Deep Neural Network (DNN) Design: The Utilization of Approximate Computing and Practical Considerations for Accuracy Evaluation

by

Issam Hammad

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
July 2021

TO HOLLY, LANNAH AND RYAN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTACT

Approximate computing is emerging as a viable way to achieve significant performance enhancement in terms of power, speed, and area for system on chip (SoC) designs. Utilizing approximate computing in the design of deep neural networks (DNNs) can significantly reduce the system's power, delay, and area at a cost of a tolerable drop in accuracy. This thesis demonstrates how approximate computing methods such as approximate multiplication, low quantization, and shared neural networks can achieve these performance enhancements in DNN designs. In terms of approximate multipliers which are the primary focus of the thesis, a study on the impact of approximate multipliers on the inference accuracy of convolutional neural networks (CNNs) is presented. Additionally, an efficient hybrid training approach using both exact and approximate multipliers is proposed. Most importantly, the thesis introduces the new concept of boosting CNN multiplication performance using a precision prediction preprocessor that controls approximate multipliers with various precisions. Another important research contribution of this thesis is studying practical considerations for accuracy evaluation of sensor-based machine learning and deep learning designs. Certain aspects can negatively impact the system's accuracy in production. These aspects are not usually considered when evaluating and comparing models' accuracy during development and prototyping. Examples include accuracy loss due to the component's variable thermal noise, component failure or partial failure, and analog-to-digital converter (ADC) quantization error. Finally, the thesis presents the new concept of utilizing machine learning for person identification through physical activity. This research finding demonstrates that machine learning can be applied not only for the identification of physical activities but also for the identification of the activity performer as well. Based on this finding, a novel multi-label shared deep neural network (DNN) to identify both the physical activity and the activity performer simultaneously is proposed.

LIST OF ABBREVIATIONS USED

ADC	analog-to-digital converter
ASIC	application-specific integrated circuit
CIFAR	Canadian Institute for Advanced Research
CPU	central processing unit
CNN	convolutional neural networks
DNN	deep neural network
DOF	degrees of freedom
DSM	dynamic segment method
DTC	decision tree classifier
ENOB	effective number of bits
GANs	generative adversarial networks
GNB	Gaussian Naïve Bayes
GPU	graphics processing unit
HDL	hardware description language
IoT	Internet of Things
KNN	k-nearest neighbors
LOD	leading one detector
LSB	least significant bit
LSTM	Long short-term memory
MCCV	Monte Carlo cross-validation
MRE	mean relative error
MAC	multiply-and-accumulate
MUX	Multiplexer
PCA	principal component analysis
PE	processing element
pJ	Picojoules
PDF	probability density function
RFC	random forest classifier
ReLU	rectified linear unit
RNN	recurrent neural network
SD	standard deviation
SNR	signal-to-noise ratio
SNDR	signal to noise and distortion ratio
SSM	static segment method
SGD	stochastic gradient descent
SoC	system on chip
TPU	tensor processing unit
UCI	University of California Irvine

AKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr. Kamal El-Sankary for his guidance, encouragement, and support during my Ph.D. program.

Also, I would like to thank Dr. Jason Gu and Dr. Guy Kember for being part of my supervisory committee, Moreover, I would like to thank Dr. Lihong Zhang from Memorial University for serving as the external Ph.D. examiner.

I would also like to express my gratitude to the department faculty members and staff and to the university employees who worked hard to enable an efficient and fast transition to online learning during the COVID-19 pandemic.

Most of all, I would like to express my special thanks and appreciation to my wife Holly Hammad for her support and encouragement throughout my Ph.D. program.

CHAPTER 1 INTRODUCTION

With the recent boom in computational power, deep learning [1] has become a viable method to solve problems in areas such as image recognition, real-time video analysis, self-driving cars, and biomedical engineering. In simple words, deep learning can be defined as the process of training a deep neural network to act as a function approximator for a specific problem. Computer scientists and engineers have proposed several successful deep learning models that are being used in various industrial applications nowadays. Most of these models rely on general-purpose processors such as a central processing unit (CPUs) or a graphics processing unit (GPUs) for deep learning training and inference.

Using a general-purpose processor for a computationally heavy process such as deep learning is slower and requires a higher power compared to using an application-specific integrated circuit (ASIC). With the continuous increase in bandwidth and data, the demand for systems that can operate with lower power and at a higher speed is increasing rapidly, especially for Internet of Things (IoT) battery-operated devices and in large servers that construct the backbone of cloud systems. In the last decade, the deep learning research community focused on building new models that achieve better prediction accuracy. For image recognition, several powerful models were proposed for using Convolutional Neural Networks (CNN) examples include AlexNet [2], VGGNet [3], GoogleNet [4], DenseNet [5], and Xception [6]. These CNNs have revolutionized the field of image classification and computer vision in general.

Despite the rapid increase in deep neural network (DNN) model development, the focus on providing optimized hardware solutions that can boost the performance of these models started only in the last five years. One example is Google's tensor processing unit (TPU) [7] which supports the backend of deep learning on Google Cloud. Other examples include the deep learning ASIC designs which are proposed in [8-11]. The primary focus of designs [7-9] is to achieve high parallelism and near memory computation to achieve higher performance in terms of power, area, and speed.

Besides high parallelism, reducing energy cost and improving the performance in terms of speed, power, and area can be achieved by utilizing approximate computing which is the focus of this thesis. Using approximate computing as a method to achieve energy-efficient designs was previously discussed in [12-16]. Approximate computing can achieve significant performance enhancement at a cost of certain inaccuracy in the output. However, for systems such as a DNN, what matters is the accuracy impact on the entire network and not on each neuron within the network.

1.1 Thesis Objectives

The primary objective of this thesis is to utilize approximate computing methods such as approximate multiplication, low quantization, and shared neural networks to achieve significant performance enhancement in deep learning hardware designs. The thesis aims to study the impact of approximate computing on the accuracy of DNNs and to propose efficient methods where approximate computing methods can be utilized in ways that can achieve the best trade-off between performance gains in terms of power, area, and speed and the cost in terms of accuracy loss. Additionally, the thesis aims to define practical accuracy validation methods that can be applied in the production of sensor-based machine learning and deep learning solutions. This includes studying the accuracy loss due to the component's variable thermal noise, component failure or partial failure, and analog-to-digital converter (ADC) quantization error. These aspects are not usually considered when evaluating and comparing sensor-based models' accuracy during development and prototyping.

1.2 Thesis Contributions

The thesis presents a combination of three published [17-19] journal papers and two published international conference papers [20-21].

The published research papers [17][18][20] focused on utilizing approximate multipliers in the design of DNNs. The research objective is to achieve significant performance gains

in terms of improving power, area, and speed while having a minimal cost in terms of accuracy loss.

The research journal paper [17] presented an early research study on the impact of approximate multipliers on the inference accuracy of the VGG deep neural network. The paper demonstrated that utilizing approximate multipliers can achieve a significant performance enhancement while having a negligible impact on the inference accuracy. The simulations in [17] demonstrated that implementing a hardware solution for DNNs does not necessarily require high precision floating-point multiplications which were used in models such as [2-6]. DNNs are function approximators, therefore, using approximate multipliers to implement these function approximators is an efficient way to significantly improve the hardware performance while having a minimal impact on the accuracy. Additionally, the work in [17] presented that the impact of approximate multipliers on the network varies per layer, deeper layers are impacted less by the usage of approximate multipliers, therefore a hybrid design can be adopted where approximate multipliers are utilized only in these deep layers allowing for significant performance gains while having a negligible accuracy loss.

The work in [20] was an expansion to the work presented in [17] to include the training stage. The work in [20] presented by simulation how approximate multipliers can be utilized to enhance the training performance of convolutional neural networks (CNNs). The paper proposed a new hybrid training method. Using this method, the training can start using approximate multipliers, then it can switch to exact multipliers during the last few epochs. This attains the performance benefits of the approximate multipliers in terms of speed, power, and area for a large portion of the training stage. On the other hand, the negative impact on the accuracy is diminished by using the exact multipliers for the last epochs of the training.

The journal [18] proposed a new method that allows for the utilization of approximate multipliers with various precision simultaneously using a preprocessing precision controller. The proposed controller is a tiny two-class CNN that determines the required

approximate multiplier precision for the input image. The controller can be utilized in a system that contains multiple approximate-multiplier based CNN inference accelerators with various precisions, or in a single CNN inference accelerator built with precision reconfigurable approximate multipliers. The controller's objective is to maximize the overall performance gains by maximizing the usage of lower precision approximate multipliers whenever this does not cause an additional accuracy loss. This augments the performance of the CNN inference by allowing for the utilization of existing low precision and high precision approximate multiplier hybridly. The journal also proposed a new reconfigurable approximate multiplier to utilize this concept in single-core designs.

The published journal [19] presented a study on the practical considerations for accuracy evaluation in sensor-based machine learning and deep learning. To select an appropriate machine learning model for production in a sensor-based application, several practical aspects should be considered beyond the basic train/test split when comparing different models' performance. These aspects include studying the impact of thermal noise on the inference accuracy, finding the adequate level of quantization, and evaluating model accuracy tolerance to sensor failure. By working in the industry, I have seen the importance of studying such practical considerations and therefore it is crucial to select a machine learning model that is resilient to these untested variables. It is common that machine learning datasets are built using the same group of sensors. Therefore, even with the train/test split, the model is generalizing only to the sensors used in building the dataset which can lead to a significant unexpected accuracy drop in production.

The conference paper [21] proposed the new concept of person identification through physical activity. The presented research contribution opens the door for sharing wearable technologies where the user profile can be detected automatically from their physical activity. The paper also proposed a multi-label shared DNN for simultaneous classification of the physical activity and the activity performer.

This thesis excludes two additional two journal papers [22-23] and one conference paper [24] that I have worked on during the period in which I commenced my Ph.D. degree. These research articles are outside of the thesis scope.

1.3 Thesis Outline

The thesis is organized as follows with each chapter representing a published journal or conference paper:

- Chapter 2 presents a study on the impact of approximate multipliers on the VGG deep learning network. This work was published at the IEEE Access in October 2018 [17].

- Chapter 3 proposes a method for deep learning training with simulated approximate multipliers. This work was presented at the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO). It also received the best paper in AI award at the conference [20].

- Chapter 4 proposes a method for CNN inference using a preprocessing precision controller and approximate multipliers with various precisions. This work was published at the IEEE Access in January 2021 [18].

- Chapter 5 proposes practical considerations for accuracy evaluation in sensor-based machine learning and deep learning. This work was published at Sensors in August 2019 [19].

- Chapter 6 presents the concept of using machine learning for person identification through physical activities. This work was presented at the 2020 IEEE International Symposium on Circuits and Systems (ISCAS) [21].

- Chapter 7 presents the research conclusion and future work.

CHAPTER 2 IMPACT OF APPROXIMATE MULTIPLIERS ON VGG DEEP LEARNING NETWORK

Issam Hammad and Kamal El-Sankary

2.1 Abstract

This paper presents a study on the applicability of using approximate multipliers to enhance the performance of VGGNet deep learning network. Approximate multipliers are known to have reduced power, area, and delay with the cost of an inaccuracy in output. Improving the performance of the VGGNet in terms of power, area, and speed can be achieved by replacing exact multipliers with approximate multipliers as demonstrated in this study. The simulation results show that approximate multiplication has a very little impact on the accuracy of VGGNet. However, using approximate multipliers can achieve significant performance gains. The simulation was completed using different generated error matrices that mimic the inaccuracy that approximate multipliers introduce to the data. The impact of various ranges of the mean relative error (MRE) and the standard deviation (SD) was tested. The well-known datasets CIFAR-10 and CIFAR-100 were used for testing the network's classification accuracy. The impact on the accuracy was assessed by simulating approximate multiplication in all the layers in the first set of tests, and in selective layers in the second set of tests. Using approximate multipliers in all the layers leads to very little impact on the network's accuracy. Additionally, an alternative approach is to use a hybrid of exact and approximate multipliers. In the hybrid approach, 39.14% of the deeper layer's multiplications can be approximate while having a negligible impact on the network's accuracy.

2.2 Introduction

Deep learning using convolutional neural networks (CNNs) has gained increased momentum in recent years. Image classification is one of the primary applications for deep learning using CNN. Several successful CNN architectures were proposed in the literature. One of the most used architectures is the VGGNet proposed by Karen Simonyan and Andrew Zisserman [3]. One of the widely used VGG configurations is the VGGNet-16 (referred to as configuration D in [3]), which consists of 13 convolutional layers, and 3 fully connected layers with max pooling applied between the layers. VGGNet has a

uniform architecture and uses 3x3 convolutions. Figure 2.1 depicts the network architecture as proposed in [3] including the number of channels in each stage.

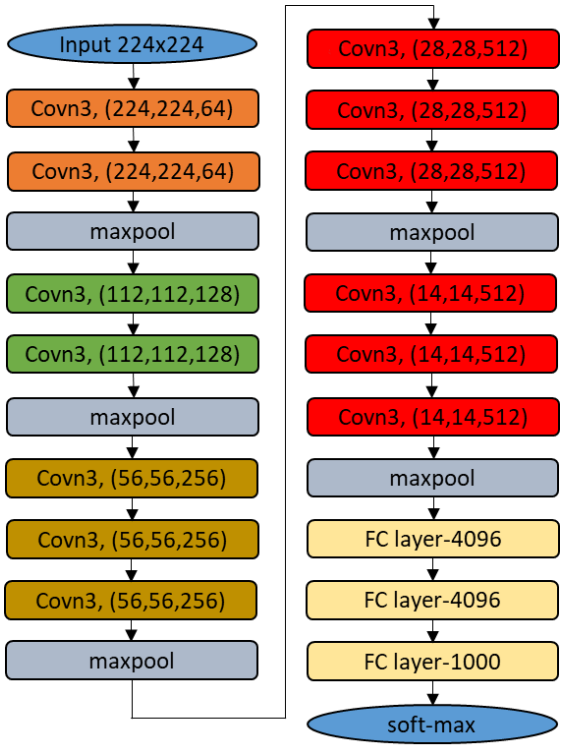

Figure 2.1: VGGNET-16 as proposed by [3]

According to [3], the VGGNet-16 has 138 million parameters. Since the convolution in a CNN is completed via multiplication and addition, any improvement on the performance or the cost of multiplication will have a significant impact on the overall performance and cost of the entire network. According to [25], the VGGNet-16 required 15.5G multiply-and-accumulates (MACs) operations to complete the classification of one image.

The concept of approximate computing has emerged in recent years to increase systems performance and power efficiency. Usage of approximate computing is promoted for media related systems due to its ability to tolerate error. One of the applications of approximate computing is the approximate multiplier. Compared to an exact multiplier, the approximate multiplier has a reduced power, area, and delay. However, it has a cost of inaccuracy which is usually defined by metrics such as the MRE and the SD.

The research objective is to demonstrate that approximate multipliers can be used to optimize the performance of VGGNet in terms of power, area, and delay. This work assesses the impact of various approximate multiplication error ranges on the VGGNet accuracy and identifies the network layers that are the least impacted by approximate multiplication. Additionally, the research provides a baseline for researchers to apply the proposed simulation methods to explore the impact of approximate computing on various deep learning architectures.

VGGNet was selected for this study as it is one of the most popular deep learning architectures for image classification. Additionally, it has a very uniform architecture with 3x3 convolutions and with a modest number of layers comparing to other popular architectures. These features enable a homogeneous evaluation of the impact of approximate multiplication on the network layers.

The focus of this research is on the optimization of pre-trained networks which exclude the training phase. Several systems rely on pre-trained weights especially in low power applications. For these systems the training is done on a central server, then the weights are downloaded to many client devices which usually have limited hardware resources. One example is the internet of things (IoT) hardware accelerator presented in [26] which uses pre-trained weights.

In this paper, a simulation for the impact of approximate multipliers on the accuracy of VGGNet is presented. The simulation included testing the impact of approximate multiplication on all the layers in the first set of tests and on selective layers in the second set of tests. The simulation results of both approaches show approximate multipliers can be used to achieve significant performance gains with a very minimal cost of added accuracy error.

This paper is organized as follows: Section 2.3 provides a background on the approximate multiplier. Section 2.4 demonstrates the concept of approximate multipliers simulation by adding error matrices to the network's layers. Section 2.5 presents the results of simulating

approximate multiplication in all VGGNet layers. Section 2.6 focuses on the impact of applying approximate multiplication in selective network layers and proposes the hybrid approach. Section 2.7 summarizes the research conclusion.

2.3 Background on The Approximate Multiplier

Approximate multipliers can be utilized in applications that are tolerant to inaccuracy. Several different approximate multiplier designs were recently proposed as a replacement for the exact multiplier such as [27-32]. Approximate multipliers can lower the design cost in terms of power, delay and chip size, with the cost of having a certain calculation error. The mean relative error (MRE) is used along with other metrics such as the SD to assess inaccuracy of approximate multipliers. The MRE is the primary common metric used to evaluate the error in approximate multipliers, this can be found in the approximate multiplier publications [27-32]. The MRE is defined as:

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Yi' - Yi|}{|Yi|} \tag{2.1}$$

In equation (2.1), Yi is the exact value while Yi' is the approximate value. One example of an approximate multiplier is the 16-bit design proposed by S. Venkatachalam et. al. [27]. The multiplier in [27] has achieved power, area and delay savings of 72%, 56%, and 31% respectively while introducing an MRE of 7.6%. Another example is the approximate multiplier in [30] which introduces an MRE of 1.47% while having a reduction of 59%, 50%, and 47% in the power, area, and delay. Several other implementations for the approximate multiplier can be found in [28-29] and [31-33]. The design in [33] showed that a decreased area and power in an approximate multiplier can be achieved by introducing a higher error. Therefore, this allows for a flexibility in the design by balancing the trade-off between the accuracy loss and the achieved savings in power and area.

2.4 Approximate Multiplier Simulation

This section presents the concept of approximate multiplier error simulation for deep learning networks. To test the impact of approximate multipliers on the accuracy of the network, the multiplication accuracy should contain a certain MRE that an approximate multiplier would have introduced if it was used instead of an exact multiplier. To test the impact of approximate multiplication on the accuracy of the VGGNet, a pre-trained VGGNet was used [34]. This network was built using the python based deep learning platform Keras [35]. This network was modified to add an error matrix prior to completing the convolution at certain layers of the network, depending on whether they are part of the test case or not. The error matrix was applied through element-wise multiplication with the layer input *Y*. The error matrix was also applied to the fully connected layers in some test cases. In equation (2.2), *Y'* is the modified layer input after applying a certain MRE.

$$Y' = Y \odot E \tag{2.2}$$

Where $\odot$ is element-wise multiplication operator. The matrix *E* is tuned to apply the required MRE value. For example, to simulate an MRE value of 2.5% using a Uniform probability density function (PDF), the matrix *E* will contain random values ranging from (0.95-1.05) to simulate the impact of approximate multipliers on the network's accuracy. Several test cases for Uniform and Gaussian PDFs error matrices were applied. These test cases are generic to cover the characteristics of various approximate multipliers in the literature. The MRE and the PDFs of these simulated test cases can be mapped to these published approximate multipliers designs as demonstrated in the next section. Figure 2.2 shows the histogram of a sample of Uniform and Gaussian error matrices used for one test image in the first layer of the network.

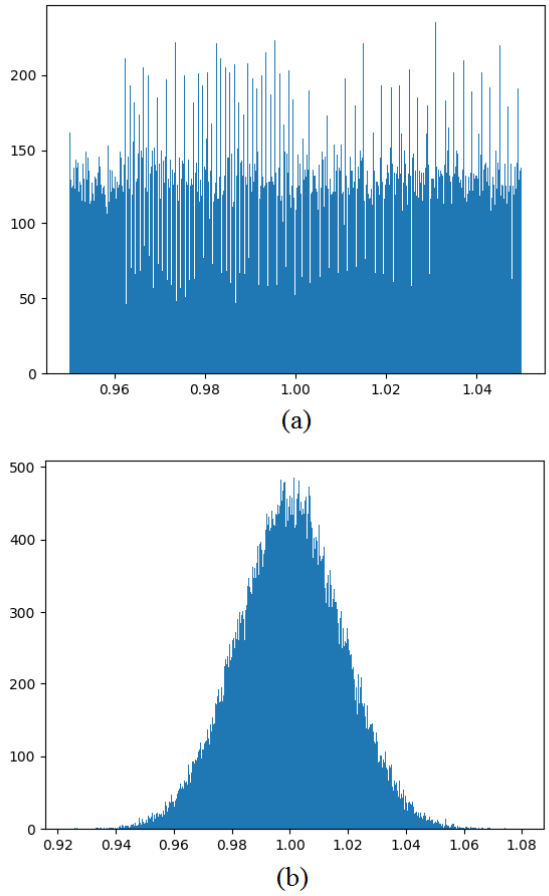

Figure 2.2: A histogram for an error matrix using (500 bins).
(a) a sample Uniform error matrix (MRE=~2.5%).
(b) a sample Gaussian error matrix (MRE=~1.4%).

In this simulation, each test case error range was divided into 10000 unique values. The error matrix E was constructed randomly from these values using Uniform or Gaussian PDFs. Each error matrix for a layer in a test case was generated using a different seed. All the tests were executed using 'float16' precision.

To test the impact on the accuracy, CIFAR-10 and CIFAR-100 datasets were used. CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. CIFAR-10 has 50000 training images and 10000 test images [36]. CIFAR-100 has 100 classes containing 600 images each divided as 500 training images and 100 testing images

per class. As CIFAR-10 and CIFAR-100 datasets are used, the network architecture in this section is slightly different than the original VGGNet-16 as per [3]. The network architecture contains changes that were proposed in [37] which handled the design of VGGNet for CIFAR-10 and CIFAR-100 datasets. The network design in [37] is tailored for inputs with size 32x32 instead of 224x224, consequently, the dimensions of this network's layers are smaller than the original VGGNet as proposed in [3]. The network changes also include using 2 fully connected layers instead of 3 fully connected layers, and changes in other settings such as batch normalization and dropout to reduce overfitting [37]. Figure 2.3 demonstrates this modified VGGNet that is used for the simulation. As the focus of this simulation is to assess the impact of approximate multipliers on a pre-trained network, the error matrices were applied to the test images only.

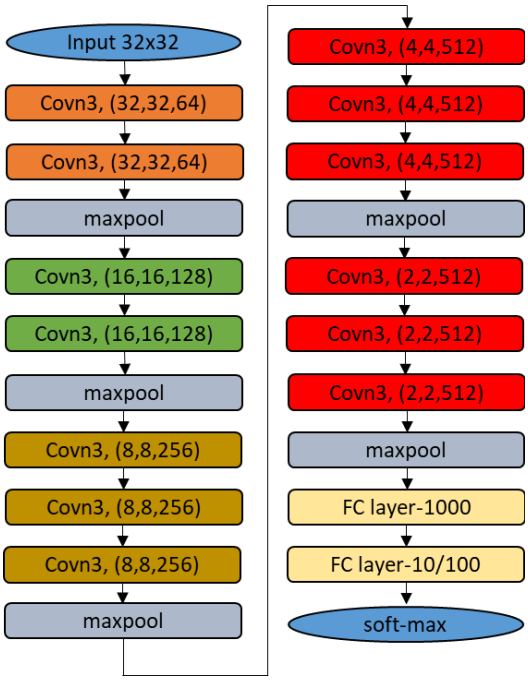

Figure 2.3: Modified VGGNET as per [37]

To evaluate the impact of approximate multipliers, two set of tests were executed. The first set of tests are the "MRE Tests" while the second tests are "Layer Impact Tests". In the "MRE Tests", the error matrix was added to each layer of the network. In each test case, a specific MRE value was simulated to assess its impact on the network accuracy. In the "Layer

Impact Tests", the error matrix was applied only to a selective group of layers in each test case. This was done to evaluate the impact of having approximate multiplication in these particular layers. The next two sections will discuss the simulation results for both the "MRE Tests" and the "Layer Impact Tests". During the simulation, each test case consisted of 100 loops, in each loop the entire dataset (CIFAR-10 or CIFAR-100) was applied. In each test case, an error matrix with approximately the same MRE and PDF was applied to the tested layers of the network. However, the error matrices had different seeds in each layer of every loop. The tables in this section list the average network error of the 100 loops. Additionally, due to having different random seeds for the error matrices the listed MRE and SD values in all the tables are approximate.

2.5 MRE Tests

Several Uniform and Gaussian error matrices with different MRE values were simulated to assess their overall impact on the network's accuracy. Table 2.1 shows the results of the simulation using CIFAR-10 dataset while Table 2.2 shows the results of the simulation using CIFAR-100 dataset. For each test case, Table 2.1 and Table 2.2 list the MRE, the SD, the network error rate, and the error difference compared to an exact multiplier. The network error rate in Table 2.1 and Table 2.2 refers to the percentage of incorrect image classifications by the network. The error difference reflects the additional error which resulted from using an approximate multiplier instead of an exact multiplier. In the used architecture for this simulation an execution with an exact multiplier using CIFAR-10 dataset results in a network error rate of 6.4%, while the error is 29.51% for CIFAR-100 dataset. These numbers will be used as a baseline to assess the impact of the approximate multiplier. The network error rate can have minor variations based on the training settings and the used hyperparameters. According to [37], the human rater for CIFAR-10 is 6%, therefore, the used baseline rate is very close to the human classification error rate.

As can be seen from Table 2.1 and Table 2.2, the error difference resulting from the added MRE is very minimal and can be considered negligible especially for lower MRE cases. Similar or exceeding error differences can be a result of a tweak to one of the

hyperparameters during network training. Figure 2.4 illustrates the approximate relation between the increase in MRE and the increase in the error difference. While all tests in Tables I and II were executed using float-16 datatype, float-32 datatype was also simulated using a subset of tests from Table 2.1 and Table 2.2. Using float-32 gave very similar results to float-16, therefore, these simulation results are applicable to both 16 bits and 32 bits approximate multipliers.

The error matrices used in Table 2.1 and Table 2.2 are generic to cover a broad spectrum of possible errors resulting from the usage of approximate multipliers. However, these test cases can be mapped to the reported performances of proposed approximate multipliers in the literature as Table 2.3 presents. Table 2.3 lists the reported performance enhancements for various approximate multipliers in comparison to exact multipliers. By mapping the simulation results from Table 2.1 and Table 2.2 to the approximate multipliers performance result in Table 2.3, the advantage of using approximate multipliers in VGGNet can be clearly seen. An example is the approximate multiplier DRUM [30], this multiplier has a Gaussian error with MRE of 1.47% and SD of 1.803%. By replacing the VGG's exact multipliers by DRUM's approximate multipliers, the multiplication cost can have approximate savings in power, area, and delay of 59%, 50%, and 47% respectively. One CIFAR image classification using the modified VGGNet as per [37] requires 313.41M MACs. According to the Gaussian simulation results in Table 2.1 and Table 2.2, this replacement will cause an approximate additional network error of only 0.038% using CIFAR-10 and 0.064% using CIFAR-100. This is based on the closest simulation test case with MRE of 1.4% and SD of 1.8%. This additional network error is minimal considering the achieved reduction in power and area and the significant increase in speed. The Speed increase is very critical for deep learning applications, researchers are vigorously trying to speed up deep learning training and testing. More examples can be seen by mapping the performance enhancements of the approximate multipliers in [27-28] and [31-32] to the closest simulated test in Table 2.1 and Table 2.2. Note that (G) and (U) in Table 2.3 refers to Gaussian and Uniform distribution, respectively.

Table 2.1: Test results using CIFAR-10

Matrix *E* Type	MRE	SD	Network Error Rate	Error Diff. from Exact Multiplier
No Error	0%	N/A	6.4%	0%
Uniform	~0.75%	~0.087%	6.422%	+0.012%
Uniform	~1.25%	~1.443%	6.43%	+0.03%
Uniform	~2.5%	~2.887%	6.495%	+0.095%
Uniform	~5%	~5.774%	6.598%	+0.198%
Uniform	~7.5%	~8.66%	6.828%	+0.428%
Uniform	~10%	~11.55%	7.358%	+0.958%
Gaussian	~0.6%	~0.75%	6.423%	+0.023%
Gaussian	~1.4%	~1.80%	6.438%	+0.038%
Gaussian	~2.4%	~3.0%	6.477%	+0.077%
Gaussian	~3.6%	~4.5%	6.53%	+0.13%
Gaussian	~4.8%	~6.0%	6.599%	+0.199%

Table 2.2: Test results using CIFAR-100

Matrix *E* Type	MRE	SD (σ)	Network Error Rate	Error Diff. from Exact Multiplier
No Error	0%	N/A	29.51%	0%
Uniform	~0.75%	~0.087%	29.556%	+0.046%
Uniform	~1.25%	~1.443%	29.563%	+0.053%
Uniform	~2.5%	~2.887%	29.580%	+0.070%
Uniform	~5%	~5.774%	29.779%	+0.269%
Uniform	~7.5%	~8.66%	30.206%	+0.696%
Uniform	~10%	~11.55%	30.937%	+1.427%
Gaussian	~0.6%	~0.75%	29.563%	+0.053%
Gaussian	~1.4%	~1.80%	29.574%	+0.064%
Gaussian	~2.4%	~3.0%	29.626%	+0.116%
Gaussian	~3.6%	~4.5%	29.718%	+0.208%
Gaussian	~4.8%	~6.0%	29.85%	+0.34%

Table 2.3: Reported performance of approximate multipliers in the literature compared to exact multipliers

Design	MRE	Power Savings	Area Reduction	Delay Decrease
DRUM [30]	1.47% (G)	59%	50%	47%
Vasileios [28]	3.6% (G)	34.14%	34.17%	11.11%
Suganthi [27]	7.63%	71.7%	55.6%	30.9%
Georgios [31]	2.5% (U)	47%	38%	35%
Tongxin [32]	1.64%	59.9%	50.1%	36.3%

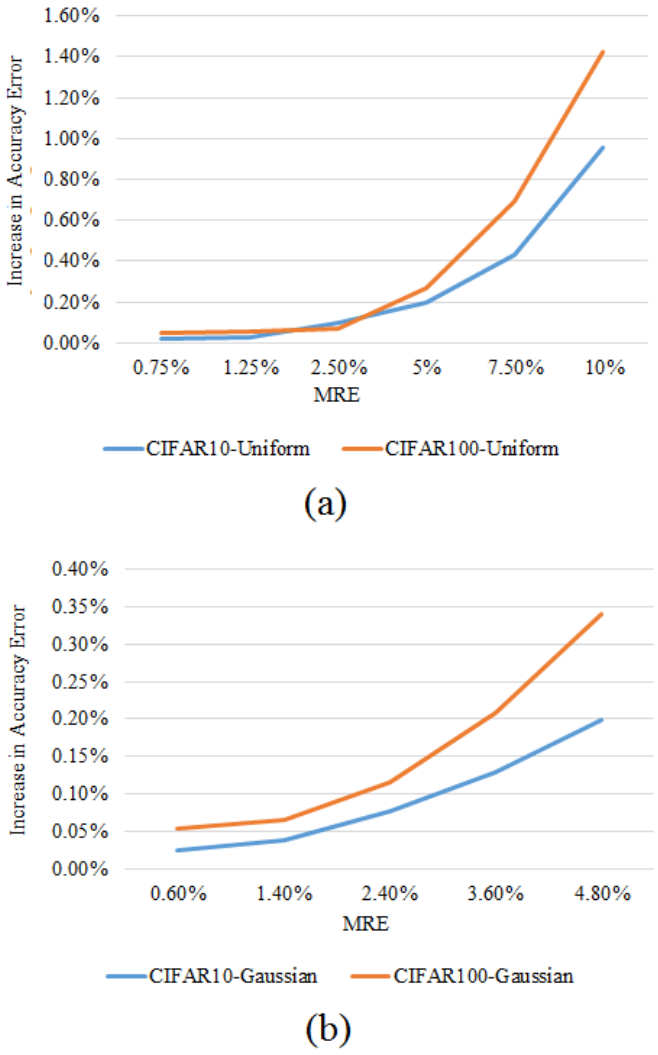

Figure 2.4: An estimated relationship between the MRE of approximate multipliers and the additional error in the accuracy of VGGNet.
(a) Uniform MRE Impact
(b) Gaussian MRE Impact

The approximate multipliers listed in Table 2.3 proposes different design methods on the hardware level for the approximate multiplication. For example, the design [28] uses a method of approximation that **is** performed by rounding the high radix values to their nearest power of two. The design [30] uses a different method which is applied by dynamic range selection scheme and truncation. These different design methods lead to different PDF characteristics for the MRE as specified in Table 2.3. The simulations presented in this

section are generic as presented in Table 2.1 and Table 2.2. These simulation results can be used for approximate mapping between an approximate multiplier's MRE error and the impact on the accuracy of the VGGNet.

Using 8-bits and 12-bits approximate multipliers, [33] has shown that a decrease in the required power and area can be achieved by increasing the error. For example, in the proposed 12-bit approximate multiplier, an increase in the maximum relative error from 1% to 2% has dropped the power from 475 µW to 284 µW and the area from 720.8 μm^2 to 523.8 μm^2. Also, the power has dropped from 247 µW to 125 µW and the area from 483 μm^2 to 285 μm^2 by increasing the error from 5% to 10%.

As mentioned earlier, Table 2.1 and Table 2.2 test cases list the network accuracy by averaging the simulation results of 100 loops. Figure 2.5 shows the network error for 100 loops of simulation for the Uniform PDF with MRE=~2.5% test case. The figure shows the loop number and the corresponding network accuracy. Interestingly, some of the loops achieved better results compared to an exact multiplier. The applied error matrices in these cases have randomly reshaped the layer's input for a better classification by the network.

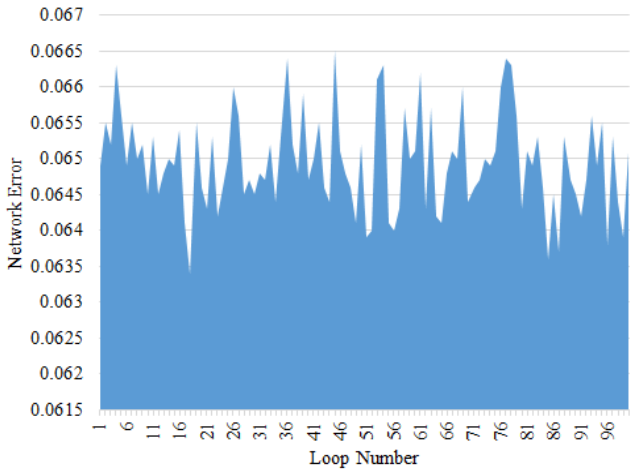

Figure 2.5 Accuracy results for 100 loops of simulation for the Uniform PDF with MRE=~2.5% test case.

2.6 Layers Impact Tests

In this section, the impact of applying approximate multiplication on specific layers is evaluated. Table 2.4 details the number of parameters per layer, the number of MACs in each layer and their percentage of the total number of MACs in the network. Table 2.4 information is based on one CIFAR-10 image.

To apply the layer impact testing, several test cases were simulated, where, consecutive layers were grouped together in each test case. A total of 6 sets of layer groups were used to assess the impact of approximate multiplication on these segments of the network. Table 2.5 illustrates the details of the applied test cases. For all the test cases CIFAR-10 was used with a uniform error matrix (MRE =~7.5%). The error matrix was applied on the specified layers only. As can be seen from Table 2.5, having an approximate multiplier on groups "L512-1" and "L512-2" have the least impact on the overall accuracy. These layers count for 39.14% of the total multiplications required as per Table 2.4. Therefore, a hybrid approach can be used in which the approximate multiplication is used in only these deeper layers.

The hybrid approach simulation results are presented in Table 2.6. In this approach, the approximate multiplication was applied only on the 512 channel layers of the network. A subset of tests from Table 2.1 were repeated in Table 2.6 to test this hybrid approach. Table 2.6 lists the reduction in network error achieved compared to the "all layers" approach in the previous section as per Table 2.1.

As can be seen from Table 2.5, the hybrid approach has a negligible impact on the accuracy compared to an exact multiplier, for the case of Gaussian PDF with MRE=~1.4% which is similar to the approximate multiplier proposed in [30], the added network error rate is only 0.017%. Additionally, this approach has a reduced additional network error compared to the "all layers" approximate multiplier approach which was presented in the previous section. In summary, using an approximate multiplier similar to [30], 39.14% of the

network's MACs can have approximately half the power, area and delay with the cost of 0.017% of an added network error.

Table 2.4: Number of MACs per layer

Layer	Param Count	MACs per Image	(%) of MACs.	Output Size per Image
Conv-64-1	1.79k	1.77M	0.564%	(32,32,64)
Conv-64-2	36.93k	37.75M	12.042%	(32,32,64)
Conv-128-1	73.86k	18.87M	6.021%	(16,16,128)
Conv-128-2	147.58k	37.75M	12.042%	(16,16,128)
Conv-256-1	295.17k	18.87M	6.021%	(8,8,256)
Conv-256-2	590.08k	37.75M	12.042%	(8,8,256)
Conv-256-3	590.08k	37.75M	12.042%	(8,8,256)
Conv-512-1	1.18M	18.87M	6.021%	(4,4,512)
Conv-512-2	2.36M	37.75M	12.042%	(4,4,512)
Conv-512-3	2.36M	37.75M	12.042%	(4,4,512)
Conv-512-4	2.36M	9.44M	3.011%	(2,2,512)
Conv-512-5	2.36M	9.44M	3.011%	(2,2,512)
Conv-512-6	2.36M	9.44M	3.011%	(2,2,512)
FC-1	262.66k	262.14k	0.084%	(512)
FC-2	5.13k	5.12k	0.002%	(10)
Total	15M	313.46M	100%	N/A

Table 2.5: Layers testing results using Uniform error matrix (MRE=~7.5%)

Group ID	Error Matrix Location	Network Err. Rate	Error Diff. from Exact Multiplier
L64	The 2 conv-64 layers	6.715%	+0.315%
L128	The 2 conv-128 layers	6.484%	+0.084%
L256	The 3 conv-256 layers	6.513%	+0.113%
L512-1	The first 3 conv-512 layers	6.445%	+0.045%
L512-2	The second 3 conv-512 layers	6.42%	+0.02%
LFC	The fully-connected layers	6.612%	+0.212%

Table 2.6: Tests results for the hybrid approach

Matrix *E* Type	MRE	Network Error Rate	Error Diff. from Exact Multiplier	Err. Diff from All Layer Appr. Multiplier
Uniform	~1.25%	6.414%	+0.014%	-0.016%
Uniform	~2.5%	6.419%	+0.019%	-0.076%
Uniform	~5%	6.437%	+0.037%	-0.161%
Uniform	~7.5%	6.488%	+0.088%	-0.340%
Uniform	~10%	6.502%	+0.102%	-0.856%
Gaussian	~1.4%	6.417%	+0.017%	-0.021%
Gaussian	~2.4%	6.420%	+0.020%	-0.057%
Gaussian	~3.6%	6.432%	+0.032%	-0.098%
Gaussian	~4.8%	6.436%	+0.036%	-0.163%

2.7 Conclusion

This research work demonstrates that deep learning can be optimized using approximate computing. Approximate computing can reduce the chip power and area while increasing the speed. The paper started by providing a background on the approximate multiplier, then the

concept of approximate multiplier simulation was introduced. The primary contribution was to simulate the impact of the approximate multipliers on the accuracy of image classification using VGGNet. The simulation included applying several error matrices with various MRE values which cover the impact of several proposed approximate multipliers in the literature. The simulation covered both Uniform and Gaussian PDFs and assessed their impact on the network's accuracy. The simulation results show that approximate multipliers have very little impact on the network's accuracy. This comes with the advantage of significant reductions in power, area, and delay. Additionally, an alternative hybrid approach was proposed which uses a mix of exact and approximate multipliers. In the hybrid approach, the approximate multiplication can be used in deeper layers of the network which has the least impact on the accuracy. The hybrid approach simulation leads to a reduced negligible impact on the accuracy while having significant savings in power, area, and delay on a large portion of the network.

CHAPTER 3 DEEP LEARNING TRAINING WITH SIMULATED APPROXIMATE MULTIPLIERS

Issam Hammad, Kamal El-Sankary and Jason Gu

 I. Hammad, K. El-Sankary and J. Gu, "Deep Learning Training with Simulated Approximate Multipliers," *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 47-51, doi: 10.1109/ROBIO49542.2019.8961780.

3.1 Abstract

This paper presents by simulation how approximate multipliers can be utilized to enhance the training performance of convolutional neural networks (CNNs). Approximate multipliers have significantly better performance in terms of speed, power, and area compared to exact multipliers. However, approximate multipliers have an inaccuracy which is defined in terms of the Mean Relative Error (MRE). To assess the applicability of approximate multipliers in enhancing CNN training performance, a simulation for the impact of approximate multipliers error on CNN training is presented. The paper demonstrates that using approximate multipliers for CNN training can significantly enhance the performance in terms of speed, power, and area at the cost of a small negative impact on the achieved accuracy. Additionally, the paper proposes a hybrid training method which mitigates this negative impact on the accuracy. Using the proposed hybrid method, the training can start using approximate multipliers then switches to exact multipliers for the last few epochs. Using this method, the performance benefits of approximate multipliers in terms of speed, power, and area can be attained for a large portion of the training stage. On the other hand, the negative impact on the accuracy is diminished by using the exact multipliers for the last epochs of training.

3.2 Introduction

With the accelerated increase in computational power, cloud computing resources, and the availability of data, deep learning [1] has become a viable approach to solve artificial intelligence problems in various fields. Deep learning is used nowadays in many fields and applications such as self-driving cars, image recognition and classifications, robotics, health care, and security. One of the major challenges that deep learning faces is the slow training time especially using very deep neural networks with enormous data to train. Training a deep convolutional neural network usually requires thousands of feed-forward and back-propagation iterations. In each iteration, all the network weights are updated. These weights can be in millions as in the case of the VGGNet-16 network in

[2] which has 138M weights. The primary mathematical operation in a deep convolutional network is multiplication, therefore, any reduction in the cost of the multiplication will lead to a major enhancement to the performance of the entire system.

Approximate computing provides a solution to enhance performance in terms of speed, power, and area at the cost of a pre-defined error range in the obtained output. One of the primary applications for approximate computing is the approximate multipliers. Several approximate multiplier designs were proposed in the literature such as [27-28],[30], and [32]. Using these multipliers can lead to significant performance enhancements. However, these enhancements have a cost of inaccuracy in the output which is usually defined by the Mean Relative Error (MRE). As an example, the multiplier in [30] achieves performance enhancements of 47% in speed, 50% in area, and 59% in power. However, these enhancements have a cost of an inaccuracy in the output defined by a Gaussian MRE of 1.47% with Standard Deviation (SD) of 1.803%. Like [30], the approximate multiplier designs [27-28] and [32]. have different performance enhancements with a predefined MRE error. The MRE is defined in equation (3.1), where, Xi is the exact multiplication value, Xi' is approximate multiplication value from an approximate multiplier, and n is the total number of samples.

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Xi' - Xi|}{|Xi|} \qquad (3.1)$$

In a previous work [18], we have studied the impact of using approximate multipliers on the inference stage of a pre-trained CNN network. The simulated MRE and SD in [18] were selected to approximately simulate the reported inaccuracies by various approximate multipliers in the literature such as [27-28],[30], and [32]. The work in [18] has demonstrated that with minimal cost of added inaccuracy, approximate multipliers can be used to enhance to significantly boost the inference performance of a pre-trained deep convolutional neural network (CNN) in terms of speed, power, and area.

Lower the training cost in terms of the power, area, and speed can be very beneficial in the case of training on the edge. Training is computationally very expensive, and it is usually performed using high powerful servers. However, in certain instances, training will have to be performed at least partially on the edge. When training on the edge is performed, the model will be trained partially or fully using the low power embedded hardware.

Fully autonomous mobile robots with image classification abilities are a perfect example of the need for training on the edge at least partially. These robots are used in various industries today such as aerospace applications, nuclear power plants, oil refineries, chemical factories, underwater and military applications. In many instances, these robots will be operating in offline areas without any connection to the main server. Therefore, for the purpose of improving prediction accuracy, continuous model training could be required and will have to be performed on the edge. Hence, as a result of this need for deep learning train the edge, proposing methods to improve training performance in terms of power, area, and speed becomes vital. This can be achieved by utilizing approximate multipliers during training as this paper will present.

This paper's objective is to propose new methods to enhance the training stage performance for a deep CNN by simulating the impact of the approximate multiplier error during training. One of the primary research contributions is a new training methodology which enhances the training performance of deep convolutional networks without any negative impact on the accuracy. This is accomplished by training the network using two phases. In the first phase, the training starts using approximate multipliers, then in the second phase the training switches to exact multipliers. Using this methodology all the performance gains of approximate multipliers can be obtained during the first phase of the training, while in the second phase, any negative impact on the accuracy caused by the approximate multipliers will be diminished. The number of iterations for each phase is a variable that is determined by the inaccuracy of the approximate multiplier.

This paper is structured as follows, in section 3.3 presents the details of the used deep CNN and dataset. Section 3.4 demonstrates the achieved inference accuracy by training the deep

CNN with simulated approximate multiplier error. Section 3.5 presents the new proposed hybrid training approach. In Section 3.6 the research conclusions are summarized.

3.3 Simulation Environment

For the simulation, a modified version of the VGGNet was used as a model. This modified version was proposed by [37] and it slightly differs from the original design which was proposed by [2]. The design in [37] is tailored to work with CIFAR-10 image dataset [36] which is used in this study. The model in [37] is smaller than the original model in [2] as it has a 32x32 input size rather than a 224x224 input, additionally, it has 2 fully dense layers rather than 3 fully dense layers. It also includes batch normalization and dropout to reduce overfitting. Figure 3.1 demonstrates the modified architecture of the VGGNet which is used for the simulations in this paper. Note that in Figure 3.1, the first two numbers in the brackets contain the image dimension while the third number reflects the number of filters. CIFAR-10 dataset [36] consists of 60000 color images divided equally into 10 classes with 50000 images used for training and 10000 images used for testing.

For development, the popular deep learning Python library Keras was used [35]. The used model was adopted from the repository in [34], which presents an implementation of the design proposed in [37]. In this paper, the repository in [34] was modified to evaluate the applicability of training with approximate multipliers. Table 3.1 specifies the used training configurations for all the test cases in this paper. During the simulation, the type “float16” was used. To simulate the impact of approximate multipliers on the training, the multiplication should have an inaccuracy defined by a certain MRE and SD. This was implemented using a Keras custom layer functionality. These layers were added before every convolutional and dense layer. These custom layers were programmed to mimic the impact of the error in approximate multipliers by creating a multiplication inaccuracy based on a specific MRE and SD during both backpropagation and forward propagation. These layers simulate this inaccuracy through elementwise multiplication between the weights and a generated error matrix. Each network layer had a unique error matrix which simulated

a certain MRE and SD. Having these custom layers throughout the network simulated the impact of the approximate multiplier error on the overall accuracy.

Table 3.1: Training configurations

Parameter	**Value/Method**
Epochs	200
Batch Size	128
Output Classes	10
Activation Function	ReLU
Loss Function	Categorical crossentropy
Optimizer	Stochastic gradient descent (SGD) optimizer with learning rate decay
Dataset	CIFAR-10
Training Images	50000
Testing Images	10000
Regularization	L2 Regularization with weight decay (0.0005) and Dropout of 30%-50%
Normalizaion	Input Normalization and Batch Normalization

Figure 3.2 illustrates a histogram of a sample error matrix which is used to simulate an MRE of ~3.6% and an SD of ~4.5%. Many of the reported approximate multipliers MREs have a near zero-mean Gaussian distribution, this can be seen in the approximate multipliers [30] and [28]. Therefore, to provide a generic simulation that can be applicable to many approximate multiplier models, all the simulated MRE values in this paper were based on a near zero-mean Gaussian distribution.

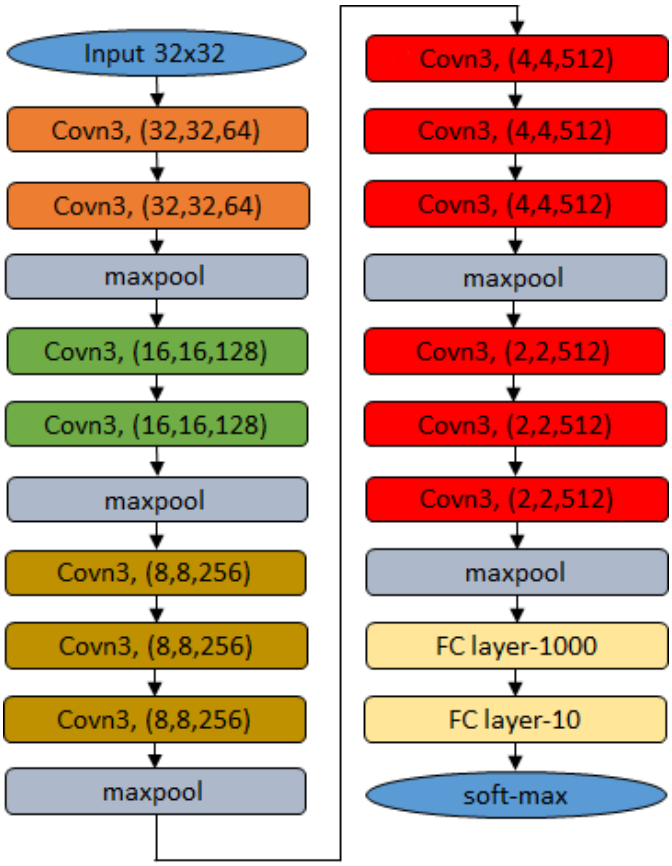

Figure 3.1: Modified VGGNet architecture which was used for this study

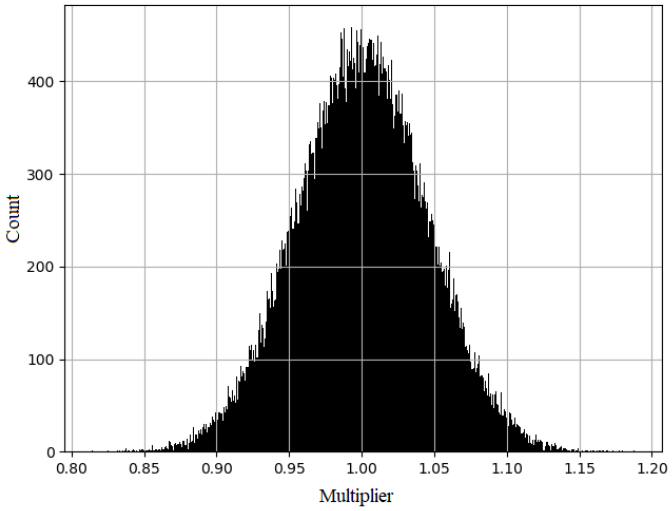

Figure 3.2: A histogram (500 bins) of a sample error matrix (MRE=~3.6%, SD=~4.5%)

3.4 Training with Simulated Approximate Multiplier Error

As described in the previous section, simulating the approximate multiplier error during the training stage was achieved using Keras custom layers. These layers create a multiplication inaccuracy based on the tested MRE and SD. The error simulation is achieved by multiplying the layers' weights with an error matrix which is generated to simulate the desired MRE.

Figure 3.3 demonstrates the followed process for this simulation. After loading the data, an error matrix with an approximate MRE and SD was generated for each layer. This simulated the impact of an approximate multiplier on the accuracy of the network. By using Keras custom layers as described in the previous section, the approximate multiplier error simulation was applied during all forward propagation and backpropagation iterations. During the training, the weights after certain training epochs were downloaded. This allowed the training to resume from that epoch when reloading the model, also this was needed to implement the hybrid approach which will be discussed in the next section. After completion, the final model weights were downloaded, and the model was reloaded for testing purposes. The testing stage excluded the simulation of the approximate multiplier error, therefore, all the added Keras custom layers were removed. This ensured that any impact on the inference accuracy is resulting from applying the approximate multipliers simulation during the training stage only.

Table 3.2 presents the achieved inference accuracy as a result of training with simulated approximate multiplier error. The table lists the results for different approximate multiplier configurations based on their MRE and SD. The first row presents the inference accuracy achieved as a result of training with an exact multiplier which excludes any error simulations. This achieved accuracy will be referred to as the baseline accuracy (93.6%). The remaining cases are reported based on the simulated MRE and SD values. In each test case, the achieved inference accuracy and the difference in accuracy compared to the baseline accuracy are reported. The simulation differences between the presented test cases were limited to the ranges of MRE and SD, this guaranteed a fair performance comparison

among the presented test cases. As can be seen from the table the impact of the approximate multiplier error during training on the achieved inference is very small, especially for lower MRE cases.

To clearly demonstrate the benefit of this simulation, a mapping can be done between the simulated test cases and the reported performances of approximate multipliers in the literature. For example, DRUM [30] reported performance enhancements of 47%, 50% and 59% in the speed, area, and power, respectively with a cost of a near zero-mean Gaussian distribution with MRE=1.47% and SD=1.803%. This is very close to test case 2 in Table 3.2 with MRE=~1.4% and SD=~1.8% which also has a zero-mean Gaussian distribution. In other words, using the approximate multiplier DRUM [28] in a custom design can approximately accelerate all the multiplications of the network during training by 47% with a cost of a drop in the inference accuracy by only 0.07%.

Any improvement in the multiplication performance will directly boost the convolution performance as convolution is just a series of Multiplication and Accumulation (MACs) operations. Based on [38], the convolution in a CNN consumes 90.7% of the total computational time required by the network. Thus, any performance improvement on the multiplication will directly affect the performance of the entire network.

Based on [33], there is a high correlation between the approximate multiplier error and the performance gains achieved. Hence, using approximate multipliers with higher error leads to higher performance gains for a custom hardware design for CNN training. Nevertheless, the network's ability to tolerate error is limited, after a certain level the network accuracy will collapse. This can be seen in the significant drop in accuracy in test cases 7 and 8 in Table 3.2. Therefore, a balance must be maintained between the approximate multiplier error and the CNN performance gains.

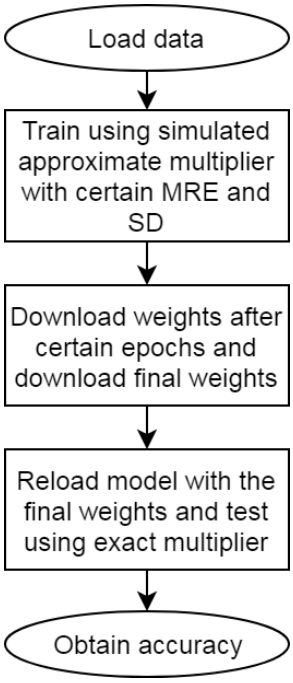

Figure 3.3: The followed procedure for simulating the impact of approximate multipliers on the training stage

Table 3.2: Inference accuracy based on training with simulated approximate multiplier error

Test ID	**MRE**	**SD*(σ)***	**Achieved Accuracy**	**Diff. From Exact**
0	0%	0%	93.6%	N/A
1	~1.2%	~1.5%	93.59%	-0.01%
2	~1.4%	~1.8%	93.53%	-0.07%
3	~2.4%	~3.0%	93.35%	-0.25%
4	~3.6%	~4.5%	93.23%	-0.37%
5	~4.8%	~6.0%	93.11%	-0.49%
6	~9.6%	~12%	93%	-0.60%
7	~19.2%	~24%	92.23%	-1.37%
8	~38.2%	~48%	65.65%	-27.95%

3.5 The Hybrid Training Approach

As presented in the previous section, despite the great performance gains that can be achieved by training with approximate multipliers, a cost of a slight drop in the network accuracy is inevitable. To eliminate this cost, a hybrid training methodology can be applied which involves using both approximate multipliers and exact multipliers. Using this methodology, the training starts with approximate multipliers then switches to exact multipliers for the last epochs of the training By evaluating this hybrid training methodology, test cases 1-6 in Table 3.2 and up to MRE=~9.6% reached an accuracy within 0.02% of baseline accuracy.

Table 3.3 illustrates, the number of epochs that were used by the approximate multipliers then the exact multipliers to achieve an inference accuracy which is equal or greater than 93.58% (0.02% less than the baseline accuracy).

In deep learning, neural network weights can be downloaded at any point and the training can be resumed from pre-loaded weights. Therefore, realizing the hybrid approach using a custom hardware design is not complicated. For example, one chip can be designed for training using approximate multipliers and the other using exact multipliers, the exact multiplier training chip can resume and finish what was partially training by the approximate multiplier training chip.

Table 3.3: Hybrid training configurations for different MRE values

Test ID	MRE	Approximate Multiplier Epochs	Exact Multiplier Epochs	Approximate Multiplier Utilization
1	~1.2%	200	0	100%
2	~1.4%	191	9	95.5%
3	~2.4%	180	20	90%
4	~3.6%	176	24	88%
5	~4.8%	173	27	86.5%
6	~9.6%	151	49	75.5%

The results in Table 3.3 were obtained by following the procedure presented in the flowchart in Figure 3.4. Table 3.3 presents the optimal hybrid solution found for each test case and. In this procedure, the training started by loading partially trained model weights from a simulated approximate multiplier up to certain epoch. These weights were saved after certain epochs during the simulations which were presented in the previous section. After loading these partially trained models, the remainder of the training was resumed by an exact multiplier up to 200 epochs as specified in Table 3.1. Finding this optimal solution required tuning the switching epoch between the approximate and the exact multiplier increasing it or decreasing it until finding the optimal combination.

Table 3.3 presents the optimal hybrid solution for this hybrid approach. Nevertheless, in production, any repeat in training must be avoided as it defeats the purpose of performance enhancement. Subsequently, it will be challenging to obtain this optimal switching epoch prior to training and without computational costs. However, using a non-optimal solution by approximating the switching epoch index for the hybrid approach still achieves significant performance gains. If the final achieved inference accuracy by the hybrid approach is almost equal to the exact multiplier accuracy, any utilization of the approximate multipliers for the initial epochs are pure performance gain with almost no cost.

In general, developers usually keep training until there are no further improvements to the cross-validation accuracy. Therefore, regardless of what the initial switching epoch index was, the target accuracy can be achieved if the approximate multiplier error is suitable for the application. In the case of starting with an initial switching epoch index less than the optimal, the target accuracy should be achieved by the final epoch. On the other hand, if the initial switching epoch was larger than the optimal, the target accuracy can be achieved by training for a few additional epochs. In both cases, the norm is to keep training until the cross-validation accuracy flattens. Therefore, the advantage of training the initial epochs with approximate multipliers can be attained regardless.

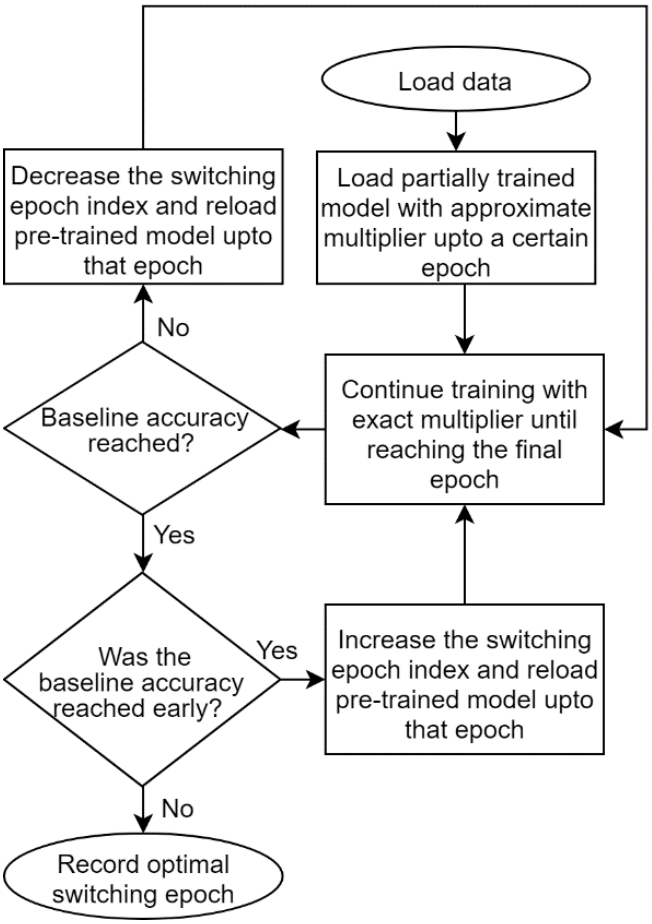

Figure 3.4: The followed procedure for finding the optimal solution for the hybrid training approach

3.6 Conclusion

In this paper, the concept of utilizing approximate multipliers to enhance the training performance of deep CNN was proposed. Simulation results show that using approximate multipliers for CNN training result in a minimal drop in accuracy while having the potential to achieve significant performance gains in custom hardware designs. Additionally, a hybrid approach was proposed in which the training starts approximate multipliers then switches to exact multipliers after a certain epoch. The simulation results of the hybrid

approach show that using exact multipliers for the last epochs can eliminate any accuracy drop caused by the usage of approximate multipliers initially. Therefore, significant performance gains can be achieved by utilizing approximate multipliers for a large portion of the training while having almost no negative impact on the final achieved accuracy.

While this concept can be used to enhance the performance of deep learning training in general, it is particularly beneficial in the case of training on the edge. Training on the edge is required in the case of offline systems such as in the case of offline mobile robots performing in remote harsh environments. Lowering the edge training cost for these robots in terms of power, area, and speed is vital for performance. This performance enhancement can be achieved using approximate multipliers for the training with minimal impact on the accuracy as the paper presented.

CHAPTER 4 CNN INFERENCE USING A PREPROCESSING PRECISION CONTROLLER AND APPROXIMATE MULTIPLIERS WITH VARIOUS PRECISIONS

Issam Hammad, Ling Li, Kamal El-Sankary, and W. Martin Snelgrove

 I. Hammad, L. Li, K. El-Sankary and W. M. Snelgrove, "CNN Inference Using a Preprocessing Precision Controller and Approximate Multipliers with Various Precisions," in IEEE Access, vol. 9, pp. 7220-7232, 2021, doi: 10.1109/ACCESS.2021.3049299.

4.1 Abstract

This paper proposes boosting the multiplication performance for convolutional neural network (CNN) inference using a precision prediction preprocessor which controls various precision approximate multipliers. Previously, utilizing approximate multipliers for CNN inference was proposed to enhance the power, speed, and area at a cost of a tolerable drop in the accuracy. Low precision approximate multipliers can achieve massive performance gains; however, utilizing them is not feasible due to the large accuracy loss they cause. To maximize the multiplication performance gains while minimizing the accuracy loss, this paper proposes using a tiny two-class precision controller to utilize low and high precision approximate multipliers hybridly. The performance benefits for the proposed concept are presented for multi-core multi-precision architectures and single-core reconfigurable architectures. Additionally, a design for a merged reconfigurable approximate multiplier with two precisions is proposed for utilization in single-core architectures. For performance comparison, several segments-based approximate multipliers with different precisions were synthesized using CMOS 15nm technology. For accuracy evaluation, the concept was simulated on VGG19, Xception, and DenseNet201 using the ImageNetV2 dataset. The paper will demonstrate that the proposed concept can achieve significant performance gains with a minimal accuracy loss when compared to designs that utilize exact multipliers or single-precision approximate multipliers.

4.2 Introduction

Approximate computing is emerging as a viable way to achieve significant performance enhancement in terms of power, speed, and area for computationally heavy digital system on chip (SoC) designs [12-16]. Even though a significant performance enhancement can be achieved using approximate computing, these techniques have the obvious cost of certain levels of inaccuracy in the output. However, for large systems, what matters is the impact of the approximate computing on the accuracy of the entire system and not on each sub-module that resides within the system. Approximate multipliers are one of the most common operators for approximate computing. These multipliers produce an approximated

output for the multiplication which contains a certain inaccuracy, However, they can achieve significant performance gains in terms of power, speed, and area compared to exact multipliers when utilized in SoC design. Improving the performance by increasing the speed and lowering the power allows for reducing the energy consumption per operation. Several approximate multiplier designs have been proposed in the literature such as [29-31] and [39-41].

Image recognition using deep learning [1] has been booming in the last few years. Using fixed-point arithmetic to improve the performance of convolutional neural network (CNN) accelerators was proposed by the industry as can be seen in the articles published by Qualcomm in [42] and IBM in [43]. Additionally, several application-specific integrated circuit (ASIC) designs for fixed-point CNN accelerators have been proposed in the literature such as [8-9],[11] and [44]. Using a 16-bit base for the design of ASIC CNN accelerators is common as can be seen in [8-9] and [44]. Additionally, several field-programmable gate array (FPGA) designs for 16-bit based fixed-point CNN accelerators have been proposed such as [45-47].

Based on the architectures in [8-9],[11], and [44-47] CNN accelerators are designed using arrays of processing elements (PEs), at the core of each PE, a multiply and accumulate (MAC) unit exists. Hence, the multiplier is a primary component in the design of CNN accelerators, and any improvement in the performance of the multiplier will scale up to improve the performance of the entire accelerator. The focus of CNN accelerators has been on improving the inference performance. This is because CNN training is usually done once using powerful graphics processing units (GPUs), following that inference is performed thousands or even millions of times before a model update or retraining is required.

The utilization of approximate multipliers in the hardware design of convolutional neural networks (CNNs) has been proposed previously to enhance the performance in terms of power, speed, and area [18][20][33][48-50]. Moreover, using a reconfigurable approximate multiplier based on calculating the error variance was proposed in [51]. Lower precision

approximate multipliers can achieve higher performance gains as can be seen in [18],[20],[33], and [48-49]. However, this performance enhancement has a cost of a drop in the CNN accuracy which is inversely proportional to the precision. This creates a trade-off in terms of how much performance gains can be achieved vs. how much accuracy can be sacrificed. Hence, utilizing low precision approximate multipliers might not be feasible when the accuracy loss is large. Based on the challenge that this trade-off presents, the paper proposes the concept of predicting and dynamically configuring the precision of approximate multipliers for CNN inference. The paper will demonstrate that the impact of approximate multipliers' precision on the inference accuracy varies widely between the different image classes. An image class contains a group of images that belong to the same category. (e.g: hen, bee, zebra … etc). For certain image classes, the CNN achieves a lower accuracy when lowering the approximate multiplier precision, for other image classes the CNN accuracy stays constant, and interestingly, in other smaller percentages of image classes, the CNN achieves higher accuracy with lower precision approximate multipliers. Reaching this finding was accomplished experimentally, were utilizing lower precision approximate multiplier results in a more optimal CNN solution for certain image classes. Accordingly, the image classes in a dataset can be divided into two precision categories, a low precision category which contains image classes that can be predicted with the same CNN accuracy or better using lower precision approximate multipliers, and a high precision category which contains the rest of the image classes.

To predict the adequate processing precision for each image as low or high, the paper proposes using a tiny two-class CNN preprocessing precision controller. The controller can be utilized in a system that contains multiple approximate-multiplier based CNN inference accelerators with different precisions, or in a single CNN inference accelerator built with precision reconfigurable approximate multipliers. The controller's objective is to maximize the overall performance gains by maximizing the usage of lower precision approximate multipliers whenever this does not cause an additional accuracy loss. This paper proposes a methodology which augments the performance of CNN inference using the concept of the precision controller to enable the utilization of existing low precision and high precision approximate multiplier hybridly. It also proposes a new precision

reconfigurable approximate multiplier to utilize the precision controller concept in single-core designs. Figure 4.1 illustrates a high-level design for the proposed concept which includes a preprocessing precision controller which controllers a CNN network complied on inference accelerators with reconfigurable approximate multipliers.

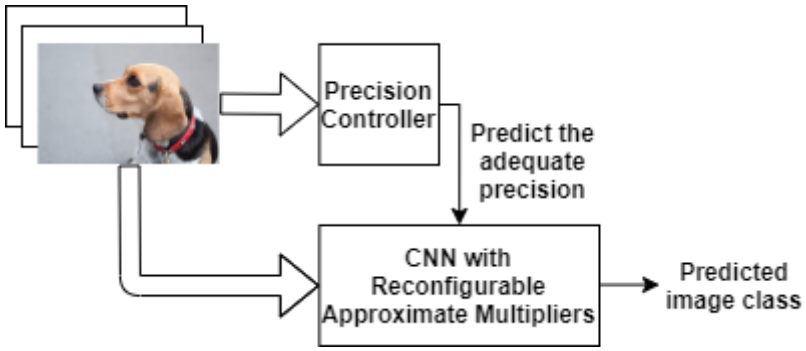

Figure 4.1: High-level demonstration of the proposed concept.

To demonstrate the performance gains of the proposed concept in terms of power, speed, and area, several approximate multiplier designs using the static segment method (SSM) and the dynamic segment method (DSM) with different precisions were synthesized using CMOS 15nm technology. For CNN accuracy analysis and comparison, Keras [35] was used to simulate the proposed design on VGG19 [3], Xception [6], and DenseNet201 [5] using ImageNetV2 TopImages dataset [52]. Two architectures for utilizing the proposed concept are presented in the paper. The first is a multi-core architecture that uses approximate-multiplier based CNN inference accelerators with various precisions, and the second is using a single-core accelerator built with precision reconfigurable approximate multipliers. To enable the utilization of the proposed concept in single-core designs, a new merged reconfigurable approximate multiplier with two precisions is proposed. This is an additional research contribution that the paper presents. Using both architectures, the paper will demonstrate that the proposed concept can achieve significant performance enhancements with minimal accuracy loss compared to architectures with 16-bit exact signed multipliers or a single-precision approximate multiplier. All simulations were based on a 16-bit representation as it is common in CNN accelerators as can be seen in [8-9],[11], and [45-47].

This paper is structured as follows: In section 4.3 the architecture of the segment based approximate multipliers using the static segment method (SSM) and the dynamic segment method (DSM) is presented. Section 4.4 presents the concept of training the precision prediction preprocessing controller. Section 4.5 presents the baseline performance and accuracy simulation SSM and DSM approximate multipliers with various segment sizes using VGG19, Xception, and DenseNet201. Section 4.6 illustrates how the proposed concept of using a precision controller with reconfigurable approximate multipliers can be utilized in both multi-core and single-core architectures. Section 4.7 concludes the research findings of this article.

4.3 Segment Based Approximate Multiplier

Several approximate multipliers techniques are proposed in the literature such Segmentation, High Radix, Rounding, and Perforation [53]. The proposed concept of using a pre-processing precision controller can be applied to any approximate multiplier technique with controllable precision. Nevertheless, segment-based approximate multipliers using both SSM and DSM were selected for the simulation to present the precision controller concept in this paper. This provides a comparison between a high precession multiplier (the DSM) and lower power and area one (the SSM). The DSM based multiplier using DRUM's design [30] can provide notably high accuracy, although it has a larger area and energy consumption compared to other multipliers such as [29],[39-40].On the other hand, the SSM based multiplier [39] has a more efficient circuit compared to other approximate multipliers such as [29-30] and [40]. Moreover, the segment-based technique allows for an efficient implementation for the merged reconfigurable approximate multiplier which is presented in this paper. Segment based approximate multiplication is one of the efficient and simple techniques used to design approximate multipliers. Using this technique, only a segment of the multiplicand is passed to the multiplier to approximate the multiplication. This allows for the multiplication of n × n number using an m × m multiplier, where m < n. As an example, a 16 × 16 bit multiplication can be approximated using an 8 × 8 bit multiplier with a byte segment or even 4 × 4 bit

multiplier with a nibble segment. Controlling the precision of these multipliers is performed by adjusting segment size (m). Figure 4.2 demonstrates the general design of a segment based approximate multiplier for an n × n number.

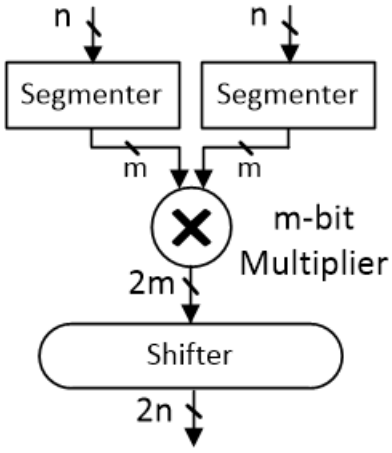

Figure 4.2: Segment based approximate multiplier.

Several techniques for the segment selection have been proposed in the literature, this includes the segment static segment method (SSM) which was adopted in the approximate multiplier design Narayanamoorthy et al. in [39] and the dynamic segment method (DSM) which was also proposed by [39] and was implemented in the approximate multiplier design DRUM in [30]. Based on [30] and [39], an inversely proportional relationship exists between the segment size and the achieved performance gains.

Using an SSM segmenter, an n-bit integer number is divided into a static i (m-bit) segments with a (k) offset between the starting bits of two consecutive segments. The m-bit segment with the leading one is selected to approximate the multiplication. Figure 4.3 demonstrates the circuit implementation of the SSM segmenter. As can be seen from the Figure 4.3, the SSM segmenter consists of a multiplexer (MUX) to select between the (i) segment. On the selection lines, the m-bits of each segment are passed through an OR gate.

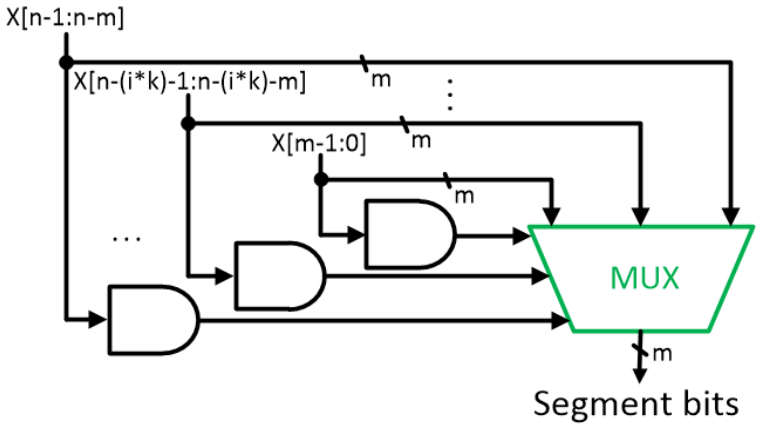

Figure 4.3: The SSM segmenter

Figure 4.4 illustrates an example of the possible segments for SSM using k=4 for with segment sizes of m=8 and m=4. The example shows the segmentation based on a signed 16-bit integer number. This slightly differs compared to the presented unsigned format in [39].

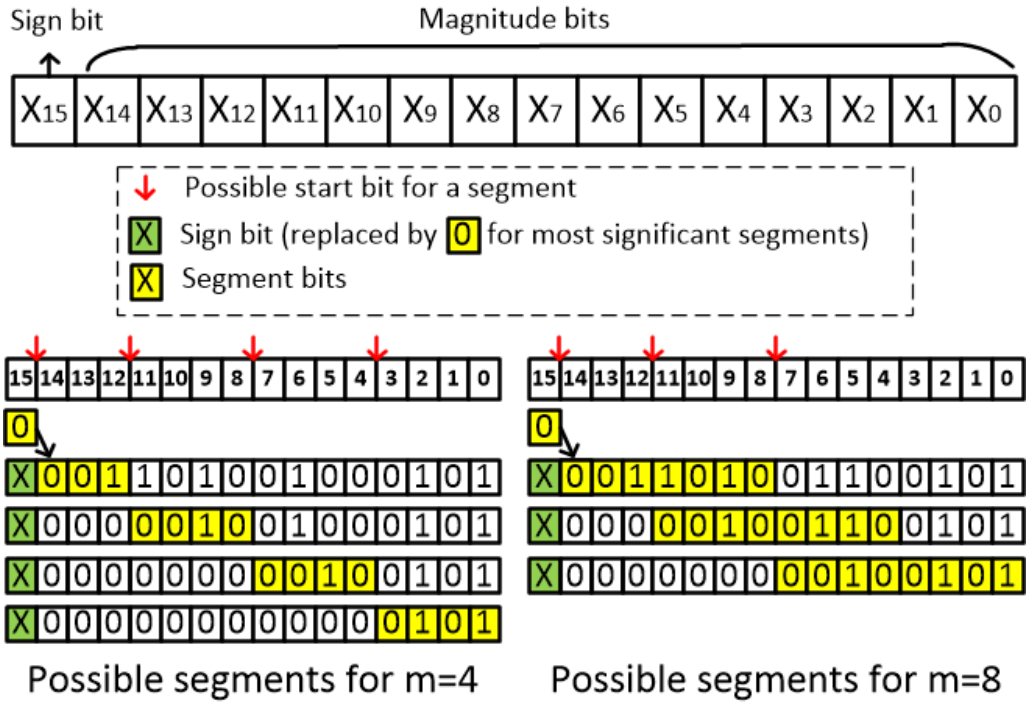

Figure 4.4: SSM segmentation example using k=4.

Figure 4.5 shows an example for an SSM based approximate multiplication using m=8. As can be seen in Fig 5., the 16 × 16 bit number can be estimated using two SSM segmenters, an 8 × 8 bit multiplier, a shifter, and an XOR for the sign bit. For the most significant segment, the sign bit is replaced by zero to create equal size segments.

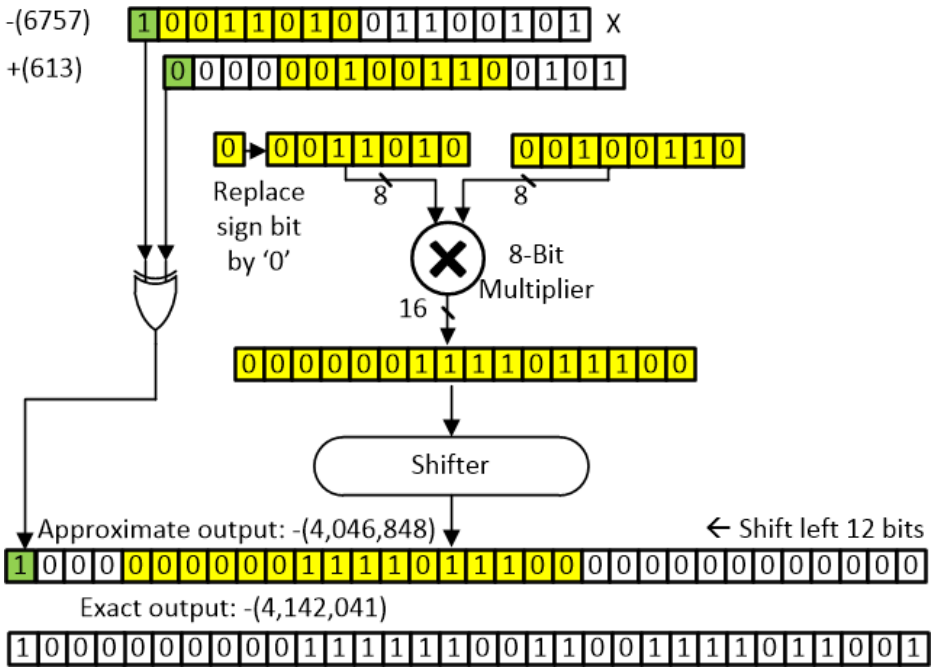

Figure 4.5: SSM approximate multiplication example using m=8

The DSM segmenter is more complicated and costly compared to the SSM segmenter. The DSM segmenter detects the leading one in an n-bit number then extracts the following m-1 bits to enable the utilization of an m × m bit multipliers to approximate n × n bit multiplication. The DSM method as originally proposed by [39] but was further improved in the approximate multiplier DRUM [30]. DRUM's approximate multiplier works by detecting the leading one in the number then extracting the following m-2 bits, any remaining truncated portion is estimated by setting the segment's least significant bit (LSB) to '1'. Figure 4.6 shows the circuit implementation for the DSM segmenter. The segmenter consists of a leading one detector (LOD) circuit in addition to an encoder and a MUX.

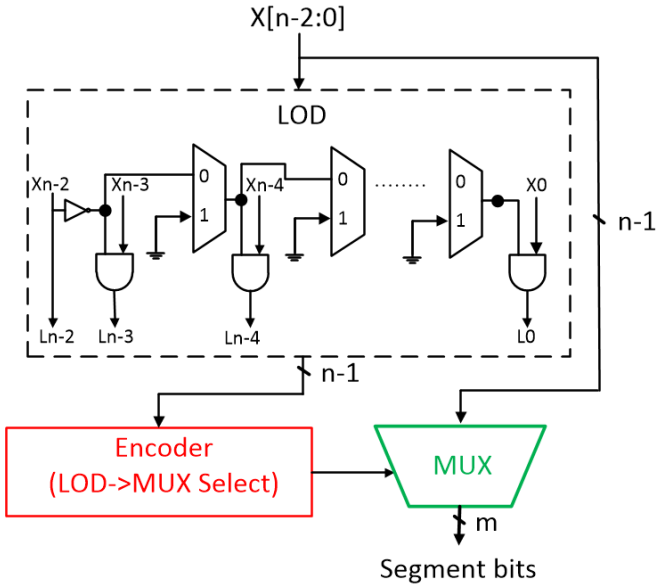

Figure 4.6: The DSM segmenter

Figure 4.7 shows an example of the DSM segmentation based on DRUM for both m=8 and m=4. As can be seen in Figure 4.7, the segment starts with the leading one, followed by the next m-2 bits. The segment's LSB is set to '1' to approximate the remaining truncated bits assuming a uniform distribution for the operands [30].

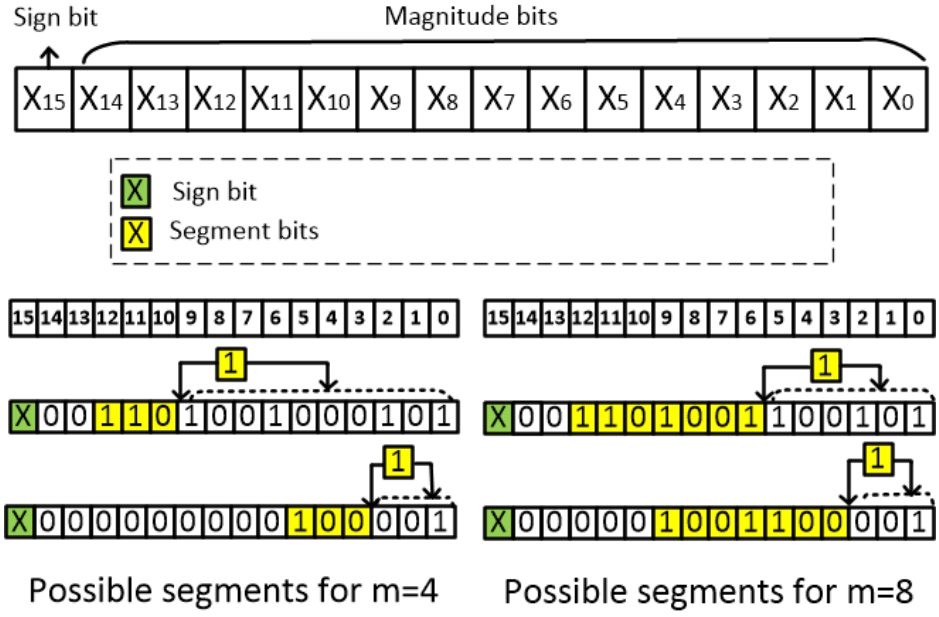

Figure 4.7: DSM segmentation example based on DRUM.

Figure 4.8 shows an example of the DSM multiplication using *m=8*. As can be seen in the figure, both *16 × 16* input numbers are segmented by extracting the leading one followed by the next *m-2* bits while the segments LSB bit is set to '1'. Following the segmentation, an *8x8* bit multiplier is used then the multiplier's output is shifted. For the sign bit, an XOR is used.

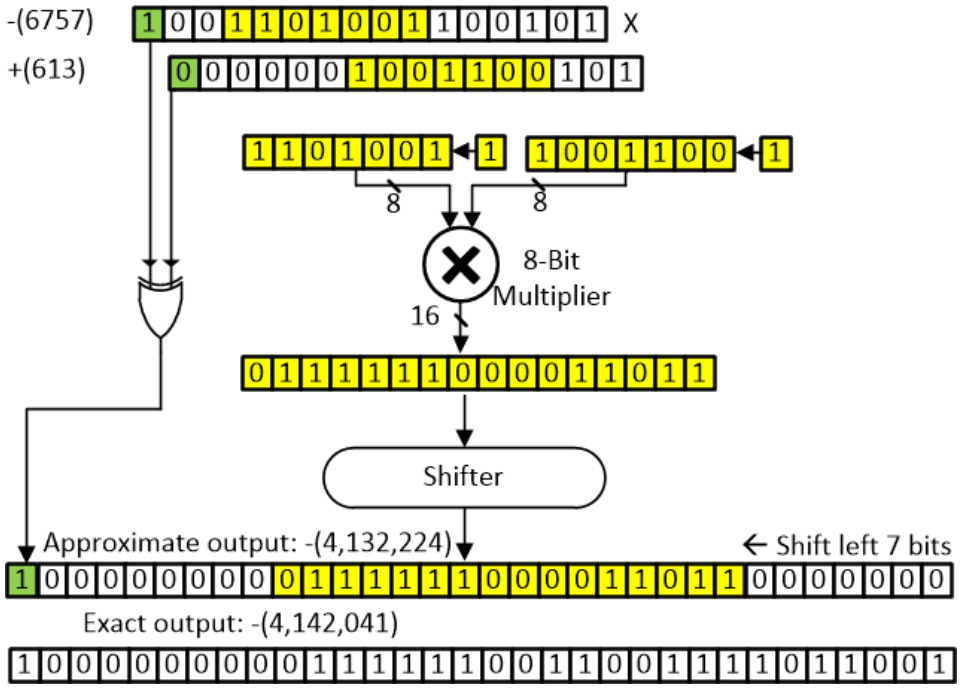

Figure 4.8: DSM approximate multiplication example using m=8

4.4 Building a Precision Preprocessing Controller

As previously discussed, utilizing a low precision approximate multiplier in CNN inference will not be feasible if it leads to large accuracy loss despite the massive performance gains it can achieve. As an example, an SSM approximate multiplier with m=4 can achieve a 153% speed increase, an 88% power reduction, and an 82% area reduction compared to an exact multiplier. However, using it in VGG19 inference based on ImageNetV2 causes a 15.3% accuracy loss compared to an exact multiplier as will be seen in the next section. This dilemma has created a motivation for finding a solution that allows for partial utilization of these multipliers only when the accuracy loss is minimal. A solution was found after studying the impact of the approximate multiplier precision on each image class

individually, it was found that the CNN's inference accuracy with low precision approximate multipliers varies widely between image classes. Surprisingly, certain image classes can be classified with higher accuracy using low precision approximate multipliers such as m=4 compared to higher precision approximate multipliers such as m=8. For another set of image classes, there was no difference in CNN classification accuracy between low and high precision approximate multipliers. The remaining larger set of image classes achieves better classification using a higher precision approximate multiplier.

To utilize this important finding in improving the overall performance by maximizing the usage of low precision approximate multipliers while minimizing the cost in terms of accuracy loss, a preprocessing precision controller was developed to predict the adequate precision category for the input image. This allows for the development of a system that contains approximate multipliers with different precisions or precision reconfigurable approximate multipliers. Such a system can maximize performance by utilizing low precision approximate multipliers only when achieving a better or the same CNN classification accuracy is predicted. This preprocessing controller is a two-class tiny CNN network that can predict the adequate approximate multiplier precision as either low or high. The exact segment sizes for what is considered low and what is considered high will depend on how the training was performed.

To train a preprocessing precision controller a labeled dataset with the adequate approximate multiplier precision for each image class must be created for each CNN network. To do that, the CNN predictions for each image using different approximate multiplier precisions should be obtained. Following that, if it was determined that on average an image class can be classified with the same or better accuracy using the lower precision mode, the entire images in this class will be labeled '0'. If using higher precision mode achieves better classification accuracy for that image class, the entire images in that class will be labeled as '1'. Figure 4.9 illustrates using a flowchart the process for developing a precision preprocessing controller for the CNN's approximate multipliers. Using the illustrated process in Figure 4.9, a precision controller with different precisions (m) was trained for VGG19, Xception, and DenseNet201 using a subset of ImageNet [54].

The training set consisted of 50 images from each image class with a total of 50000 images, while the cross-validation test set consisted of 10 images from each image class with a total of 10000 images. For the final accuracy evaluation, the controller along with the image classification CNN networks were tested using ImageNetV2 Top-Images dataset which consists of 10000 images using both SSM and DSM

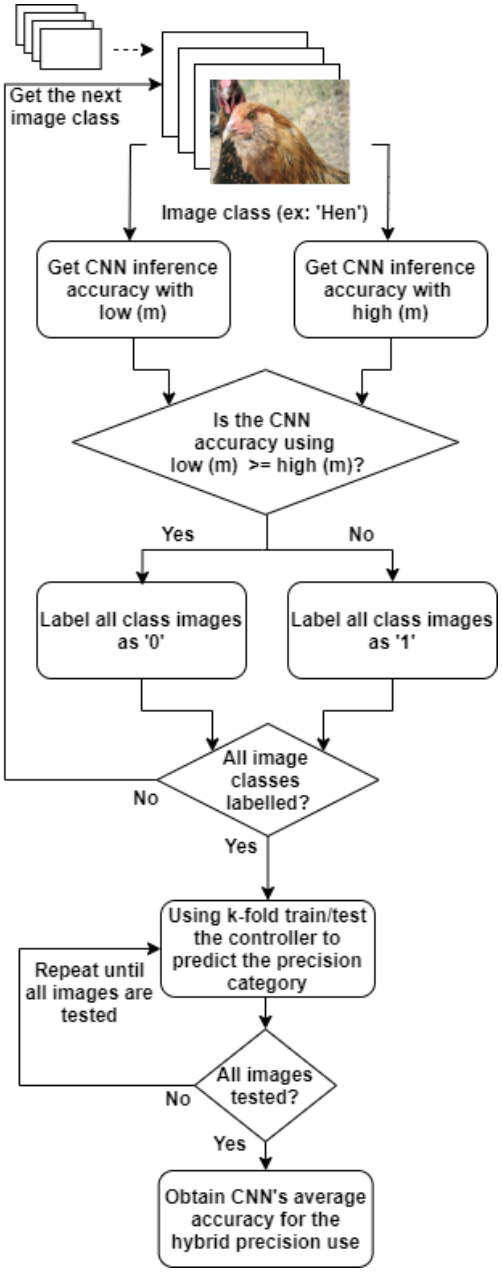

Figure 4.9: Developing a precision controller flowchart.

Figure 4.10 demonstrates the architecture of the preprocessing precision controller. The controller is a tiny CNN that has two output nodes with Softmax activation and contains three 2D convolution layers. These layers have shapes of (224,224,3), (56,56,6), and (14,14,12) consecutively. Following each convolutional layer, a (4x4) max-pooling layer exists. ReLU [55] activation was used in all the layers. The controller was designed to have a minimal overhead in terms of the number of parameters and the multiply and accumulate (MAC) operations. Therefore, the smallest possible design with the least parameters but with an acceptable accuracy was selected. Table 4.1 details the controller overhead compared to VGG19, Xception, and DenseNet201.

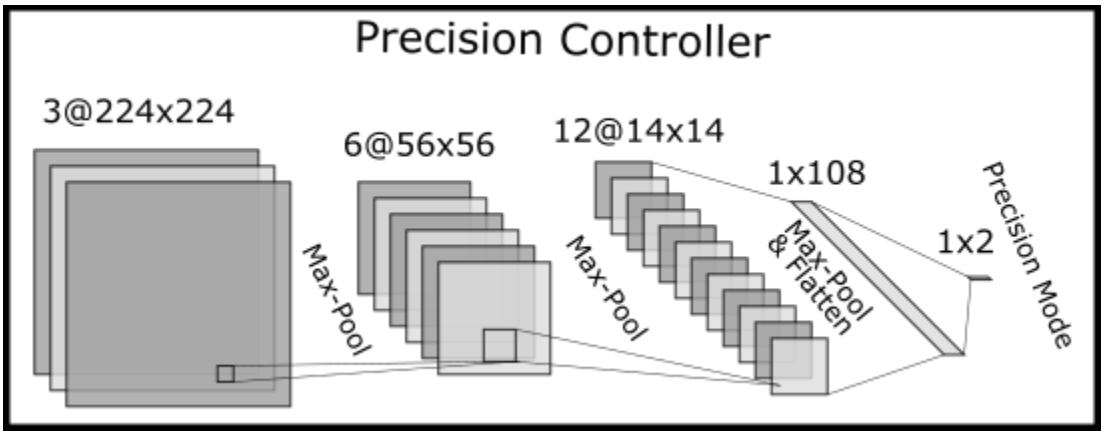

Figure 4.10: A proposed tiny CNN precision controller.

Table 4.1: Precision controller overhead comparison

Network	**Total Param.**	**Total MACs**	**Controller Param. Overhead**	**Controller MACs Overhead**
Controller	1130	4.7M	N/A	N/A
VGG19[3]	143.7M	19.64G	0.0008%	0.0239%
Xception[6]	22.9M	8.404G	0.0049%	0.0559%
DenseNet201[5]	20.2M	3.347G	0.0056%	0.1404%

As can be seen in Table 4.1, the controller parameters overhead is as low as 0.008% in the case of VGGNet19 and as high as 0.0056% in the case of DenseNet201. In terms of the MACs, overhead is as low as 0.0239% in the case of VGGNet19 and as high as

0.1404% in the case of DenseNet201. Based on these numbers, the preprocessing precision controller overhead can be considered negligible.

The next section will present a performance comparison between the SSM and the DSM approximate multipliers using m=4, m=6, and m=8 against the exact multiplier. Also, it will present the inference accuracy simulation for using these approximate multipliers using VGG19, Xception, and DenseNet201. Section V will build on section III and section IV to demonstrate how the precision controller can be utilized in multi-core and single-core architectures. Section V will also present the achieved controller accuracy for each network and accordingly, the achieved image classification accuracy with hybrid precision approximate multipliers.

4.5 Baseline Performance and Accuracy

To demonstrate and compare the performance gains that the SSM and DSM approximate multipliers can achieve, they were implemented using Verilog hardware description language (HDL) using Nangate 15nm FinFET standard cell library [56]. The Verilog implementation was synthesized using Synopsys Design Compiler. The analysis is applied to check the setup violations of the multipliers. The delay-annotated netlists of the multipliers are simulated using Modelism SE to verify their operations. In this implementation, the delay, power, and area using m=4, m=6, and m=8 precisions were obtained. Table 4.2 details the SSM performance and performance gains compared to an exact multiplier for each precision. Table 4.3 details the DSM performance and performance gains compared to an exact multiplier for each precision.

Table 4.2: The SSM [39] approximate multiplier performance compared to an exact multiplier

Design	Delay (ps)	Power (µW)	Area (μm^2)	Speed Increase	Power Reduction	Area Reduction
Exact	330.8	230.1	337.03	N/A	N/A	N/A
m=8	206.06	69.22	108.28	60.54%	69.92%	67.87%
m=6	182.75	52.09	83.705	81.01%	77.36%	75.16%
m=4	130.91	27.66	60.65	152.69%	87.98%	82.00%

Table 4.3: The DSM (Based on DRUM [30]) approximate multiplier performance compared to an exact multiplier

Design	Delay (ps)	Power (uW)	Area (μm^2)	Speed Increase	Power Reduction	Area Reduction
Exact	330.8	230.1	337.03	N/A	N/A	N/A
m=8	305.26	181.9	204.16	8.37%	20.95%	39.42%
m=6	261.61	133.3	170.289	26.45%	42.07%	49.47%
m=4	199.98	65.63	125.154	65.42%	71.48%	62.87%

As can be seen from Table 4.2 and Table 4.3, both the SSM and DSM can achieve significant performance gains compared to the exact multiplier. These gains are inversely proportional to the segment size (m). The SSM multipliers can achieve higher performance gains compared to the DSM multipliers due to having a simpler segmenter as was presented in Section II.

To evaluate the impact of the SSM and DSM approximate multiplication techniques on the CNN's inference accuracy, a simulation for these techniques were performed using three popular CNN networks, the VGG19, Xception, and DenseNet201. Pretrained models using ImageNet for these CNN networks are available as part of the deep learning platform Keras [35]. For the inference simulation, ImagenetV2 Top-Images [52] dataset was used. ImageNetV2 is a new test set built based on the ImageNet benchmark and was released primarily for inference accuracy evaluation. ImageNetV2 contains 10000 images based on 1000 different classes which are identical to the classes in the ImageNet dataset.

To simulate the impact of the approximate multiplication on the inference accuracy, Keras's custom layer functionality was utilized. This functionality allows for the addition of custom mathematical or logical operations anywhere inside the CNN. By utilizing this functionality, the SSM and DSM segmenters were simulated by creating segmentation layers. These segmentation layers were injected before all convolution layers and fully connected layers. In these layers, the previous layer's outputs were scaled and cast to Int16 numbers, then using bitwise operations, the segmenter was simulated by masking the numbers using a mask which was created based on the segment size (m). For the model

weight, all weights were scaled, cast, and masked to simulate the segmentation before starting the inference.

Table 4.4 lists the achieved CNN inference accuracy for VGG19, Xception, and DenseNet201 using the SSM technique with m=4, m=6, and m=8 precisions. Table 4.5 lists similar information but by utilizing the DSM technique. In both tables, the approximate multipliers' accuracies are compared against the accuracy of an exact signed Int16 multiplier. In both tables, the accuracy loss compared to an exact multiplier is listed between brackets.

Table 4.4: CNN inference accuracy using the SSM [39] approximate multiplication

Design	**VGG-19[3]**		**Xception[6]**		**DenseNet-201[5]**	
	Top-1 Accuracy	**Top-5 Accuracy**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Top-1 Accuracy**	**Top-5 Accuracy**
Exact	73.59%	92.28%	80.78%	95.92%	79.9%	95.59%
m=8	73.12% (-0.47%)	92.06% (-0.62%)	79.88% (-0.90%)	95.11% (-0.81%)	78.41% (-1.49%)	94.38% (-1.21%)
m=6	64.94% (-8.65%)	88.23% (-4.45%)	71.51% (-9.27%)	89.56% (-6.36%)	75.20% (-4.70%)	92.39% (-3.20%)
m=4	58.28% (-15.31%)	85.78% (-6.90%)	62.36% (-18.42%)	85.22% (-10.70%)	29.45% (-50.45%)	62.58% (-33.01%)

Table 4.5: CNN inference accuracy using the DSM (Based on DRUM [30]) approximate multiplication

Design	**VGG-19 [3]**		**Xception [6]**		**DenseNet-201 [5]**	
	Top-1 Accuracy	**Top-5 Accuracy**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Top-1 Accuracy**	**Top-5 Accuracy**
Exact	73.59%	92.28%	80.78%	95.92%	79.9%	95.59%
m=8	73.38% (-0.21%)	92.15% (-0.13%)	80.55% (-0.23%)	95.61% (-0.31%)	79.31% (-0.59%)	95.38% (-0.21%)
m=6	66.86% (-6.73%)	89.39% (-2.89%)	72.13% (-8.65%)	91.06% (-4.86%)	76.40% (-3.50%)	93.54% (-2.05%)
m=4	62.20% (-11.39%)	88.31% (-3.97%)	63.36% (-17.42%)	86.65% (-9.27%)	33.29% (-46.61%)	65.20% (-30.39%)

As can be seen in Table 4.4 and Table 4.5, the inference accuracy loss for all simulated networks increases when reducing the precision (m). Nevertheless, the inference accuracy loss using approximate multipliers with m=8 was very minimal. In the case of DSM, the accuracy loss was 0.21% for VGG19, 0.23% for Xception, and 0.59% for DenseNet201. For SSM, the difference was a bit higher with 0.47% for VGG19, 0.62% for Xception, and 1.49% for DenseNet201. Using SSM with a precision of m=4, the accuracy losses were significant where the losses were 15.31%, 18.42%, and 50.45% for VGG19, Xception, and DenseNet201, respectively. The losses were also significant using DSM as can be seen in Table 4.5.

For both the SSM and the DSM, DenseNet201 had a significant drop in accuracy when the precision was dropped from m=6 to m=4. In the case of SSM, the accuracy loss increased from 4.7% to 50.45%, while for DSM, the loss increased from 3.5% to 46.61%. This eliminated the possibility of utilizing the case of m=4 for DenseNet201 in any hybrid precision design as the loss is very large.

4.6 Using The Precision Preprocessor To Control Various Precisions Approximate Multipliers

4.6.1 Multi-Core Architecture

As CNN networks vary in terms of input shape, depth, and the number of filters, ASIC CNN accelerators are designed using a large-scale integration of PEs. This allows the accelerators to be reconfigurable which allows for the compilation of the CNN network that the user wants to deploy. The core component in the PE is the multiplier. This design can be seen in the fixed-point CNN accelerators such as [8-9],[11], and [44-47]. In addition to the floating-point accelerators such as [7] and [57].

The proposed concept of using a preprocessing precision controller with approximate multiplier-based CNN accelerators can be utilized in a large system such as a cluster of inference accelerators. Such a cluster can contain hundreds or thousands of approximate multiplier-based CNN inference accelerators with various precisions and can be deployed

in a data center or in a cloud backbone. Once a CNN model is compiled on the cluster for inference, a small overhead of PEs can be allocated for the deployment of the preprocessing precision controller as well. As was presented in Table 4.1, the controller overhead is negligible and can be as low as 0.008% in the case of VGGNet19 and as high as 0.0.056% in the case of DenseNet201. In terms of the MACs, the overhead is as low as 0.0239% in the case of VGGNet19 and as high as 0.1404% in the case of DenseNet201. If needed, the controller can be mimicked to (i) number of controllers to allow for parallel inference of the compiled model. The controller training can occur as part of the model's compilation on the cluster or prior to that where the controller's parameters and shape can be provided alongside with the model's parameter and shape.

For each compiled network, (n) low precision accelerators (p) high precision accelerators can be allocated. The (n/p) ratio can be determined based on the expected usage of the low vs. high precision for that model. The allocated accelerators can be shared among different complied CNN networks. The cluster resource manager and scheduler will handle the sharing of the various accelerators among the various complied CNN networks. The design of hardware accelerators in data centers is discussed in [58]. Figure 4.11 provides a high-level illustration of how preprocessing precision controllers can be used with multiple approximate-multiplier based CNN accelerators with various precisions in a cluster.

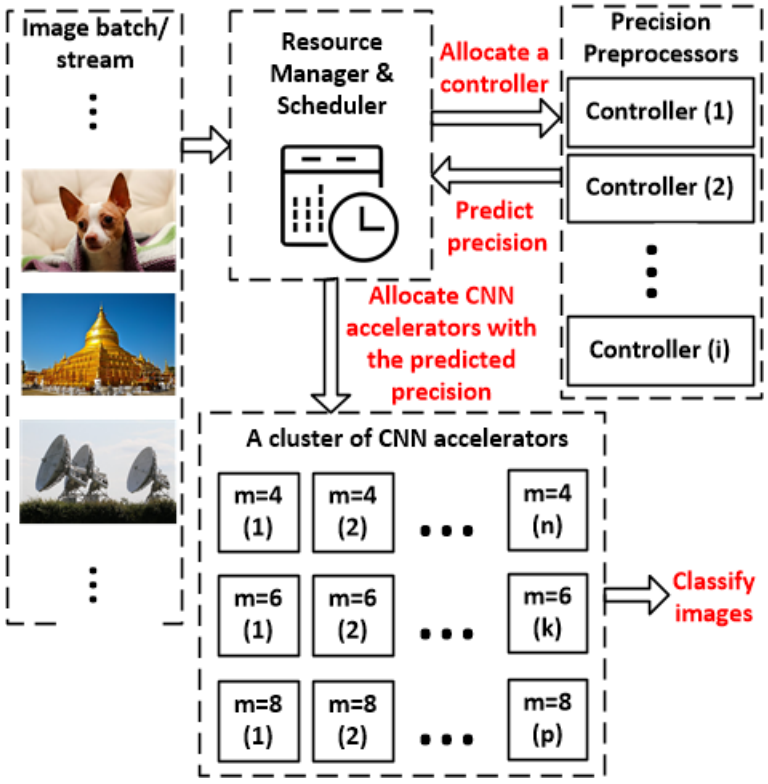

Figure 4.11: Using preprocessing precision controllers with multiple approximate-multiplier based CNN accelerators with various precisions in a cluster.

To demonstrate the performance benefits of the hybrid use of approximate multipliers with various precisions based on precision prediction, preprocessing precision controllers were trained using a subset of ImageNet for VGG19, Xception, and DenseNet201 by following the process in Figure 4.9 For each network both, the SSM and the DSM techniques were simulated. Table 4.6 details the performance gains in terms of power and speed that the hybrid use of approximate multiplier with various precisions can achieve compared to an exact multiplier. For VGG19 and Xception the controller was trained to select either a precision of m=4 or precision of m=8. In the case of the DenseNet201, the lower precision was selected as m=6 because m=4 had extremely low accuracy. As can be seen in Table 4.6, using a hybrid of approximate multipliers with various precisions can achieve large performance gains with minimal accuracy loss compared to exact multipliers. Overall, the SSM achieves significantly higher speed and lower power compared to the DSM in all the cases, however, the cost in terms of the network's accuracy loss is higher.

Table 4.6: Performance and accuracy for the hybrid use of approximate multipliers with various precisions compared to an exact multiplier

Type	Delay (ps)	Power (µW)	Speed Inc.	Power Reduct -ion	Accuracy	Accuracy Loss	Controller Accuracy	Low Precision Utilization
VGG-SSM (m=4&8)	174.42	51.72	89.66%	77.52%	72.13%	1.46%	79.4%	42.1%
VGG-DSM (m=4&8)	260.31	132.25	27.08%	42.52%	72.61%	0.98%	80.2%	42.7%
Xception-SSM (m=4&8)	180.73	55.21	83.03%	76.00%	78.20%	2.58%	78.1%	33.7%
Xception - DSM (m=4&8)	269.25	142.14	22.86%	38.23%	79.37%	1.41%	78.6%	34.2%
Dense-SSM (m=6&8)	191.96	58.86	72.33%	74.42%	78.76%	1.14%	77.9%	60.5%
Dense-DSM (m=4&8)	278.15	151.72	18.93%	34.06%	79.52%	0.38%	78.5%	62.1%

For each simulation in Table 4.6, the accuracy of the precision controller is listed as well. As can be seen from the table, the controller's accuracy ranges between 77.9% and 80.2% depending on the network, the segmentation type, and the precision modes. Therefore, the approximate multiplier precision prediction is not ideal. Nevertheless, the accuracy loss and performance gains calculations in Table 4.6 includes simulating this imperfection in the controller. Table 4.6 includes the low mode utilization percentage which represents the actual usage of the low precision mode including the false positives. If an ideal controller with 100% accuracy existed, the hybrid precision accuracy will surpass the accuracy of an exact multiplication. That is because a subset of image classes can be classified with better accuracy using lower precision approximate multipliers compared to higher precision approximate multipliers or even exact multipliers.

Using hybrid approximate multipliers with predicted precision can also achieve a better performance-accuracy trade-off compared to using a single-precision approximate multiplier. This is illustrated in Figure 4.12 where the performance gains and the accuracy loss compared to the exact multiplier are plotted for both designs. Figure 4.12 contains a

sub-plot for each scenario listed in Table 4.6. An example can be seen in Figure 4.12 (a), where the hybrid SSM multiplier with m=(4&8) achieved an 89.66% speed increase and a 77.52% power reduction which is better compared to the performance gains for the case of m=6 with an 81.01% speed increase and a 77.36% power reduction. Nevertheless, the network's accuracy loss was only 1.5% in the case of m=(4&8) compared to 8.7% in the case of m=6. From the energy perspective, the SSM multiplier with m=(4&8) can achieve an 88.15% energy reduction compared to 87.49% in the case of m=6. The energy of each multiplier was obtained by calculating the product of the power and the delay.

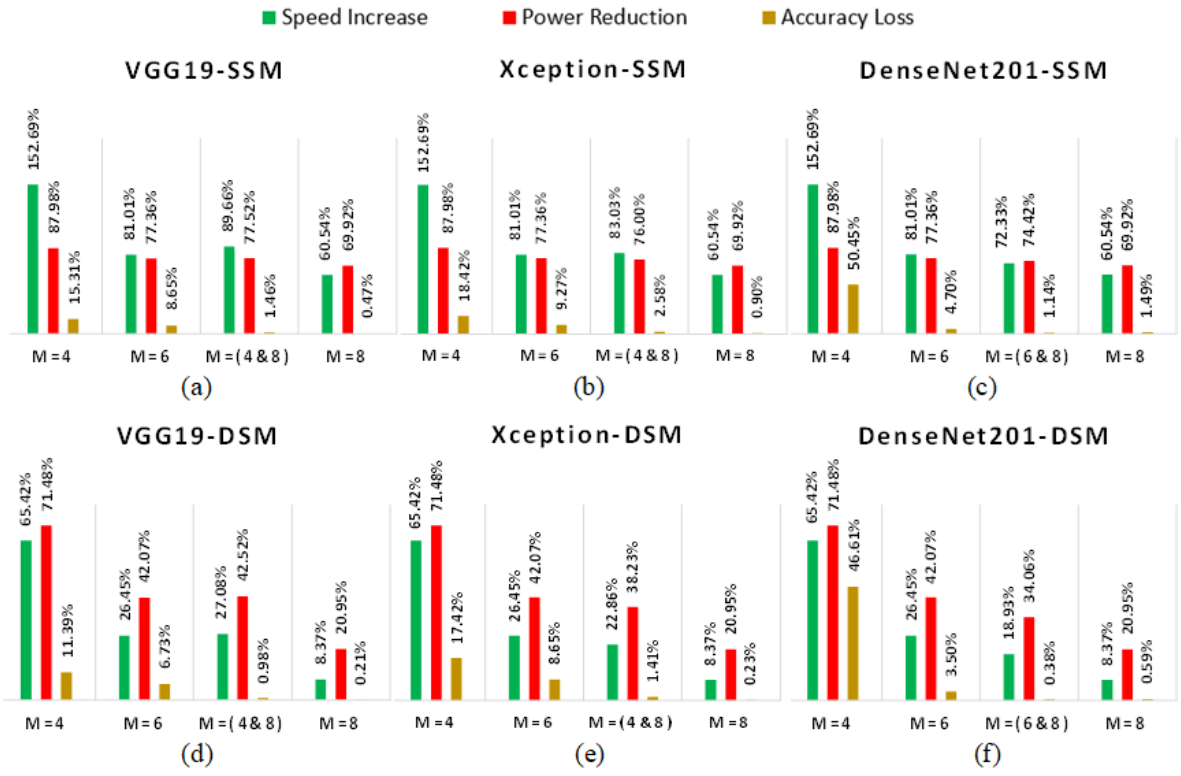

Figure 4.12: Performance gains and accuracy loss using hybrid and single precision approximate multipliers compared to an exact multiplier.

Another example can be seen in the case of DenseNet201 in Figure 4.12 (c) and (f), wherein both the SSM and the DSM, using hybrid approximate multiplier precisions with m=(6&8) achieved better performance gains with lower accuracy loss compared to the case of m=8. Having a lower accuracy loss in the hybrid configuration might be surprising. However,

this is due to having a subset of image classes which can be classified with better accuracy using m=6 compared to m=8 as explained previously.

On the cluster level, an additional performance advantage can be also achieved in terms of the total area. The area is proportional to the approximate multiplier precision as per Table 4.2 and Table 4.3. Therefore, building a cluster using (n) accelerators with high precision approximate multipliers only will require more area than a cluster which contains (n) accelerators built using various approximate multipliers precisions.

4.6.2 Single-Core Reconfigurable Architecture

Utilizing the proposed concept of predicting and configuring the precision of approximate-multiplier based CNN accelerators is not limited to the multi-core architecture. This section will present how a reconfigurable approximate multiplier with two precisions can be implemented to allow for the utilization of the concept in single-core systems. This will expand the proposed concept to include the design of approximate multiplier-based CNN accelerators for embedded systems and low power applications such as [59-60].

Figure 4.13 demonstrates how a reconfigurable approximate multiplier with two precisions can be implemented inside each MAC unit. Using this design either the high or the low precision approximate multiplier will be power up or down based on the precision controller signal “CN”. Tri-state buffers are used to control the multiplier's output flow.

As can be seen from Figure 4.13, this architecture requires implementing two approximate multipliers with high and low precisions inside the MAC. This will result in area overhead compared to a single-precision approximate multiplier design. Nevertheless, this hybrid reconfigurable precision design can achieve better performance in terms of speed and power compared to the single-precision design. Also, its total area is still less than an exact multiplier design. Optimizing the design in Figure 4.13 can be achieved by designing a merged reconfigurable approximate multiplier that supports two precisions.

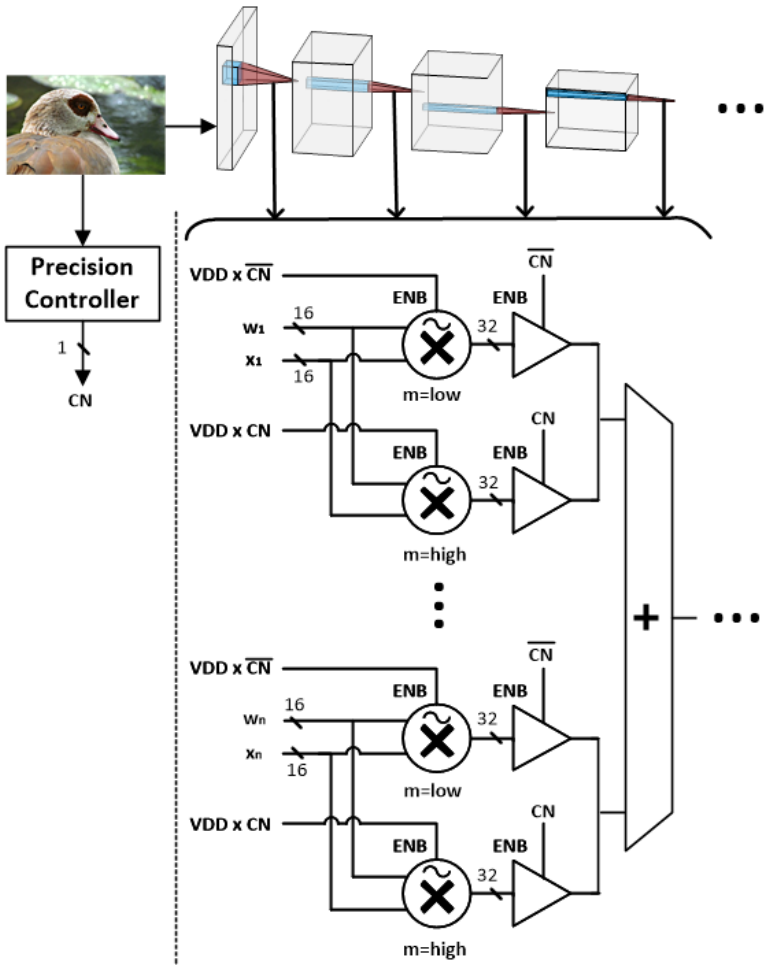

Figure 4.13: Reconfigurable design with two separated approximate multipliers.

Figure 4.14 shows a proposed design for a merged reconfigurable approximate multiplier that supports precisions of m=(4&8). As can be seen from the figure, the merged reconfigurable multiplier can be designed using four (4x4) bit multipliers, segmenters, a shifter, tri-state buffers for flow control, and an XOR for the sign bit. This multiplier will be partially turned off when m=4 is activated using a VDDxCN control signal allowing for higher speed and lower power execution. In terms of component sharing, one 4x4 multiplier is shared between the two precision, in addition to the shifter, the merged segmenter, and the sign bit XOR. Figure 4.14 illustrates a generic merged reconfigurable multiplier for both the SSM and DSM, the difference between these two techniques is contained in the design of the merged segmenter.

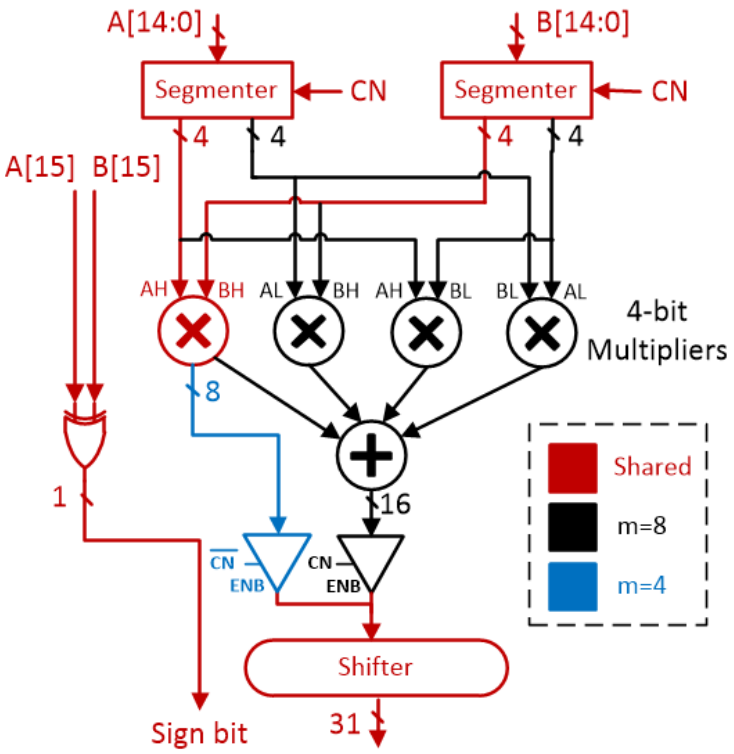

Figure 4.14: Reconfigurable design with a merged approximate multiplier for m=(4&8).

Table 4.7 lists the performance comparison between the merged and the separated reconfigurable design using VGG19 and based on the low precision utilization rate as per Table 4.6.

Table 4.7: Performance comparison for single-core the separated and merged reconfigurable designs using vgg19

Hybrid Design Type for m=(4&8)	**Seg. Type**	**Delay (ps)**	**Power (µW)**	**Area (µm²)**	**Speed Increase vs. Exact**	**Power Reduction vs. Exact**	**Area Reduction vs. Exact**	**Accuracy Loss**
Separated	SSM	175.240	52.641	171.100	88.77%	77.12%	49.23%	1.46%
Merged	SSM	181.239	55.223	114.930	82.52%	76.00%	66.20%	1.46%
Separated	DSM	262.312	137.203	332.114	26.11%	40.37%	1.46%	0.98%
Merged	DSM	273.432	146.410	214.980	20.98%	36.37%	36.21%	0.98%

Figure 4.15 Illustrates the performance-accuracy trade-off for various approximate multiplier precisions compared to an exact multiplier based on VGG19. As can be seen in the figure, the reconfigurable approximate multiplier achieves a better performance-accuracy trade-off compared to single-precision approximate multipliers. The SSM *m=(4&8)* merged design achieves an 82.52% speed increase, a 76% power reduction, and a 65.9% area reduction with a 1.46% accuracy loss compared to an exact multiplier. On the other hand, the DSM *m=(4&8)* merged design achieves a 20.98% speed increase, a 36.37% power reduction, and a 36.21% area reduction with a 0.98% accuracy loss compared to an exact multiplier. The merged *m=(4&8)* reconfigurable designs have a slight area overhead and a slightly higher accuracy loss compared to a single-precision design with *m=8*. Nevertheless, the performance gains in terms of speed and power are large.

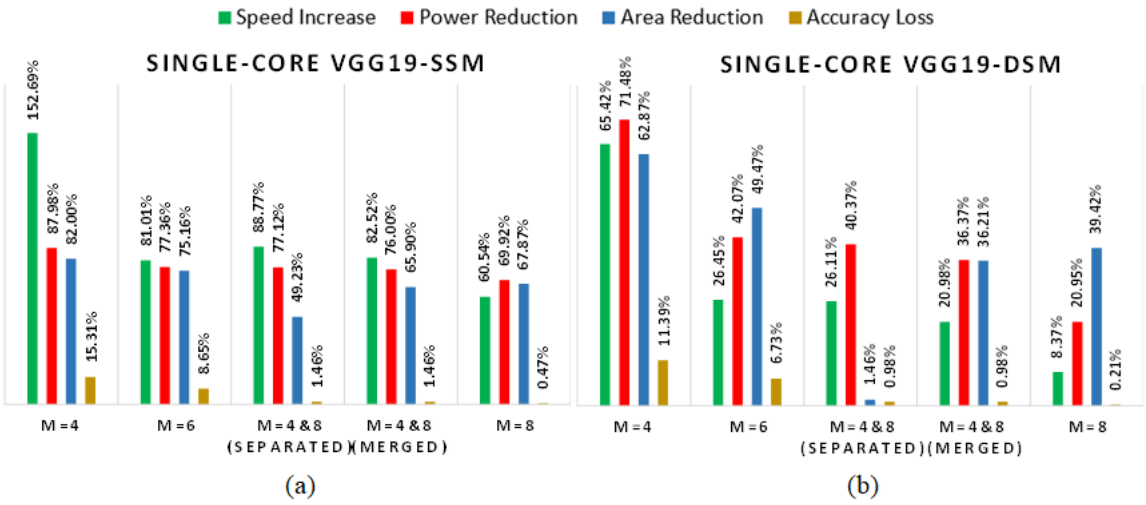

Figure 4.15: The single-core performance-accuracy trade-off for various approximate multiplier precisions compared to an exact multiplier based on VGG19.

Figure 4.16 illustrates the speed and power performance gains of the hybrid m=(4&8) designs compared to a single-precision design with m=8. As can be seen in the figure, the SSM m=(4&8) merged design can achieve a 13.7% speed increase and 25.35% power reduction compared to a single-precision design with m=8. For DSM, the gains are an 11.64% increase in speed and a 24.24% power reduction. This is an additional energy

savings of 29.83% in the case of merged SSM and 27.9% in the case of merged DSM. The cost in terms of the area overhead is 5.3% for DSM merged design and 6.1% for SSM merged design. While the accuracy loss difference is 0.77% for DSM merged design and 0.99% for SSM merged design.

For cases where reducing the area is less important, the hybrid separated design can be utilized to achieve higher speed and power lower and therefore lower energy consumption. The SSM separated m=(4&8) design can achieve a higher speed gain and a comparable power reduction compared to the case of m=6 with far less accuracy loss. The SSM separated m=(4&8) design achieved a speed increase of 88.77% and a power reduction of 77.12% and an accuracy loss of 1.46% which is better than m=6 with a speed increase of 81.01% and 77.36%. and an accuracy loss of 8.65%.

Overall, the merged hybrid design achieves more balanced power-speed-area gains compared to the separated hybrid design. Nevertheless, the merged design achieves lower gains in terms of the speed increase and the power reduction and therefore in energy saving. Hence, for designs that prioritize increasing the speed or reducing the energy consumption over reducing the area, the separated design might be a better option.

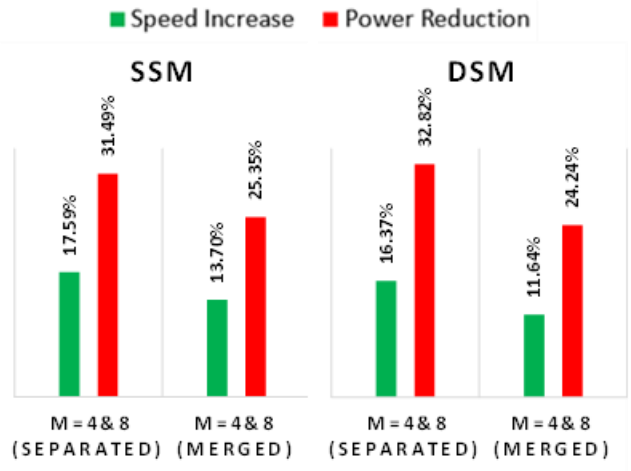

Figure 4.16: Hybrid m=(4&8) performance gains compared to a single-precision with m=8 based on VGG19.

4.7 Conclusion

This paper proposed a new architecture for improving the multiplication performance for CNN inference. The proposed architecture consists of a preprocessing precision controller at the system level and approximate multipliers with various precisions at the PE level. The proposed preprocessing controller determines the adequate precision for the network's approximate multipliers to classify the input image. The precision controller concept was inspired after discovering that a subset of image classes can be classified with the same or better accuracy using lower precision approximate multipliers. Based on this finding, the controller was trained to determine if the input image belongs to that subset or not. The controller is built using a tiny two-class CNN which has a negligible overhead in terms of parameters and MACs compared to the controlled image classification CNN. A performance analysis was presented based on the SSM and DSM methods using CMOS 15nm technology. Additionally, an accuracy analysis using VGG19, Xception, and DenseNet201 was presented. Overall, utilizing the SSM achieves a better performance-accuracy trade-off compared to using the DSM. As an example, for VGG-19 using a multi-core hybrid design with m=(4&8), the SSM achieves 89.66% speed increase and 77.52% power reduction with an accuracy loss of 1.46% compared to a speed increase of 27.08% and power reduction of 42.52% and an accuracy loss of 0.98% in the case of the DSM. Additionally, the total area of two DSM multipliers with m=(4&8) is 94.9% larger than two SSM multipliers with m=(4&8). Using the DSM is an option if minimizing the accuracy loss is critical for the designer. Maximizing the benefits of the proposed design can be achieved in a multi-core architecture with a large number of CNN inference accelerators, where the accelerators are built using different approximate multiplier precisions. Using the multi-core architecture, the proposed concept can achieve significant performance gains compared to designs with exact multipliers. Additionally, it can achieve a significantly better performance-accuracy trade-off compared to designs that uses single-precision approximate multipliers. The paper has also presented the performance benefits of utilizing the proposed concept in single-core designs. To optimize this utilization, a new merged approximate multiplier with two configurable precisions was proposed. The proposed concept of using a tiny preprocessing controller to determine the adequate

processing precision is not limited to the DSM and SSM architectures and can be expanded to other approximate multiplier designs, also it is expandable beyond approximate multipliers. The concept can be investigated in floating-point designs to control cores with variable precisions. It is also not limited to image classification and can be explored to improve the hardware performance of other problems such as video and audio classification.

CHAPTER 5 PRACTICAL CONSIDERATIONS FOR ACCURACY EVALUATION IN SENSOR-BASED MACHINE LEARNING AND DEEP LEARNING

Issam Hammad and Kamal El-Sankary

 Hammad, I.; El-Sankary, K. Practical Considerations for Accuracy Evaluation in Sensor-Based Machine Learning and Deep Learning. Sensors 2019, 19, 3491. https://doi.org/10.3390/s19163491

5.1 Abstract

Accuracy evaluation in machine learning is based on the split of data into a training set and a test set. This critical step is applied to develop machine learning models including models based on sensor data. For sensor-based problems, comparing the accuracy of machine learning models using the train/test split provides only a baseline comparison in ideal situations. Such comparisons won't consider practical production problems that can impact the inference accuracy such as the sensors' thermal noise, performance with lower inference quantization, and tolerance to sensor failure. Therefore, this paper proposes a set of practical tests that can be applied when comparing the accuracy of machine learning models for sensor-based problems. First, the impact of the sensors' thermal noise on the models' inference accuracy was simulated. Machine learning algorithms have different levels of error resilience to thermal noise, as will be presented. Second, the models' accuracy using lower inference quantization was compared. Lowering inference quantization leads to lowering the analog-to-digital converter (ADC) resolution which is cost-effective in embedded designs. Moreover, in custom designs, analog-to-digital converters' (ADCs) effective number of bits (ENOB) is usually lower than the ideal number of bits due to various design factors. Therefore, it is practical to compare models' accuracy using lower inference quantization. Third, the models' accuracy tolerance to sensor failure was evaluated and compared. For this study, University of California Irvine (UCI) 'Daily and Sports Activities' dataset was used to present these practical tests and their impact on model selection.

5.2 Introduction

The primary objective of solving a problem using machine learning is to obtain a model for generalized predictions. In production, when deploying a pretrained machine learning model for inference, it should be expected that this model will perform predictions with an accuracy close to the achieved test accuracy during prototyping. Large deviations between the reported test accuracy and the actual accuracy in production can be a serious design problem. Therefore, the test accuracy should consider any practical aspects or variables

that don't necessarily exist in the development but can occur in production. In the early stages of machine learning research, many papers reported the training accuracy as the model's prediction accuracy. This practice doesn't truly reflect how the model can generalize to new data, instead, it reflects how good the model can fit the training set, which often can be a case of overfitting. Achieving a low generalization error that characterizes prediction performance and avoids overfitting and underfitting is discussed with more details in [61] and [62]. One of the early research papers that emphasized the importance of using a separate test set for model evaluation is [63]. Nowadays, machine learning model accuracy evaluation is performed by splitting the available data into training, cross-validation, and test sets. The cross-validation set is usually used to tune the models' hyperparameters, while the test set determines the prediction accuracy of the model. In many instances, the dataset is split into training and testing sets only, where the test set is used for cross-validation and to determine the prediction accuracy.

Using the train/test split is very common and is considered an acceptable practice in machine learning research today. Many train/test split techniques are used in the literature such as k-fold cross-validation and Monte Carlo cross-validation (MCCV). These techniques were used in different sensor-based machine learning research problems such as [64–68]. These papers built different machine learning models for sensor-based problems and compared their accuracy using the common train/test split. It is common in machine learning to build multiple models using different algorithms for one problem, then determine the best model for that problem based on the top achieved test set accuracy. However, for sensor-based machine learning problems, models' accuracy and suitability comparison should take into consideration practical factors that can occur in production.

Presenting these practical production considerations and their impact on model selection in sensor-based problems is the focus of this paper. These presented practical tests include the impact of thermal noise on the models' inference accuracy, models' accuracy tolerance to lower inference quantization, and model tolerance to sensor failure. The next section will provide a detailed background on these practical tests and their purpose. The paper

will demonstrate that when considering these practical production problems, the decision regarding the appropriate machine learning model for a problem can be majorly impacted.

For this study, the University of California Irvine (UCI) 'Daily and Sports Activities' dataset was employed. The dataset was constructed by [64] and is posted on the online repository [69]. The dataset contains sensor readings from accelerometers, gyroscopes, and magnetometers which correspond to a number of physical activities performed by different participants.

This paper is divided as follows, Section 5.4 provides a background on the proposed practical accuracy tests and their role in model selection. Section 5.5 describes the details and structure of the dataset used in this study. Section 5.6 presents the achieved baseline accuracies for various machine learning models using k-fold train/test split. In Section 5.6 the experimental results for the proposed practical tests are demonstrated. Finally, Section 5.7 presents the research conclusions.

5.3 Background on the Proposed Practical Accuracy Tests

According to [70], it can be reasonably assumed that each accelerometer creates its own independent thermal noise. Based on [71], an accelerometer's thermal noise can be modeled as an additive zero-mean Gaussian noise. The presented thermal noise simulation in this paper will demonstrate that machine learning models can have significantly different levels of accuracy loss due to thermal noise. Another practical production aspect is finding the adequate inference quantization level and the best model for this level of quantization. With the rise of edge artificial intelligence (AI) technology, pretrained machine learning models are deployed directly on embedded hardware, which is often low-power. Lowering the inference quantization can reduce the hardware cost of the analog-to-digital converter (ADC) and other digital signal processing (DSP) components. Speed–power–accuracy trade-offs in high-speed complementary metal-oxide-semiconductor (CMOS) ADCs are detailed in [72]. Based on [72], higher bit accuracy requires larger devices that result in lower speed and/or higher power consumption. Evaluating models' inference accuracy

using different quantization levels will determine the adequate level and will impact the decision on the model selection. Also, it is known that the ADC effective number of bits (ENOB) is usually lower than the ideal number of bits [73]. Hence, it will be critical to evaluate models' tolerance with lower inference quantization to simulate this practical ADC problem. A final aspect to consider is the impact of sensor failure on machine learning inference accuracy. Models with more tolerance to sensor failure might be favorable for production. Also, redundancy or other failure mitigation solutions can be applied for sensors with greater impact on the accuracy. The simulation for all these practical problems is presented in Section 5.

5.4 Dataset Details and Simulation Tools

For this study, the UCI 'Daily and Sports Activities' dataset was employed. The dataset was published by [64] and is posted on the UCI online repository [69]. This dataset was constructed by using five Xsens MTx 3-DOF (degrees of freedom) orientation trackers.

Figure 5.1 illustrates the used orientation tracker which was developed by Xsens Technologies. A total of 8 participants, 4 males and 4 females aged 20–30 contributed in the construction of the dataset [69]. For each participant, the orientation trackers were placed on the torso (tracker #1), the right arm (tracker #2), the left arm (tracker #3), the right leg (tracker #4), and the left leg (tracker #5). Data from these orientation trackers were captured during 19 different physical activities that the participants performed. These 19 activities were [64]: (1) sitting, (2) standing, (3) lying on back, (4) lying on right side, (5) ascending stairs, (6) descending stairs, (7) standing in an elevator still, (8) moving around in an elevator, (9) walking in a parking lot, (10) walking on a treadmill in flat with a speed of 4 km/h, (11) walking on a treadmill with 15° incline with a speed of 4 km/h,(12) running on a treadmill with a speed of 8 km/h, (13) exercising on a stepper, (14) exercising on a cross trainer, (15) cycling on an exercise bike in horizontal position, (16) cycling on an exercise bike in vertical positions, (17) rowing, (18) jumping, and (19) playing basketball. This dataset presents a classification problem, where the input is constructed from the readings of the 5 orientation trackers, while the output represents one of the 19

physical activity classes. Based on the Xsens MTx manual [74], each orientation tracker contains a 3D accelerometer, 3D gyroscope, and a 3D magnetometer.

For analog to digital conversion, 16 bits ADC is used [74]. This configuration resulted in having 9 sensor readings per tracker and a total of 45 sensor readings per record. The dataset was constructed using a sampling frequency of 25 Hz with 5 s representing each labeled instance. Therefore, 125 records, with 45 sensor readings each, construct one labeled instance. Accordingly, each instance had 5625 attributes. The entire dataset contained 9120 instances. The dataset was balanced as there were 60 instances for each activity per participant.

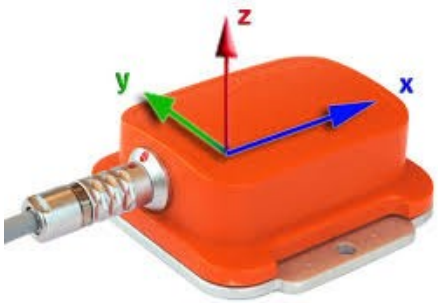

Figure 5.1: Xsens MTx 3-DOF (degrees of freedom) orientation tracker (photo from [74]).

This dataset was selected to study the proposed practical considerations for accuracy as it provides the raw data readings of the sensors. Also, it specifies that Xsens MTx orientation trackers [74] were used, which is unlike many machine learning datasets that don't include information on the hardware used. Providing the raw data and the hardware details facilitated the presented theory and simulations in this paper. Many research papers have proposed different machine learning models for this dataset, including the dataset publisher in [64-65]. In addition to that, several other papers proposed additional machine learning models and techniques such as [75–77]. The research goal was to demonstrate how the baseline accuracy achieved by the train/test split can be significantly impacted when considering the proposed practical tests such as the thermal noise impact, the impact of

lower inference quantization, and model's accuracy tolerance sensor failure. Model development and the simulation of the practical tests were implemented using the popular machine learning and data analysis python libraries: Keras [35], Sci-kit-learn [78], and NumPy.

5.5 Baseline Accuracy

Prior to presenting the proposed practical accuracy tests, baseline accuracies should be established using the basic train/test split. These baseline accuracies will be used later to demonstrate the impact of the proposed practical on the models' accuracy. Several machine learning models were trained as part of this research work on the dataset [69] using two different input sizes. One, by applying dimensionality reduction using principal component analysis (PCA), while the other keeps the original raw data size without any reduction. The number of attributes for each instance was reduced from 5626 to 30 when PCA was applied. Several popular machine learning algorithms were employed in this study. For each algorithm, multiple models with different hyperparameter settings were tested.

Table 5.1 lists the achieved test accuracy for the top model using each listed algorithm. Selecting these top models was achieved by iterating over multiple possible models representing different configurations for each algorithm, then performing the training and the cross-validation. The test accuracy results were logged and filtered, then the models with top test accuracies were trained again for confirmation. The purpose of the models' accuracies listed in Table 5.1 is to act as a reference when studying the impact of thermal noise, low quantization, and sensor failure on the overall accuracy of each model. The paper uses its own baseline model for the purpose of presenting the practical accuracy with the same training setting and testbench when using the baseline models for comparison. Therefore, any drop in the accuracy will be due to introducing the proposed practical tests. This determines how resilient the accuracy of these models is when considering these practical considerations.

Table 5.1 accuracies were obtained by applying k-fold cross-validation with k = 10. The data were divided as 90% for training and 10% for testing. This resulted in having 912 test instances in each k-fold iteration. As can be seen in Table 5.1, the deep neural network (DNN) model without PCA achieved the top accuracy, while the random forest classifier (RFC) without PCA achieved the second-best accuracy. The accuracy difference between the two models was 0.3%, which is considered very minimal and can be negligible. Additionally, the results indicated that using PCA increased the accuracy of some models, while for other models it had a negative impact. The DNN model was built using the popular deep learning [1] platform Keras [35], while the remaining machine learning models were built using scikit-learn [78]. Table 5.1 lists the DNN test accuracy for the best deep learning model. This DNN has five dense layers with the following sizes: Layer #1 (512 neurons), layer #2 (128 neurons), layer #3 (128 neurons), layer #4 (64 neurons), layer #5 (19 neurons). Layers 1–4 used ReLU activation function, while the output layer (layer #5) used a softmax activation function. Dropout was used for regularization and batch normalization was used in all the layers. For RFC, the model used 250 trees. For k-nearest neighbors (KNN) model, it was determined that the algorithm performs the best with eight neighbors. For the remaining algorithms, it was determined that the default scikit-learn settings achieved the best accuracy for each model.

Table 5.1: Baseline test accuracies using k-fold (k = 10).

Algorithm	**Train/Test Sample Size**	**Test Accuracy without PCA**	**Test Accuracy with PCA**
Deep Neural Network (DNN)	8208/912	99.26%	97.87%
K-Nearest Neighbors (KNN)	8208/912	78.34%	98.12%
Decision Tree Classifier (DTC)	8208/912	90.30%	90.72%
Random Forest Classifier (RFC)	8208/912	98.96%	98.65%
Gaussian Naïve Bayes (GNB)	8208/912	93.49%	78.55%

In the next section, the simulation results for the proposed practical accuracy tests will be presented. In order to have a manageable number of simulations that can be clearly

compared, one model per algorithm is used to present the practical accuracy simulations in this paper. The selection was done based on the model with higher accuracy, either including PCA or excluding it. An exception was applied in the cases of RFC and DTC as the test accuracies including and excluding PCA were extremely close, therefore, both models were included. This allows for an evaluation of the impact of dimensionality reduction when applying the proposed practical tests as well.

5.6 Experimental Results

This section presents the experimental results for the proposed practical accuracy tests. In the first set of tests, models' inference accuracy loss due to thermal noise was evaluated. This was achieved by simulating different levels of signal-to-noise ratio (SNR) for possible sensors' thermal noise. The results demonstrate that even though different machine learning models can have similar baseline test accuracies, their tolerance to the thermal noise can vary significantly. Therefore, the selection of the appropriate machine learning model should consider the expected levels of SNR that the sensors have.

The second set of tests evaluate models' accuracy tolerance to different quantization levels applied to the test set. The aim of these tests was to determine the adequate inference quantization level and the model that can achieve the highest accuracy at this level. Accordingly, for any custom design, the accuracy loss with lower inference quantization and the complexity of the model can be balanced against the required ADC resolution and the cost of any DSP components. Lower inference quantization also simulated the ADC ENOB problem. Models can have close baseline accuracies using high inference quantization; however, the accuracy loss with lower inference quantization can vary significantly from model to model. Additionally, a simulation for the inference accuracy with lower training quantization levels is presented. This determined whether or not a lower level training quantization is required to achieve better accuracy with lower inference quantization.

The third set of tests presents models' accuracy tolerance to sensor failure. Models with higher tolerance to sensor failure might be favorable for embedded designs. Such analysis will enable designers to evaluate the impact of a failure in a specific sensor or tracker on the models' accuracy. Therefore, more design constraints or a failure mitigation solution can be applied to sensors with higher impact on the accuracy.

The experimental results for the proposed practical accuracy tests are presented in the next three subsections. In Section 5.1, the simulation for the thermal noise impact on the accuracy is presented. Section 5.2 demonstrates the impact of low inference quantization on the accuracy, while in Section 5.3, the simulation results for the impact of sensor failure on the accuracy are presented.

5.5.1 Thermal Noise Simulation

Thermal noise, or Johnson–Nyquist noise, exists in all electrical circuits and it is caused by the random thermal motion of electrons. Thermal noise is approximately white with a Gaussian probability density function (PDF) amplitude [79]. Thermal noise is independent for each component. For example, in accelerometers, according to [70], each accelerometer has its own independent thermal noise. Also, based on [71], the thermal noise in accelerometers can be modeled as an additive zero-mean Gaussian noise. For gyroscopes, according to [80], the thermal noise in the gyroscope can be also modeled as Gaussian zero-mean independent noise. Even though the original dataset readings contain thermal noise as part of the reading, this noise is specific to the sensors used during the capture of the original dataset at the time of capture. During training, these captured sensor readings with this noise included will establish the foundations of any machine learning model. Therefore, the impact of thermal noise due to changes in the components, the timing, or the environment will impact the inference accuracy and not the training accuracy.

Examples can be provided from the literature for thermal noise SNR levels. For instance, in [81], a low noise accelerometer which can be used in medical applications can have a threshold of 20 dB SNR. Based on [70], SNR of 0 dB or above for 2D accelerometers is

considered good. In [82], an example is provided for search coil magnetometer with a thermal noise of 23 dB. Based on this information, a simulation for various levels of SNR resulting from adding zero-mean Gaussian noise is practical and realistic. For this simulation, various ranges of SNR were simulated, starting with 40 dB and going to 0 dB, with a 5 dB reduction between the test cases. These SNR values were simulated by adding a randomly generated zero-mean Gaussian noise with specific power to the test set.

Table 5.2 demonstrates the accuracy for machine learning models at each specific SNR value. The listed accuracy for each test case in Table 5.2 was obtained by averaging 25 random simulations of thermal noise for each k-fold. Hence, each listed accuracy resulted from averaging a total of 250 iterations. Figure 5.2 illustrates a histogram for a sample thermal noise distribution which was added to one accelerometer axis during one test iteration. Figure 5.2 was constructed from 11,400 points, resulting from adding the noise to one accelerometer axis for all 912 test instances. As previously mentioned, each instance represents a period of 5 s with a sampling frequency of 25 Hz. Figure 5.3 illustrates an example of thermal noise simulation for one accelerometer axis in one test instance. As can be seen from Table 5.2, the accuracy tolerance for models significantly varied due to thermal noise. For example, when comparing DNN and RFC, both models had very close baseline accuracies with a difference of 0.3% only. Therefore, a developer might prefer to deploy RFC over DNN based on certain design or performance aspects. However, when considering the models' tolerance to thermal noise, it becomes clear that DNN is superior over RFC. As an example, when considering 20 dB SNR level, the accuracy difference between these two models changed from 0.3% to 4.36%, which is significant. Another example can be seen when comparing KNN + PCA and RFC.

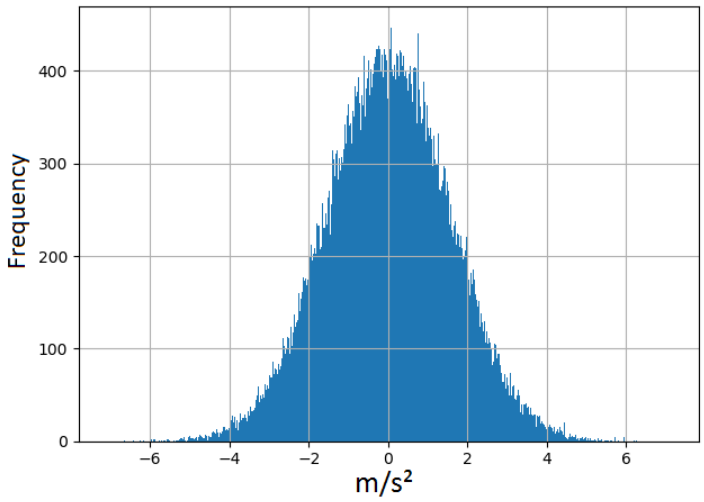

Figure 5.2: A histogram for a thermal noise sample added to one accelerometer axis in all test instances.

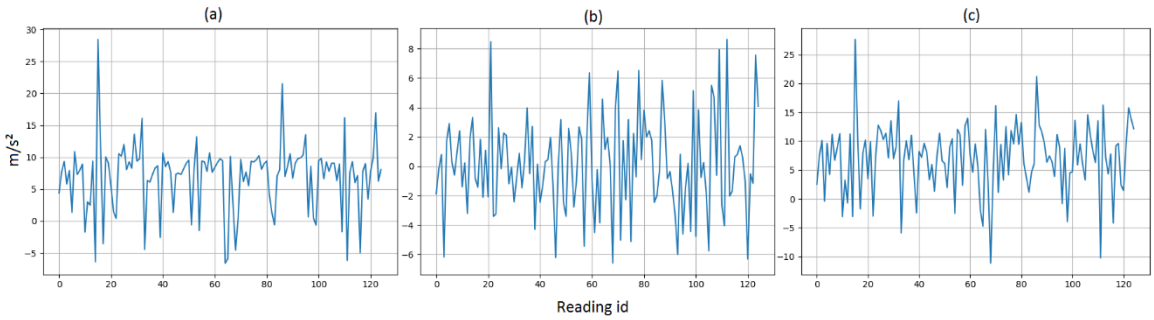

Figure 5.3: A sample for thermal noise simulation for one accelerometer axis in one instance with signal-to-noise ratio (SNR) of 5 dB.

(**a**) Original sensor readings.

(**b**) Added white noise.

(**c**) New values with SNR = 5 dB.

Table 5.2: Average inference accuracy with simulated thermal noise.

SNR	Machine Learning Model						
	DNN	**KNN + PCA**	**DTC**	**DTC + PCA**	**RFC**	**RFC + PCA**	**GNB**
Baseline	99.26%	98.12%	90.30%	90.72%	98.96%	98.65%	93.49%
40 dB	99.28%	98.11%	89.62%	90.54%	98.93%	98.59%	93.34%
35 dB	99.25%	97.97%	88.30%	90.47%	98.84%	98.51%	93.28%
30 dB	99.25%	97.98%	85.70%	89.89%	98.35%	98.44%	93.03%
25 dB	99.27%	98.02%	81.69%	88.90%	97.08%	98.24%	85.06%
20 dB	99.24%	98.03%	76.28%	87.53%	94.88%	97.60%	69.61%
15 dB	99.25%	98.01%	68.33%	84.79%	91.51%	95.74%	69.60%
10 dB	99.24%	97.82%	55.65%	80.77%	85.55%	92.35%	68.81%
5 dB	99.11%	97.73%	40.13%	74.56%	69.12%	86.90%	46.82%
0 dB	98.43%	96.37%	25.46%	63.98%	45.40%	77.61%	17.07%

According to the baseline accuracies, RFC is better than KNN + PCA. However, at 25 dB SNR, KNN + PCA surpassed the RFC model and the accuracy gap increased accuracy with higher noise Figure 5.4 shows the models' accuracy trend with the increase of thermal noise power. This analysis will enable designers to choose the appropriate model for their sensor-based machine learning problem according to the expected level of thermal noise. Also, the Figure provides a trade-off between the feasibility of using sensors with higher thermal noise, which has lower cost including the power and the willingness to have machine learning designs with lower accuracy.

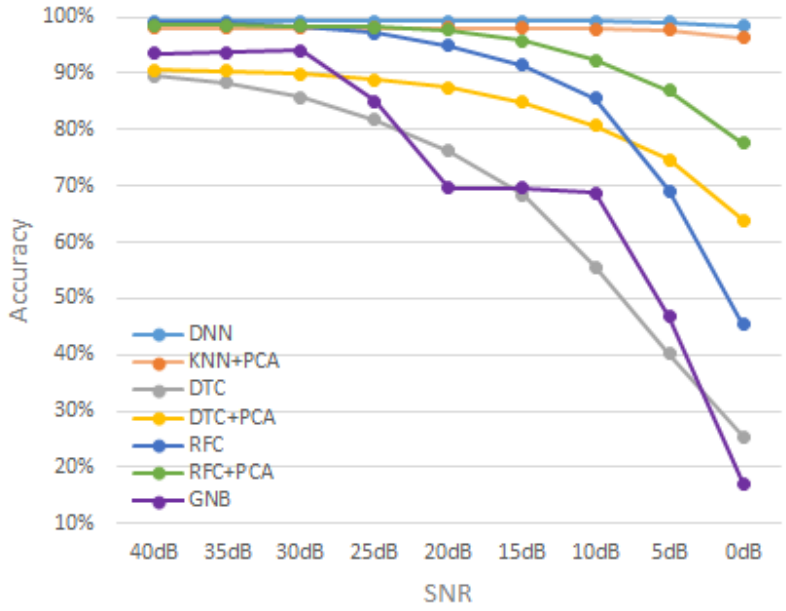

Figure 5.4: Accuracy trend for machine learning models with the increase of thermal noise power.

5.5.2 Quantization Levels Simulation

Lowering the inference quantization level can reduce the costs for possible embedded/edge AI implementations for the machine learning model. Using lower inference quantization will lower the resolution for the used ADC and the other DSP components. This will be cost-efficient in embedded implementations as lower resolution ADCs have lower power and higher bandwidth [72]. In addition to reducing the ADC resolution during inference, it is known that ADCs ENOB is usually lower than the ideal ADC bits due to the quantization error [73]. As an example, the 8 bits ADC in [73] achieved 6.6 ENOB. Therefore, applying inference on lower resolutions can simulate the ADC ENOB as well. The ADC itself, can't fully ensure the accuracy of results. Many factors including voltage reference, PCB layout, I/O switching, and analog source impedance can affect the overall ADC accuracy [83].

The proposed practical tests for lower inference quantization will answer the following questions. First, what is the accuracy loss for each model when lowering the inference quantization? This should allow the designer to determine the trade-off between the

possible ADC resolutions and accuracy loss in each model. It will also provide a practical accuracy evaluation when considering the ADC ENOB problem. Second, which model achieves the highest accuracy at the required quantization level? Third, is there a performance difference between training with high quantization, then applying inference with lower quantization vs. implementing both the training and the inference with lower quantization? Table 5.3 lists the models' inference accuracies using a range of simulated resolutions for the ADC. For the training stage, the quantization used the original dataset levels [69] at 16 bits, and the lower quantization was applied on the test set. Figure 5.5 demonstrates the simulation for 5 bits (32 levels) and 6 bits (64 levels) ADC for one accelerometer axis in one instance. In Figure 5.6 models' accuracy trend with lower inference quantization is illustrated.

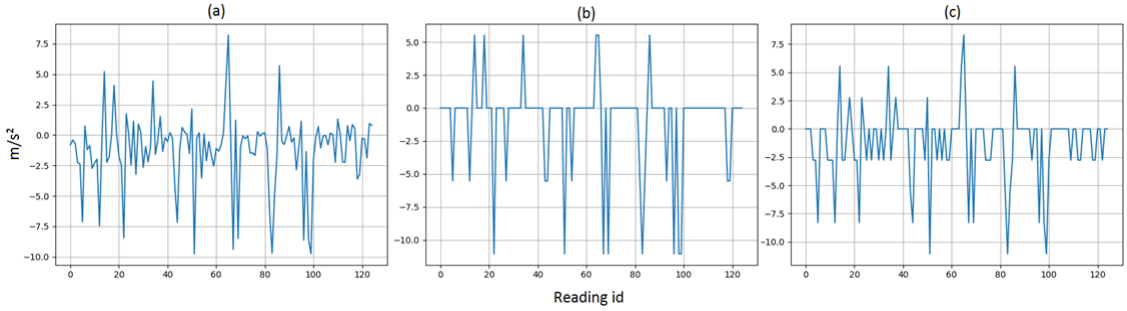

Figure 5.5: A sample for low quantization simulation for one accelerometer axis in one instance.

(**a**) Original sensors readings with 16 bits quantization.

(**b**) 5 bits quantization

(**c**) 6 bits quantization.

Table 5.3: Average inference accuracy using low resolution inference accuracy.

Resolution	Machine Learning Model						
	DNN	**KNN + PCA**	**DTC**	**DTC + PCA**	**RFC**	**RFC + PCA**	**GNB**
16 bits{baseline}	99.26%	98.12%	90.30%	90.72%	98.96%	98.65%	93.49%
14 bits	99.25%	97.95%	90.28%	90.68%	98.95%	98.61%	93.40%
12 bits	99.25%	98.02%	90.23%	90.64%	98.93%	98.61%	93.47%
10 bits	99.25%	97.99%	89.62%	90.31%	98.89%	98.56%	93.44%
8 bits	99.20%	97.93%	88.80%	87.30%	98.33%	97.50%	93.72%
7 bits	99.20%	97.74%	85.33%	83.68%	96.94%	94.53%	93.74%
6 bits	98.90%	95.48%	78.63%	76.33%	94.89%	88.11%	90.65%
5 bits	98.11%	89.12%	71.01%	63.81%	90.51%	76.29%	86.69%
4 bits	89.74%	60.91%	58.62%	38.89%	82.71%	54.52%	82.26%

As can be seen from Table 5.3 and Figure 5.6, DNN has a very high tolerance to lower quantization. For the DNN model, lowering the inference quantization from 16 bits to 6 bits resulted in an accuracy loss of only 0.36%. DNN also achieved higher tolerance to thermal noise as per the previous section. However, for the other models, the accuracy loss trend with lower quantization was different compared to the thermal noise impact, which was simulated in the previous section. This is due to the fact that the nature of these problems is different. The thermal noise is uncorrelated [84], while the quantization noise is partly correlated [85]. For example, the GNB model was intolerant to thermal noise, while it tolerated lower quantization relatively well compared to the other models. KNN + PCA showed more resilience in the case of thermal noise. On the other hand, RFC showed more resilience in the case of lower quantization. Additionally, it can be seen from Table 5.3 that models with dimensionality reduction using PCA have higher accuracy loss with lower quantization compared to models that excluded PCA. This can be seen when comparing DTC vs. DTC + PCA and RFC vs. RFC + PCA.

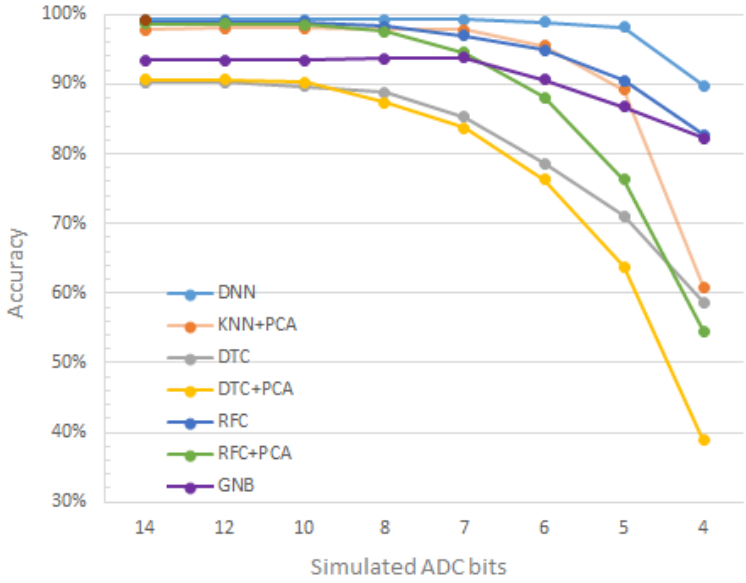

Figure 5.6: Accuracy trend in machine learning models with lower inference quantization.

Lowering the resolution of an ADC is directly proportional to its energy. An ideal N-bit ADC SNR is calculated as [52]:

$$SNR = 6.02\ N + 1.76\ dB.$$

Using Murmann's popular ADC survey [87]. A proportional relationship can be established between an ADC signal to noise and distortion ratio (SNDR) and its energy in picojoules (pJ). Hence, the number of bits in an ADC is directly proportional to its energy. Therefore, tolerating a lower inference quantization leads to a reduction in the required conversion energy for the ADC in any custom design. For embedded and low power applications, lowering quantization is primarily beneficial for inference and not for training. For such applications, it is expected that training is done using powerful computers, then machine learning models can be deployed on chip for inference.

Nevertheless, some models require training with lower quantization to achieve better accuracies with low inference quantization. Table 5.4 lists the achieved inference accuracy when applying lower quantization to the training phase as well. As can be seen, the accuracies were different compared to Table 5.3. For example, DNN with 5 bits inference quantization achieved an accuracy of 98.11%. However, if the 5 bits quantization is applied for both training and inference, the accuracy drops to 89.29%. Like DNN, GNB performed better when lower quantization was applied during the inference level only. On the other hand, DTC and RFC performed better when the lower quantization was applied for training as well. In summary, for low inference quantization, it will be critical to determine if low quantization is required during the training stage or not. This will vary from model to model, as can be seen. It is advisable in such cases to capture the dataset using high resolution ADCs, then based on the selected model, decide if the training set quantization level should be lowered or not.

Table 5.4: Average inference accuracies using lower resolution quantization applied to training and testing.

Model	**Simulated ADC Bits**				
	8 bits	**7 bits**	**6 bits**	**5 bits**	**4 bits**
DNN	99.19%	98.72%	98.54%	89.29%	81.87%
KNN + PCA	98.03%	95.29%	91.56%	72.48%	51.65%
DTC	89.06%	86.32%	87.41%	79.96%	75.35%
DTC + PCA	86.29%	81.91%	79.17%	50.88%	36.73%
RFC	98.80%	97.48%	97.70%	92.77%	89.81%
RFC + PCA	96.24%	94.59%	87.91%	67.73%	41.31%
GNB	85.62%	82.55%	81.67%	54.04%	32.11%

5.5.3 Impact of Sensor Failure on the Accuracy

Sensor failure is common and can occur at any time. In [88], statistics regarding sensor failures in smart homes are presented. Based on [88], the most common failure modes for sensors in smart homes are: Data link loss, either wired or wireless; dead battery or the loss of power' and loss of internet connection. Also, sensors can fail due to different mechanical issues. Hence, a practical machine learning design could require a certain level of tolerance to sensor failure during inference. Models with higher accuracy tolerance to sensor failure might be preferable for production. Also, such analysis will allow designers to apply a failure mitigation solution for sensors with greater impact on the accuracy. This includes using hardware redundancy or using higher quality sensors. Table 5.5 lists the accuracy for each model given a device in a tracker has failed. The table assumes one device (accelerometer, gyroscope, or magnetometer) with all 3-dimensional sensors has failed in one of the trackers. The listed accuracies are based on averaging all possible failure scenarios for each device type. Device failure is simulated by setting the device 3-dimensional readings to zero in the test set.

As can be seen from Table 5.5, the gyroscope has the least influence on the accuracy in all the models. Therefore, for production, using a lower quality gyroscope might be acceptable. On the other hand, depending on the model, the accelerometer, the magnetometer, or both, have a great impact on the accuracy. For example, the failure of a magnetometer had the greatest impact on the DNN's accuracy. However, the case was different for models with PCA, where the magnetometer had significantly less impact compared to the accelerometer. Therefore, based on the model, applying a failure mitigation solution for sensors with the greatest impact on the accuracy could be an option. This will ensure that the final design in production has a greater accuracy tolerance towards such failures.

Table 5.6 lists the inference accuracy for each model, assuming an entire tracker with all its nine sensors has failed. The same analogy that was applied to Table 5.5 can be applied

to Table 5.6. Based on the selected model, one or more trackers with the greatest impact on the accuracy can have a failure mitigation solution.

Table 5.5: Inference accuracy with a device failure in one tracker.

Model	Failed Device		
	Accelerometer	**Gyroscope**	**Magnetometer**
DNN	93.75%	98.81%	83.92%
KNN + PCA	64.42%	94.77%	94.74%
DTC	63.98%	90.28%	66.26%
DTC + PCA	30.46%	90.72%	90.48%
RFC	87.55%	98.82%	82.58%
RFC + PCA	41.38%	98.26%	96.26%
GNB	76.87%	92.79%	86.08%

Table 5.6: Inference accuracy with one tracker failure.

Model	Failed Tracker				
	#1	**#2**	**#3**	**#4**	**#5**
DNN	74.29%	86.15%	72.59%	74.69%	68.65%
KNN + PCA	78.08%	48.69%	51.07%	71.53%	72.16%
DTC	38.6%	66.89%	55.27%	64.37%	16.15%
DTC + PCA	27.89%	33.43%	30.63%	31.56%	28.9%
RFC	57.73%	82.69%	73.26%	88.59%	39.62%
RFC + PCA	42.98%	40.48%	38.88%	43.84%	40.25%
GNB	57.21%	83%	67.55%	63.45%	64.5%

5.7 Conclusion

This paper proposed a set of practical tests that can be applied to compare the accuracy of sensor-based machine learning models. To select an appropriate machine learning model

for production in a sensor-based application, several practical aspects should be considered beyond the basic train/test accuracy comparison. Using the UCI 'Daily and Sports Activities' dataset, these practical aspects were presented. First, in production, sensors' independent thermal noise will impact the models' inference accuracy negatively. Therefore, practical evaluation of the models' accuracy should consider the expected level of sensor's thermal noise. By simulating different levels of SNR, it was demonstrated that models' accuracy tolerance to thermal noise can vary significantly from model to model. Consequently, the decision on the appropriate model for deployment in production could be impacted. As an example, at 20 dB SNR, DNN had an average accuracy loss of 0.02%, while RFC had an average accuracy loss of 4.08%. Both models had a close baseline train/test accuracy with only 0.3% difference. The second presented practical tests aimed to find the adequate inference quantization level. For embedded AI applications, lowering inference quantization leads to lowering the required ADC resolution. Additionally, ADC ENOB is usually lower than its ideal number of bits. Accordingly, a simulation of lower inference quantization addressed both problems. Simulation results showed that the models' accuracy tolerance to lower inference quantization can vary significantly. DNN had the lowest accuracy loss using low inference quantization levels. DNN achieved an accuracy of 98.11% with only 5 bits quantization, which is only a 1.15% of accuracy loss compared to 16 bits quantization. The simulation results also showed that the models' dimensionality reduction using PCA were intolerant to lower inference quantization levels. Additionally, some models required lower training quantization levels to achieve better accuracies using lower inference quantization levels. This could be seen in the cases of the RFC and the DTC models, which were in contrast with the DNN and the GNB models. Therefore, to lower the ADC resolution, lower quantization should either be applied during both the training stage and inference stage or during inference only. This varies based on the model. Finally, the impact of sensors failure on the models' accuracy was presented. The proposed sensor failure tests can help designers in selecting models with higher accuracy tolerance to such failures for deployment in production. Also, it will allow designers to apply failure mitigation solutions on sensors with greater impact on the model's accuracy. While the UCI 'Daily and Sports Activities' dataset was used in this

paper, the proposed practical accuracy tests are generic and are not limited to this dataset. The same practical accuracy tests can apply to any sensor-based machine learning problem.

CHAPTER 6 USING MACHINE LEARNING FOR PERSON IDENTIFICATION THROUGH PHYSICAL ACTIVITIES

Issam Hammad and Kamal El-Sankary

 I. Hammad and K. El-Sankary, "Using Machine Learning for Person Identification through Physical Activities," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020, pp. 1-5, doi: 10.1109/ISCAS45731.2020.9181231.

6.1 Abstract

In this paper, the concept of utilizing machine learning algorithms for person identification through physical activity is proposed. Many previous machine learning research articles focused on building models to identify physical activities using a sensor fusion input. Nevertheless, there has been no focus on building models that can identify the activity performer as well. This paper will demonstrate that machine learning can be applied not only for the identification of physical activities but also for the identification of the activity performer as well. The paper will present the achieved accuracies for the person identification through physical activities using different machine learning algorithms. Additionally, a novel multi-label shared deep neural network (DNN) is proposed for identifying both the physical activity and the activity performer simultaneously. The proposed design allows for a single training/re-training which is advantageous over having to train two separate DNNs. Moreover, it is 30% smaller compared to a design that consists of two separate DNNs for identifying the physical activity and the activity performer.

6.2 Introduction

The last decade has witnessed a surge in the development of machine learning algorithms for various science and engineering problems. One of these problems is the classification of physical activities based on a sensor fusion. The University of California Irvine (UCI) 'Daily and Sports Activities' dataset [69] provides an example for a classification problem based on physical activities. The dataset contains a sensory input from five orientation trackers that were placed on the participant's bodies and the corresponding output which presents the performed physical activity. Each orientation tracker contains readings from 3-axis accelerometer, gyroscope, and magnetometer units. The dataset was constructed by collecting the orientation trackers data during the execution of 19 different physical activities by 8 participants. The dataset [69] was published by the authors of the research papers [64-65]. The focus of [64-65] was to find the most accurate machine learning algorithm for the classification of these 19 activities. Additionally, a large number of research papers which followed [64-65] used the dataset in [69] primarily for the same goal

which is building machine learning models for the activity classification. Examples can be seen in the research papers [19] and [89-91]. Moreover, additional examples can be found for research articles that also focused on the principle of activity classifications using other single or multiple physical activity datasets including the dataset [69]. This can be seen in research papers [65], [75-77], and [91-93]. Unlike the research articles [19],[64-65],[75],[77], and [89-93] this paper's focus is to evaluate and compare the accuracy of various machine learning algorithms for person identification through physical activities using a sensor fusion input. The dataset [69] will be utilized for this purpose. Additionally, the paper will present a novel multi-label shared deep neural network (DNN) design which can be used for identifying the physical activity and the performer in parallel.

Having reliable machine learning models for person identification through physical activity can have a major impact on the manufacturing of activity trackers and the field of wearable technology in general. In such a case, a user can be automatically detected through the performed physical activity rather than using manual entry. Also, person identification through physical activity can have applications in the field of biomedical engineering.

This paper is divided into the following sections. In Section 6.3 a description of the dataset structure details is provided. The section also discusses the used simulation tools in this study. Section 6.4 presents the achieved accuracy results for person identification through physical activities using different machine learning models. The section includes a comparison of the achieved physical activity identification accuracy and person identification accuracy in each model. Section 6.5 demonstrates the design details of the proposed multi-label shared DNN for simultaneous identification of the physical activity and the performer. Finally, in section 6.6 the research conclusions are presented.

6.3 Description of the Used Dataset and Tools

The dataset [69] is used to present the concept of utilizing machine learning algorithms for person identification through physical activity. Each row in the dataset contains readings

from five Xsens MTx 3-Degrees of Freedom (DOF) orientation trackers. Figure 6.1 illustrates the used orientation tracker which is developed by Xsens Technologies [74].

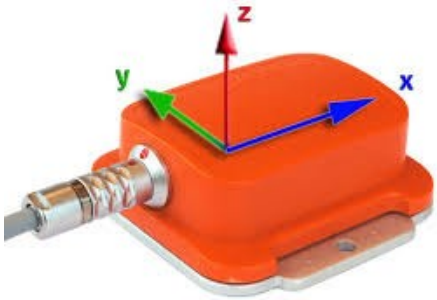

Figure 6.1: Xsens MTx 3-DOF orientation tracker [74] (courtsey of Xsens).

Each Xsens orientation tracker contains 3-axis accelerometer, gyroscope, and magnetometer units. Therefore, each row in the dataset contains readings from 45 sensors representing 9 sensor readings from each sensor. During the construction of this data set, these five orientation trackers were placed on the participant's torso (T1), right-arm (T2), the left arm (T3), right leg (T4), and left-leg (T5). Figure 6.2 demonstrates a visualization for the trackers' locations.

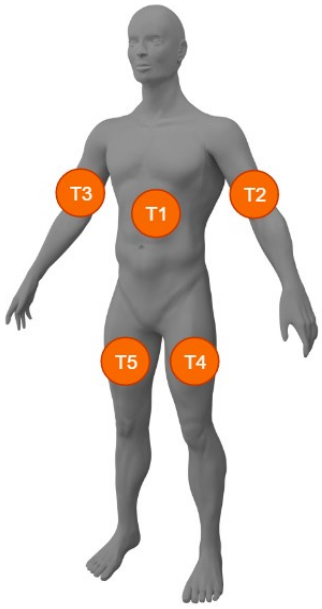

Figure 6.2:Xsens tracks location on the participants' bodies

The dataset was constructed using a total of 8 participants, 4 males and 4 females aged 20-30, where each of these participants performed 19 different physical activities. Table 6.1 lists all the physical activities that the participants performed during the construction of this dataset.

Table 6.1: List of the performed activities during the dataset construction

ID	Description
A1	Sitting
A2	Standing
A3	Lying on back
A4	Lying on right side
A5	Ascending stairs
A6	Descending stairs
A7	Standing in an elevator still
A8	Moving around in an elevator
A9	Walking in a parking lot
A10	Moving around in an elevator
A11	Walking on a treadmill in flat with a speed of 4km/h
A12	Walking on a treadmill with 15° incline with a speed of 4km/h
A13	Exercising on a stepper
A14	Exercising on a cross trainer
A15	Cycling on an exercise bike in horizontal position
A16	Cycling on an exercise bike in vertical positions
A17	Rowing
A18	Jumping
A19	Playing basketball

A sampling frequency 25Hz was used during the capture of the sensors' readings. Each labeled instance in the dataset represents a 5 seconds duration of physical activity. Hence, each instance has 5625 attributes representing 125 rows with 45 sensors readings in each row. The dataset contains a total of 9120 5-second instances. These instances were divided equally between the 8 participants, where each participant has 1140 instances with 60 instances for each activity. For models development, the python libraries Keras [35], Sci-kit-learn [78] and NumPy were used. Keras was used to build the proposed Deep Neural Network Designs (DNN), while Sci-kit-learn was used to build the other machine learning models.

6.4 Person Identification Experimental Results

Various machine learning and deep learning models were trained to assess their applicability for person identification through physical activity. This included training and evaluating Deep Neural Network (DNN) models, K-Nearest Neighbors (KNN) models, Decision Tree Classifier (DTC) models, Random Forest Classifier (RFC) models, and Gaussian Naïve Bayes (GNB) models. For each category, different model settings were evaluated by tuning the model's hyper-parameter. These models were trained using two input modes. In the first mode, dimensionality reduction was applied using Principle Component Analysis (PCA), while in the second mode the raw data from the trackers were used as-is. In the case of the dimensionality reduction, the input size was reduced from 5625 to 30 for each instance, while in the second mode the input size was kept as 5625. All tests were applied using k-fold cross-validation using k=10 with a data split of 90% for training and 10% for testing. Applying this 90/10 split resulted in having 8208 training instance and 912 testing instance in each k-fold.

Table 6.2 lists the average achieved test accuracies using raw input size, while Table 6.3 lists the average achieved test accuracies with PCA on the input. Both tables list the achieved accuracies for the person classification which was obtained during this study and the activity classification accuracies as per [19].

Table 6.2: Average accuracies using raw data input

Algorithm	Person Classification Accuracy	Activity Classification Accuracy [19]
Deep Neural Network (DNN)	95.81%	99.26%
K-Nearest Neighbors (KNN)	79.08%	78.34%
Decision Tree Classifier (DTC)	75.03%	90.30%
Random Forest Classifier (RFC)	91.29%	98.96%
Gaussian Naïve Bayes (GNB)	37.91%	93.49%

Table 6.3: Average accuracies using PCA input

Algorithm	Person Classification Accuracy	Activity Classification Accuracy [19]
Deep Neural Network (DNN)	94.86%	97.87%
K-Nearest Neighbors (KNN)	91.40%	98.12%
Decision Tree Classifier (DTC)	85.75%	90.72%
Random Forest Classifier (RFC)	91.24%	98.65%
Gaussian Naïve Bayes (GNB)	28.51%	78.55%

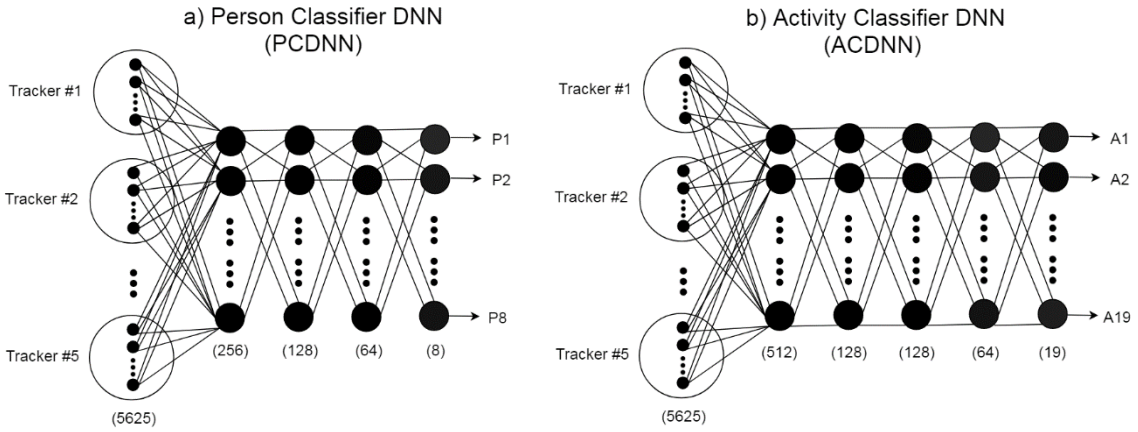

Figure 6.3: Proposed DNNs for person classification and activity classification
(a) Person classificatier DNN (PCDNN)
(b) Activity classifier DNN (ACDNN) [4]

As can be seen from Table 6.2 and Table 6.3, the DNN model with the raw input achieved the highest person classification accuracy with an average of 95.81%. This result is homogenous with the best design for activity classification as per [19], where a DNN design with raw input has achieved an average accuracy of 99.26%.

Figure 6.3 (a) illustrates the proposed person classifier DNN (PCDNN), while Figure 6.3 (b) illustrates the top activity classification DNN (ACDNN) as per [19]. As can be seen from the figure, the proposed PCDNN has an input size of 5625 representing the 5 seconds input from the 45 sensors with a sampling rate of 25 Hz. This is followed by 4 fully

connected layers with the sizes of 256, 128, 64, and 8 neurons, respectively. The proposed person classifier CNN (PCDNN) design in Figure 6.3 (a) included using batch normalization at each network layer and having a 15% dropout for regularization. The network has used rectified linear unit (ReLU) activation function in all the hidden layers while softmax activation function was used in the output layer. Besides this proposed DNN, other models have achieved acceptable accuracies for both person classification and activity classification. For example, RFC achieved the second top accuracy after DNN with average accuracies of 91.29% and 98.96% for person and activity classifications respectively.

Figure 6.4 illustrates models accuracy trends for the activity classification and the person classifications using raw and PCA inputs. Based on Figure 6.4, it can be clearly seen that some models have failed in terms of person classification even though they had an acceptable success in terms of activity classification. An example can be seen in the case of the GNB model where an average activity classification accuracy of 93.49% was achieved. However, the accuracy has dropped to 37.91% for person classification. Overall, machine learning models achieved higher activity classification accuracies. Also, DNN with the raw input was found to be the best model in terms of accuracy for both activity classification and person classification.

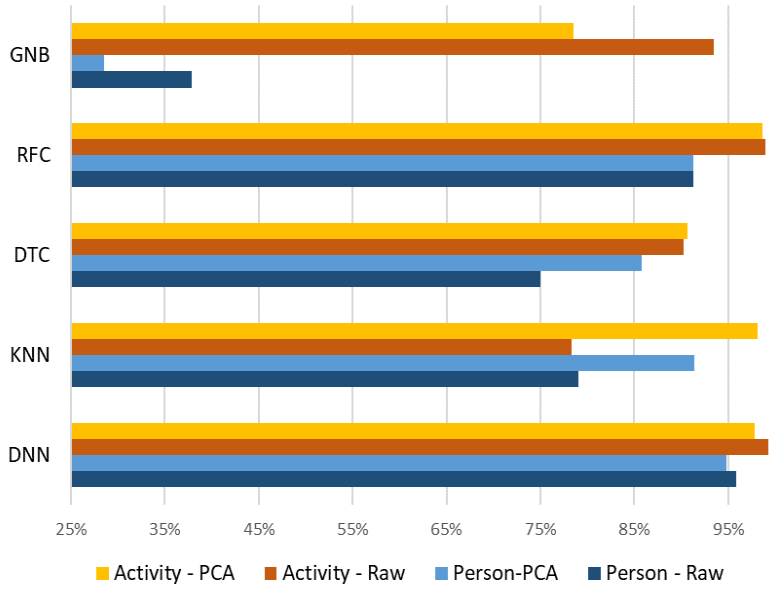

Figure 6.4: Models accuracy trends for the activity classification and the person classifications using raw and PCA inputs.

6.5 Multi-Label Shared DNN for Activity and Person Classification

The previous section demonstrated that using DNN models with the raw input has achieved the best accuracy for both person classification and activity classification. Accordingly, one critical question can be raised, can both DNNs be combined into one multi-label shared DNN which can identify the activity and the activity performer simultaneously?

To answer this question, this paper proposes the DNN design which is illustrated in Figure 6.5. This design represents a multi-label shared DNN which can identify the activity and the activity performer simultaneously. This DNN has an input size of 5625 representing the readings from the 5 trackers.

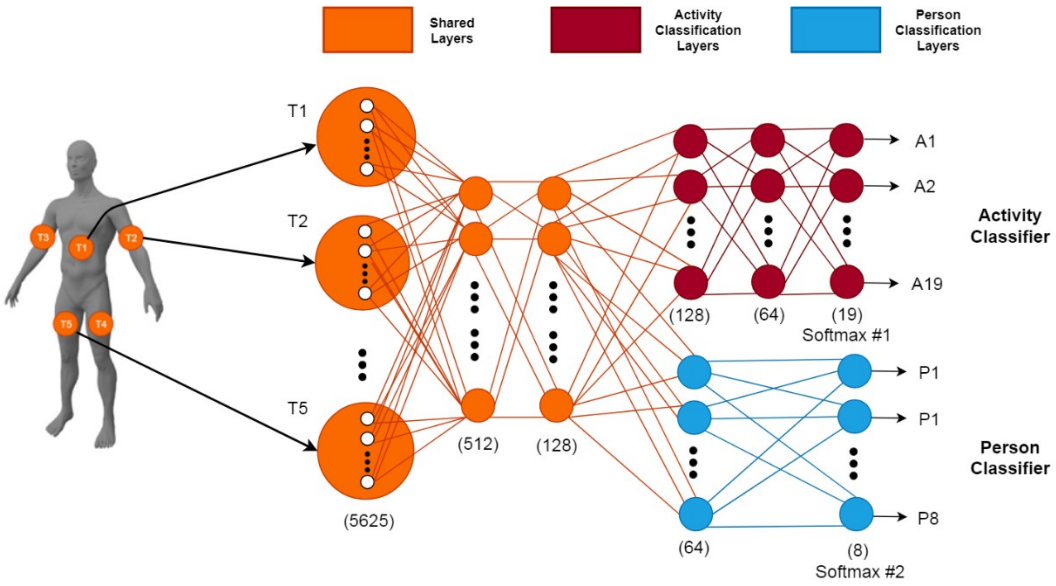

Figure 6.5: Proposed multi-label shared DNN for simultaneous classification for the physical activity and the activity performer

As mentioned earlier, each tracker has 9 sensors and the sampling rate is 25Hz based on 5 seconds instances. The first two layers in this design are shared and have 512 and 128 neurons, respectively. Following the 128-neurons layer, the neural network branches into two sections. One section is responsible for identifying the activity performed and the other is responsible for identifying the activity performer. The activity identification sub-network consists of 2 hidden layers with sizes of 128 and 64 neurons, and one output softmax layer with the 19 neurons representing all possible activities. The person identification sub-network has only one 64 neurons hidden layer and one 8 neurons softmax output layers representing all the possible performers.

Similar to proposed models in the previous section, this network was trained and tested based on applied k-fold cross-validation using k=10 with a 90/10 split. In this network batch normalization and a dropout regularization of 10% were used. The network has used ReLU activation function in all the hidden layers. The network has achieved an average accuracy of 99.24% for the activity classification and an average accuracy of 96.32% for the person classification. While the activity classification accuracy had a negligible drop

of 0.02% compared to the stand-alone ACDNN, the average accuracy for the person classification has increased by 0.51% compared to the PCDNN. This accuracy enhancement in the person classifier was as a result of having activity labels available during the training of the shared DNN. Table 6.4 summarizes the differences between the ACDNN, PCDNN and the shared DNN in terms of size and accuracy.

Table 6.4: Comparison between the ACDNN, PCDNN, and the shared DNN

Architecture	Activity Classification Accuracy	Person Classification Accuracy	Total Neurons
ACDNN	99.26%	N/A	851
PCDNN	N/A	95.81%	456
Two separate DNNs [ACDNN + PCDNN]	99.26%	95.81%	1307
Shared DNN	99.24%	96.32%	913

In addition to the improvement in the person classifier accuracy, the shared DNN has 913 neurons which is a 30% smaller design compared to a two separate ACDNN and PCDNN with a total of 1307 neurons. Moreover, having a shared DNN design allows for simultaneous training and inference which is a significant improvement and replaces the need of training and deploying two separate networks.

6.6 Conclusion

This paper presented the concept of utilizing machine learning algorithms to construct a person identification system through physical activity. UCI Daily and Sports Activities dataset was used to train and evaluate different machine learning models to evaluate the applicability of this concept. The experimental results showed that using a DNN model achieves the best accuracy compared to other machine learning models with an average accuracy of 95.81%. Moreover, the paper proposed a novel multi-label shared DNN network which can identify the activity performed and the activity performer simultaneously. The shared DNN is 30% smaller in size compared to using two separate DNN networks for identifying the activity and the activity performer. Moreover, the proposed shared DNN allows for simultaneous training and inference. The presented

research findings in this paper can be utilized in the fields of wearable technology and biomedical engineering.

CHAPTER 7 CONCLUSION

7.1 Thesis Conclusion

This thesis has presented several research contributions which focused on utilizing approximate computing in the design of DNNs. The primary focus of the thesis was on studying the applicability of approximate multipliers in DNN design and proposing new methods to maximize the performance gains in terms of power, speed, and area while minimizing the accuracy loss. The thesis also focused on other concepts which allow for lower power and higher speed designs such as low quantization and shared DNNs. Additionally, the thesis filled a gap in terms of addressing practical considerations for accuracy evaluation in the production of sensor-based machine learning and deep learning designs. These aspects include the accuracy loss due to the component's variable thermal noise, component failure or partial failure, and analog-to-digital converter (ADC) quantization error. The proposed solutions in this thesis can be utilized in both low-power designs such as battery-operated devices and in high-speed servers in data centers. The thesis included research contributions from three published journal papers and two published conference papers.

Chapters 2-4 demonstrated how CNN designs can be optimized using approximate multipliers. Approximate multipliers can achieve significant performance enhancements in terms of power, speed, and area while having minimal impact on the accuracy. Chapter 2 presented a study on the impact of approximate multipliers on the inference accuracy of VGGNet. The chapter presented simulations using various MRE and SD values that mimic the impact of several proposed approximate multipliers in the literature. The simulation covered both Uniform and Gaussian PDFs and assessed their impact on the network's accuracy. The simulation results showed that approximate multipliers have very little impact on the network's accuracy while having significant performance gains in terms of power, area, and delay. Additionally, a hybrid approach was proposed which uses a mix of exact and approximate multipliers. By applying the hybrid approach, the approximate multiplication

can be implemented only in the deeper layers of the network which was determined to have the least impact on the accuracy. The hybrid approach simulation leads to a reduced negligible impact on the accuracy while having significant savings in power, area, and delay on a large portion of the network.

The work of Chapter 2 was expanded in Chapter 3 by simulating deep learning training using approximate multipliers. The chapter presented a new hybrid training method that uses a combination consisting of exact multipliers and approximate multipliers. Based on the proposed method, the training can start using approximate multipliers, then it switches to exact multipliers for the last few epochs. The utilization of the approximate multipliers in the initial portion of the training can achieve significant performance gains in terms of speed, power, and area for a large portion of the training stage. On the other hand, any resulting accuracy loss is compensated for by using the exact multipliers for the last epochs of training.

Chapter 4 proposed the new concept of CNN inference using a preprocessing precision controller and approximate multipliers with various precisions. The proposed concept allows for the utilization of approximate multipliers with various precision simultaneously by pre-determining the required precision for the input image. The proposed controller is a tiny two-class CNN that can be utilized in large clusters that contain multiple approximate multiplier-based CNN inference accelerators with different precisions, or in a single CNN inference accelerator built with precision reconfigurable approximate multipliers. The controller aims to maximize the performance in terms of power and speed by using low precision approximate multiplier whenever it is predicted that this does not cause an accuracy loss. This augments the performance of CNN inference by allowing for the utilization of existing low precision and high precision approximate multiplier hybridly. The chapter also proposed a new design for a reconfigurable approximate multiplier to allow for the utilization of the proposed concept in single-core designs.

Chapter 5 proposed several practical considerations for accuracy evaluation in sensor-based machine learning and deep learning. This includes studying the impact of thermal noise, determining the adequate quantization level, and evaluating the accuracy tolerance

to sensor failure. In terms of thermal noise, it was demonstrated that models' accuracy tolerance to thermal noise can vary significantly from model to model. This should impact the designer's selection in terms of the appropriate machine learning model to deploy for production. Another aspect that was studied is the impact of different ADC quantization levels on accuracy. Finding the adequate quantization level for inference is crucial for low-power embedded AI applications. Lowering the quantization level will lead to lowering the required ADC resolution which will lower the power. Additionally, the chapter simulated the ADC ENOB and its impact on the model's accuracy. This was performed by simulating lower than ideal quantization levels in each model. The chapter also studied the impact of sensor failure on accuracy. This test allows for selecting models which are more resilient to such failures.

Chapter 6 proposed the new concept of person identification through physical activity. This contribution opens the door for sharing wearable technologies where the user profile can be detected automatically from their physical activity. The chapter also included an architecture for a multi-label shared DNN for simultaneous classification for the physical activity and the activity performer.

7.2 Future Work

Chapters 2-4 focused on presenting solutions to enable the utilization of approximate multipliers in CNN designs to achieve high-performance gains in terms of power, area, and speed while keeping the accuracy loss minimal. The usage of approximate multipliers can be explored in other types of DNN networks, this includes:

- Exploring the usage of approximate multipliers in Recurrent Neural Networks (RNNs), particularly for Long Short-Term Memory (LSTM) hardware designs. This can help in achieving significant performance enhancement in time-series applications such as video/audio.

- Exploring the usage of approximate multipliers for generative models with a focus on Generative adversarial networks (GANs). GANs can create new examples that resemble the training data. It will be interesting to study the impact of approximate multipliers on GANs.

Moreover, studying the applicability of other approximate computing applications such as approximate adders and approximate memory in conjunction with the usage of approximate multipliers is another area that is worth exploring. This will allow for the implementation of approximate MAC units which are designed end to end based on approximate computing concepts allowing for further performance enhancements.

Chapter 6 presented the concept of using machine learning for person identification through physical activity. The study used one dataset to present the concept and propose a shared DNN for simultaneous classification of the performed activity and the activity performer. To solidify the proposed concept of using machine learning for person identification through physical activity it is worth expanding the study using multiple different datasets. Additionally, other forms of classifications can be explored such as gender and age classification through physical activity.

References

[1] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.

[2] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

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

[4] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[5] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[6] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[7] Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit." 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2017.

[8] Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." IEEE Journal of Solid-State Circuits 52.1 (2016): 127-138.

[9] Cavigelli, Lukas, et al. "Origami: A convolutional network accelerator." Proceedings of the 25th edition on Great Lakes Symposium on VLSI. 2015.

[10] Yin, Shouyi, et al. "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications." IEEE Journal of Solid-State Circuits 53.4 (2017): 968-982.

[11] Yuan, Zhe, et al. "STICKER: An Energy-Efficient Multi-Sparsity Compatible Accelerator for Convolutional Neural Networks in 65-nm CMOS." IEEE Journal of Solid-State Circuits 55.2 (2019): 465-477.

[12] Liu, Weiqiang, Fabrizio Lombardi, and Michael Shulte. "A Retrospective and Prospective View of Approximate Computing [Point of View." Proceedings of the IEEE 108.3 (2020): 394-399.

[13] Han, Jie, and Michael Orshansky. "Approximate computing: An emerging paradigm for energy-efficient design." 2013 18th IEEE European Test Symposium (ETS). IEEE, 2013.

[14] Xu, Qiang, Todd Mytkowicz, and Nam Sung Kim. "Approximate computing: A survey." IEEE Design & Test 33.1 (2015): 8-22.

[15] Chippa, Vinay K., et al. "Analysis and characterization of inherent application resilience for approximate computing." Proceedings of the 50th Annual Design Automation Conference. 2013.

[16] Mittal, Sparsh. "A survey of techniques for approximate computing." ACM Computing Surveys (CSUR) 48.4 (2016): 1-33.

[17] **Hammad, Issam**, and Kamal El-Sankary. "Impact of Approximate Multipliers on VGG Deep Learning Network." IEEE Access 6 (2018): 60438-60444.

[18] **Hammad, Issam,** et al. "CNN Inference Using a Preprocessing Precision Controller and Approximate Multipliers With Various Precisions." IEEE Access 9 (2021): 7220-7232.

[19] **Hammad, Issam**, and Kamal El-Sankary. "Practical considerations for accuracy evaluation in sensor-based machine learning and deep learning." Sensors 19.16 (2019): 3491.

[20] **Hammad, Issam,** Kamal El-Sankary, and Jason Gu. "Deep learning training with simulated approximate multipliers." 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2019.

[21] **Hammad, Issam**, and Kamal El-Sankary. "Using Machine Learning for Person Identification through Physical Activities." 2020 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2020.

[22] **Hammad, Issam**, et al. "Using Deep Learning to Automate the Detection of Flaws in Nuclear Fuel Channel UT Scans." arXiv preprint arXiv:2102.13635 (2021).

[23] **Hammad, Issam**, Kamal El-Sankary, and Holly Hornibrook. "RETSManager: Real-estate database builder and synchronizer." SoftwareX 10 (2019): 100351.

[24] **Hammad, Issam,** Kamal El-Sankary, and Jason Gu. "A Comparative Study on Machine Learning Algorithms for the Control of a Wall Following Robot." 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2019.

[25] Sze, Vivienne, et al. "Efficient processing of deep neural networks: A tutorial and survey." Proceedings of the IEEE 105.12 (2017): 2295-2329.

[26] Du, Li, et al. "A reconfigurable streaming deep convolutional neural network accelerator for Internet of Things." IEEE Transactions on Circuits and Systems I: Regular Papers 65.1 (2018): 198-208.

[27] Venkatachalam, Suganthi, and Seok-Bum Ko. "Design of power and area efficient approximate multipliers." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25.5 (2017): 1782-1786.

[28] Leon, Vasileios, et al. "Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 26.3 (2018): 421-430.

[29] Zendegani, Reza, et al. "RoBA multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2 (2017): 393-401.

[30] Hashemi, Soheil, R. Bahar, and Sherief Reda. "DRUM: A dynamic range unbiased multiplier for approximate applications." Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. IEEE Press, 2015.

[31] Zervakis, Georgios, et al. "Design-efficient approximate multiplication circuits through partial product perforation." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24.10 (2016): 3105-3117.

[32] Yang, Tongxin, Tomoaki Ukezono, and Toshinori Sato. "Low-Power and High-Speed Approximate Multiplier Design with a Tree Compressor." Computer Design (ICCD), 2017 IEEE International Conference on. IEEE, 2017.

[33] Mrazek, Vojtech, et al. "Design of power-efficient approximate multipliers for approximate artificial neural networks." Proceedings of the 35th International Conference on Computer-Aided Design. ACM, 2016.

[34] Yonatan Geifman, VGG16 models for CIFAR-10 and CIFAR-100 using Keras, GitHub rep., https://github.com/geifmany/cifar-vgg

[35] F. Chollet, 2015 Keras, https://github.com/fchollet/keras

[36] Krizhevsky, Alex, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Vol. 1. No. 4. Technical report, University of Toronto, 2009 [online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[37] Liu, Shuying, and Weihong Deng. "Very deep convolutional neural network based image classification using small training sample size." Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on. IEEE, 2015.

[38] Cong, Jason, and Bingjun Xiao. "Minimizing computation in convolutional neural networks." International conference on artificial neural networks. Springer, Cham, 2014

[39] Narayanamoorthy, Srinivasan, et al. "Energy-efficient approximate multiplication for digital signal processing and classification applications." IEEE transactions on very large scale integration (VLSI) systems 23.6 (2014): 1180-1184.

[40] Vahdat, Shaghayegh, et al. "TOSAM: An energy-efficient truncation-and rounding-based scalable approximate multiplier." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27.5 (2019): 1161-1173.

[41] Liu, Weiqiang, et al. "Design of approximate radix-4 booth multipliers for error-tolerant computing." IEEE Transactions on Computers 66.8 (2017): 1435-1441.

[42] Lin, Darryl, Sachin Talathi, and Sreekanth Annapureddy. "Fixed point quantization of deep convolutional networks." International conference on machine learning. 2016.

[43] Gupta, Suyog, et al. "Deep learning with limited numerical precision." International Conference on Machine Learning. 2015.

[44] Yin, Shihui, and Jae-Sun Seo. "A 2.6 TOPS/W 16-Bit Fixed-Point Convolutional Neural Network Learning Processor in 65-nm CMOS." IEEE Solid-State Circuits Letters 3 (2019): 13-16.

[45] Wei, Xuechao, et al. "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs." Proceedings of the 54th Annual Design Automation Conference 2017.

[46] Li, Huimin, et al. "A high performance FPGA-based accelerator for large-scale convolutional neural networks." 2016 26th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2016.

[47] Liang, Feng, et al. "Design of 16-bit fixed-point CNN coprocessor based on FPGA." 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP). IEEE, 2018.

[48] Liu, Zhenhong, et al. "Simul: An algorithm-driven approximate multiplier design for machine learning." IEEE Micro 38.4 (2018): 50-59.

[49] Ansari, Mohammad Saeed, et al. "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 28.2 (2019): 317-328.

[50] Moons B., Bankman D., Verhelst M. (2019) Circuit Techniques for Approximate Computing. In: Embedded Deep Learning. Springer, Cham.

[51] Tasoulas, Zois-Gerasimos, et al. "Weight-Oriented Approximation for Energy-Efficient Neural Network Inference Accelerators." IEEE Transactions on Circuits and Systems I: Regular Papers (2020).

[52] Recht, Benjamin, et al. "Do imagenet classifiers generalize to imagenet?." arXiv preprint arXiv:1902.10811 (2019).

[53] Leon, Vasileios, et al. "Cooperative arithmetic-aware approximation techniques for energy-efficient multipliers." Proceedings of the 56th Annual Design Automation Conference 2019.

[54] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

[55] Agarap, Abien Fred. "Deep learning using rectified linear units (relu)." arXiv preprint arXiv:1803.08375 (2018).

[56] NanGate FreePDK15 Generic Open Cell Library. (2020). Silicon Integration Initiative, Inc (Si2). [Online]. Available: https://si2.org/open-cell-library/

[57] Lian, Xiaocong, et al. "High-performance fpga-based cnn accelerator with block-floating-point arithmetic." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27.8 (2019): 1874-1885.

[58] Kachris, Christoforos, Babak Falsafi, and Dimitrios Soudris, eds. Hardware Accelerators in Data Centers. Vol. 1. No. 1. Springer, 2019.

[59] Mao, Huizi, et al. "Towards real-time object detection on embedded systems." IEEE Transactions on Emerging Topics in Computing 6.3 (2016): 417-431.

[60] Sun, Baohua, et al. "Ultra power-efficient cnn domain specific accelerator with 9.3 tops/watt for mobile and embedded applications." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018.

[61] Emmert-Streib, F.; Matthias, D. Evaluation of Regression Models: Model Assessment, Model Selection and Generalization Error. Mach. Learn. Knowl. Extr. 2019, 1.1, 521–551.

[62] Emmert-Streib, F.; Salisou, M.; Matthias, D. A comprehensive survey of error measures for evaluating binary decision making in data science. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 2019, e1303, doi:10.1002/widm.1303.

[63] Karystinos, G.N.; Dimitrios, P.A. On overfitting, generalization, and randomly expanded training sets. IEEE Trans. Neural Netw. 2000, 11.5, 1050–1057.

[64] Altun, K.; Billur, B.; Orkun, T. Comparative study on classifying human activities with miniature inertial and magnetic sensors. Pattern Recognit. 2010, 43, 3605–3620.

[65] Barshan, B.; Murat, C.Y. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. Comput. J. 2014, 57, 1649–1667.

[66] Chung, S. Sensor Data Acquisition and Multimodal Sensor Fusion for Human Activity Recognition Using Deep Learning. Sensors 2019, 19.7, 1716.

[67] Miao, F. A Wearable Sensor for Arterial Stiffness Monitoring Based on Machine Learning Algorithms. IEEE Sens. J. 2018, 19.4, 1426–1434.

[68] Yeh, C. Machine Learning for Long Cycle Maintenance Prediction of Wind Turbine. Sensors 2019, 19.7, 1671.

[69] University of Califonia Irvine Machine Learning Repository, Daily and Sports Activities Data Set. 2013. Available online: https://archive.ics.uci.edu/ml/datasets/daily+and+sports+activities (accessed on 1 March 2019).

[70] Villeneuve, E. Signal quality and compactness of a dual-accelerometer system for gyro-free human motion analysis. IEEE Sens. J. 2016, 16, 6261–6269.

[71] Madgwick, S.O.H.; Harrison, A.J.L.; Sharkey, P.M.; Vaidyanathan, R.; Harwin, W.S. Measuring motion with kinematically redundant accelerometer arrays: Theory, simulation and implementation. Mechatronics 2013, 23, 518–529.

[72] Uyttenhove, K.; Michel, S.; Steyaert, J. Speed-power-accuracy tradeoff in high-speed CMOS ADCs. IEEE Trans. Circuits Syst. Analog Digit. Signal Proc. 2002, 49.4, 280–287.

[73] Belcher, R.A. ADC standard IEC 60748-4-3: Precision measurement of alternative ENOB without a sine wave. IEEE Trans. Instrum. Meas. 2015, 64.12, 3183–3200.

[74] Xsens Technologies MTi, B.V. MTx User Manual, Document MT0100P, Revision N, 27 May 2009. [online], http://www.xsens.com (accessed on 2 April 2019).

[75] Attal, F. Physical human activity recognition using wearable sensors. Sensors 2015, 15.12, 31314–31338.

[76] Wang, Z. An incremental learning method based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition by using wearable sensors. IEEE Trans. Inf. Technol. Biomed. 2012, 16.4, 691–699.

[77] Trabelsi, D. An unsupervised approach for automatic activity recognition based on hidden Markov model regression. IEEE Trans. Autom. Sci. Eng. 2013, 10.3, 829–835.

[78] Pedregosa, F. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 2011, 12, 2825–2830.

[79] Jeong, B.G. Simplified noise model parameter estimation for signal-dependent noise. Signal Process. 2014, 96, 266–273.

[80] Robert, L.P. Mechanical-thermal noise in MEMS gyroscopes. IEEE Sens. J. 2005, 5.3, 493–500.

[81] Lent, B. Practical considerations of accelerometers noise. Available online: https://www. endevco. com/news/arc hivednews/2009/2009_12/TP324. pdf> (accessed on 1 May 2019).

[82] Abolghasem, N. A generalized study of coil-core-aspect ratio optimization for noise reduction and SNR enhancement in search coil magnetometers at low frequencies. IEEE Sens. J. 2015, 15.11, 6454–6459.

[83] How to Increase the Analog-to-Digital Converter Accuracy in an Application; Freescale Semiconductor, Inc.: Austin, TX, USA, 2016.

[84] Large, D.; James, F. The HFC Plant. In Broadband Cable Access Networks; Elsevier Inc.: Amsterdam, The Netherlands, 2009.

[85] Alink, M.; Oude, S. Spurious-free dynamic range of a uniform quantizer. IEEE Trans. Circuits Syst. Express Briefs 2009, 56.6, 434–438

[86] Johns, D.A.; Ken, M. Analog Integrated Circuit Design; John Wiley & Sons: Hoboken, NJ, USA, 2008.

[87] Murmann, B. ADC Performance Survey 1997–2019. Available online: http://web.stanford.edu/~murmann/adcsurvey.html (accessed on 8 June 2019).

[88] Timothy, H.W. The hitchhiker's guide to successful residential sensing deployments. In Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, Seattle, WA, USA, 1–4 November 2011.

[89] Wang, Lukun. "Recognition of human activities using continuous autoencoders with wearable sensors." Sensors 16.2 (2016): 189.

[90] Lu, Feng, et al. "A multi-classifier combination method using sffs algorithm for recognition of 19 human activities." International Conference on Computational Science and Its Applications. Springer, Cham, 2016.

[91] Wang, LuKun, and RuYue Liu. "Human Activity Recognition Based on Wearable Sensor Using Hierarchical Deep LSTM Networks." Circuits, Systems, and Signal Processing (2019): 1-20.

[92] Ramasamy Ramamurthy, Sreenivasan, and Nirmalya Roy. "Recent trends in machine learning for human activity recognition—A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.4 (2018): e1254.

[93] Hu, Chunyu, et al. "A novel random forests based class incremental learning method for activity recognition." Pattern Recognition 78 (2018): 277-290.

Appendix A: IEEE Copyright Permission

The IEEE does not require individuals working on a thesis to obtain a formal reuse license. This applied to the following published articles which were reused in the thesis

- **Hammad, Issam**, and Kamal El-Sankary. "Impact of Approximate Multipliers on VGG Deep Learning Network." IEEE Access 6 (2018): 60438-60444.

- **Hammad, Issam,** et al. "CNN Inference Using a Preprocessing Precision Controller and Approximate Multipliers with Various Precisions." IEEE Access 9 (2021): 7220-7232.

- **Hammad, Issam,** Kamal El-Sankary, and Jason Gu. "Deep learning training with simulated approximate multipliers." 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2019.

- **Hammad, Issam**, and Kamal El-Sankary. "Using Machine Learning for Person Identification through Physical Activities." 2020 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2020.

The following is a printout obtained from the IEEE copyright Clearence Center (RightsLink ®) as a proof:

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Additionally, the following correspondence with IEEE confirms the permission for reuse.

From: Issam Hammad (Issam.hammad@gmail.com) on April 23rd 2021, 1:26pm

"Hi,

I am seeking permission to reuse the following published articles in my Ph.D. thesis which will be submitted to the Faculty of Graduate Studies at Dalhousie University, Halifax, NS, Canada.

1) Hammad, Issam, and Kamal El-Sankary. "Impact of approximate multipliers on VGG deep learning network." IEEE Access 6 (2018): 60438-60444.

2) Hammad, Issam, Kamal El-Sankary, and Jason Gu. "Deep learning training with simulated approximate multipliers." 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2019.

3) Hammad, Issam, et al. "CNN Inference Using a Preprocessing Precision Controller and Approximate Multipliers With Various Precisions." IEEE Access 9 (2021): 7220-7232.

4) Hammad, Issam, and Kamal El-Sankary. "Using Machine Learning for Person Identification through Physical Activities." 2020 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2020.

Thanks,

Response from M.E. Brennan (me.brennan@ieee.org) on April 26th at 12:58pm:

"Dear Issam Hammad,

The IEEE does not require individuals working on a dissertation/thesis to obtain a formal reuse license however, you must follow the requirements listed below:

Textual Material

Using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © [Year of publication] IEEE.

In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appears prominently with each reprinted figure and/or table.

If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Full-Text Article

If you are using the entire IEEE copyright owned article, the following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line. You may not use the final published version

In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to

http://www.ieee.org/publications_standards/publications/rights/rights_link.html

to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Kind regards,

M.E. Brennan

Ms M.E. Brennan
IEEE
501 Hoes Lane
Piscataway, NJ 08854-4141 USA
me.brennan@ieee.org

+1 (732) 562-2660”

Appendix B: MDPI Copyright Permission

All articles published by MDPI are made immediately available worldwide under an open access license. This means:

- everyone has free and unlimited access to the full-text of all articles published in MDPI journals;
- everyone is free to re-use the published material if proper accreditation/citation of the original publication is given;
- open access publication is supported by the authors' institutes or research funding agencies by payment of a comparatively low Article Processing Charge (APC) for accepted articles.

Permissions:

> No special permission is required to reuse all or part of article published by MDPI, including figures and tables. For articles published under an open access Creative Common CC BY license, any part of the article may be reused without permission provided that the original article is clearly cited. Reuse of an article does not imply

The statement above applies to the following reused MDPI article:

- **Hammad, Issam**, and Kamal El-Sankary. "Practical considerations for accuracy evaluation in sensor-based machine learning and deep learning." Sensors 19.16 (2019): 3491.

The printout below can confirm the permission for reuse (source: https://www.mdpi.com/openaccess)

MDPI Open Access Information and Policy

All articles published by MDPI are made immediately available worldwide under an open access license. This means:

- everyone has free and unlimited access to the full-text of *all* articles published in MDPI journals;
- everyone is free to re-use the published material if proper accreditation/citation of the original publication is given;
- open access publication is supported by the authors' institutes or research funding agencies by payment of a comparatively low **Article Processing Charge (APC)** for accepted articles.

Permissions

No special permission is required to reuse all or part of article published by MDPI, including figures and tables. For articles published under an open access Creative Common CC BY license, any part of the article may be reused without permission provided that the original article is clearly cited. Reuse of an article does not imply endorsement by the authors or MDPI.