame*, Table 3.6, they would not have experienced any improvements to performance.

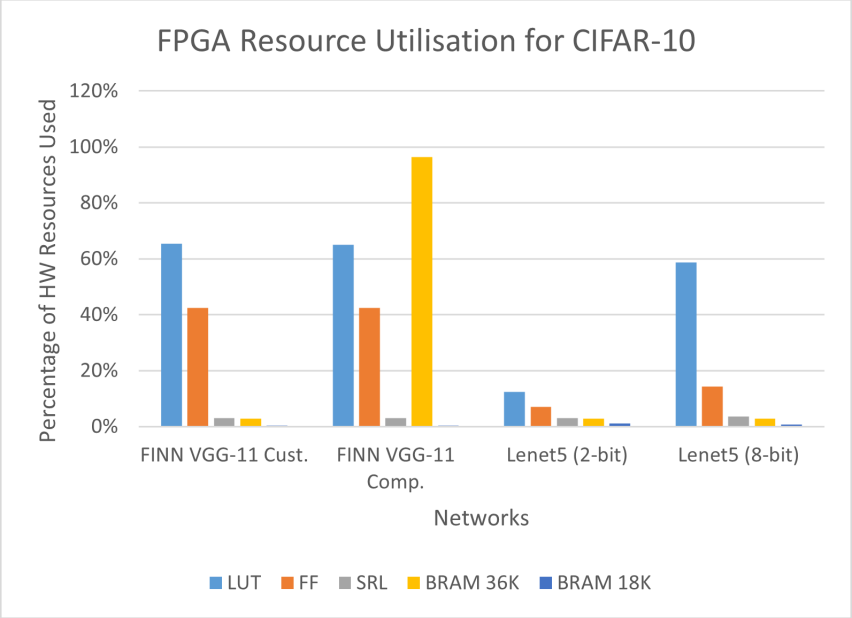

Figure 3.16: FPGA Resource Utilisation for CIFAR-10

Chapter 4

Conclusion

FPGAs provide a power efficient platform for hardware acceleration of NNs when compared to CPUs and GPUs. The work tested whether binary neural networks offered better power efficiency over there more commonly used 8-bit counterparts on FPGAs. The results showed that for the MNIST networks the BNN was able to easily surpass the power efficiency of the 8-bit equivalent. The BNN disadvantage of limited accuracy required the network to be scaled up significantly in the number of trainable parameters. However, the increase in parameters did not offset the benefits provided by the binary quantisation. It was determined that the throughput of a network plays a large part in the network's power efficiency. Networks that had a smaller HW footprint on the FPGA where able to decrease the amount of folding and thus increase their parallelism and throughput.

The CIFAR-10 networks where unable to be reliably tested in the work. The networks where able to run on the FPGA but in a broken state. Whilst attempts where made to fix the networks none were successful. As a result, no conclusions could be reliably drawn from the data. The only CIFAR-10 network that was able to run without suffering accuracy loss and thus provided accurate information was the FINN VGG-16 network. This network ran significantly slower than the other, broken, networks. It is unclear if the broken networks experienced higher throughput due to their broken state or not. Future work needs to be done to fix the CIFAR-10 networks in order to gain accurate information.

One of the limiting factors of the work was the lack of PMBus. The method used to measure the FPGA power usage was insufficient at providing high sampling rates. It also was unable to correct completely for board processes that ran in the background when the tests where being conducted. The recorded information demonstrated that the power is not consistent between different types of networks. Whether this is an artifact of the poor measurements or a tangible aspect of running different networks cannot be verified. Future work needs to be done using a PMBus enabled FPGA to determine how network topology changes the amount of power drawn.

4.1 BNNs and the Trinary Alternative

To test the impact of binary operations without the added benefit of a decrease in the bit width trinary networks where added to the work. The trinary network provides a adequate representation of the binary networks level of quantisation, with only a single additional bit. The work demonstrated that it is the quantisation that provides the most resource savings, not the binary operations. The impact of binary operations needs to be further examined in future work. They provided additional savings in terms of hardware requirements but the need to use a BNN to gain these savings can undermine their benefits. The MNIST data set used in this work doesn't adequately stress test the BNN. Future work should test larger networks on more complex data sets. The larger CIFAR-10 data set already started to stress test the BNN topology with accuracies dipping below 90% and still requiring three orders of magnitude more trainable parameters than the 8-bit Lenet5. It was the lack of FPGA HW resources that stopped larger versions of 8-bit networks being developed on the computer. Thus an adequate comparison between the 8-bit and BNN networks in terms of accuracy could not be made on the computer. Future work will need to take into consideration whether larger 8-bit networks are viable at all on FPGAs. Trinary networks show promise in their ability to be trained and use 8-bit network topologies, on a much greater scale, but also have the benefit of a reduced HW footprint.

4.2 FINN Work Flow

The creation and training of NN on brevitas to be compatible with FINN needs to be further investigated. The method of quantisation and the layer order continued to be a point of difficulty through out the project. The overall FINN work flow allowed NNs to be rapidly developed and deployed to FPGAs when compared to the alternative of hand coding. Without FINN the work would not have been able to be achieved to the same level. FINN presents a great opportunity for future work and research. Some research topics include: the impact of quantisation; determine optimal folding methods using different types of memory modes; extending FINN to other FPGAs, increasing the amount of compatible layers, and providing additional automation in the transformation stage.

The size of the FPGA was also a limiting factor. The Z-7020 series FPGA was unable to handle larger networks with the original Lenet5 implementation being to large. To test the impact of binary operations and quantisation on larger networks and still use the 8-bit reference will require a larger FPGA. The Pynq-Z2 board was selected for this work over larger boards due to its compatibility with FINN. If a larger FPGA is to be used in future work care must be taken to ensure that FINN would be compatible. FINN's compatibility with the PYNQ-Z2 board eliminated a point of complication. The FINN work flow was already significant in its complexity. Developing a method for FINN to talk to unsupported boards would have been infeasible for this thesis.

4.3 Closing Remarks

The viability of FPGAs as hardware accelerators for NNs has increased with the development of tools like FINN. The topology and design of networks needs to be optimised before FPGAs can become a dominate accelerator in the world of computer vision. BNNs provide increased power efficiency over the traditional 8-bit networks. But this work has shown that trinary networks might provide a better alternative to both.

References

[1] Stylianos I Venieris and Christos-Savvas Bouganis. fpgaconvnet: A framework for mapping convolutional neural networks on fpgas. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 40–47. IEEE, 2016.

[2] Sparsh Mittal. A survey of fpga-based accelerators for convolutional neural networks. *Neural computing and applications*, 32(4):1109–1139, 2020.

[3] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 65–74, 2017.

[4] Vimal Kumar and Uma Maheshwari. Improving the usability and scalability of finn, a dnn compiler for fpgas, 2021.

[5] Ahmed Ghazi Blaiech, Khaled Ben Khalifa, Carlos Valderrama, Marcelo AC Fernandes, and Mohamed Hedi Bedoui. A survey and taxonomy of fpga-based deep learning accelerators. *Journal of Systems Architecture*, 98:331–345, 2019.

[6] Fumio Hamanaka, Takashi Odan, Kenji Kise, and Thiem Van Chu. An exploration of state-of-the-art automation frameworks for fpga-based dnn acceleration. *IEEE Access*, 2023.

[7] Ying Wang, Jie Xu, Yinhe Han, Huawei Li, and Xiaowei Li. Deepburning: Automatic generation of fpga-based learning accelerators for the neural network family. In *Proceedings of the 53rd Annual Design Automation Conference*, pages 1–6, 2016.

[8] Shuang Liang, Shouyi Yin, Leibo Liu, Wayne Luk, and Shaojun Wei. Fp-bnn: Binarized neural network on fpga. *Neurocomputing*, 275:1072–1086, 2018.

[9] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

[10] Taylor Simons and Dah-Jye Lee. A review of binarized neural networks. *Electronics*, 8(6): 661, 2019.

[11] Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 15–24, 2017.

[12] Duy Thanh Nguyen, Tuan Nghia Nguyen, Hyun Kim, and Hyuk-Jae Lee. A high-throughput and power-efficient fpga implementation of yolo cnn for object detection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(8):1861–1873, 2019.

[13] Ruzica Jevtic and Carlos Carreras. Power measurement methodology for fpga devices. *IEEE Transactions on Instrumentation and Measurement*, 60(1):237–247, 2010.

[14] Li Shang, Alireza S Kaviani, and Kusuma Bathala. Dynamic power consumption in virtex™-ii fpga family. In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pages 157–164, 2002.

[15] Xilinx. Xilinx power estimator, 8 2019. URL https://www.xilinx.com/products/technology/power/xpe.html.

[16] Raju Machupalli, Masum Hossain, and Mrinal Mandal. Review of asic accelerators for deep neural network. *Microprocessors and Microsystems*, 89:104441, 2022.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[18] Mairin Kroes. Optimizing memory mapping for dataflow inference accelerators: Efficient memory utilization on fpgas, 2020.

[19] Mirza Mrahorovic. Finn dataflow accelerator examples, 05 2023. URL https://github.com/Xilinx/finn-examples.

[20] Yaman Umuroglu and Magnus Jahre. Streamlined deployment for quantized neural networks. *arXiv preprint arXiv:1709.04060*, 2017.

[21] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV*, pages 525–542. Springer, 2016.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[23] Tim Dettmers. 8-bit approximations for parallelism in deep learning. *arXiv preprint arXiv:1511.04561*, 2015.

[24] Yao Fu, Ephrem Wu, Ashish Sirasao, Sedny Attia, Kamran Khan, and Ralph Wittig. Deep learning with int8 optimization on xilinx devices. *White Paper*, 2016.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Alaa M Salman, Ahmed S Tulan, Rana Y Mohamed, Michael H Zakhari, and Hassan Mostafa. Comparative study of hardware accelerated convolution neural network on pynq board. In *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 578–582. IEEE, 2020.

[28] Alejandro Baldominos, Yago Saez, and Pedro Isasi. A survey of handwritten character recognition with mnist and emnist. *Applied Sciences*, 9(15):3169, 2019.

[29] Alessandro Pappalardo. Xilinx/brevitas, 2023. URL https://doi.org/10.5281/zenodo.3333552.

[30] Sebastian Raschka. deeplearning-models, Feburary 2023. URL https://github.com/rasbt/deeplearning-models/tree/master.

[31] Rasim Caner Çalik and M Fatih Demirci. Cifar-10 image classification with convolutional neural networks for embedded systems. In *2018 IEEE/ACS 15th International Conference on*

Computer Systems and Applications (AICCSA), pages 1–2. IEEE, 2018.

[32] Alessandro Pappalardo. Xilinx/brevitas, 2023. URL `https://github.com/Xilinx/brevitas/tree/super_res_r1/src/brevitas_examples/bnn_pynq`.

[33] Inc. Advanced Micro Devices. *Anatomy of a Quantizer*, 2023. URL `https://xilinx.github.io/brevitas/tutorials/anatomy_quantizer.html`.

[34] Xilinx. *Zynq-7000 SoC Data Sheet: Overview*, 06 2018. URL `https://www.mouser.com/datasheet/2/903/ds190-Zynq-7000-Overview-1595492.pdf`.

[35] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66, 2015.

[36] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia computer science*, 132:377–384, 2018.

[37] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[38] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

[39] Andrew Boutros, Sadegh Yazdanshenas, and Vaughn Betz. You cannot improve what you do not measure: Fpga vs. asic efficiency gaps for convolutional neural network inference. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11(3):1–23, 2018.