Photonic Neuron for Artifical Neural Networks

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Student Name
(03010104)

under the guidance of

Your guide name

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY PATNA
PATNA - 800013, BIHAR

<u>CERTIFICATE</u>

*This is to certify that the work contained in this thesis entitled "**Photonic Neuron for Artifical Neural Networks**" is a bonafide work of **Student Name (Roll No. 03010104)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Patna under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Your guide name**

Assistant/Associate Professor,

May, 2023 — Department of Computer Science & Engineering,

Patna. — Indian Institute of Technology Patna, Bihar.

Acknowledgements

Write acknowledgements, if your want to.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ever since machine learning has been introduced into the field of computer science, it has been spearheading breakthroughs in a number of fields. It has taken the world by storm and now, many fields will simply cease to function without these techniques. Deep learning is one sub-field of machine learning which is deeply rooted in todays society. Deep learning is now part of common man's everyday life and it will remain so for the forseeable future.

As data collection and storage becomes more prominent, machine and deep learning has continued to dominate the data space, provide insights into complex data which is simply not possible with other mathematical methods. Deep learning is one of the most rapidly expanding machine learning technologies, relying on multi-layered artificial neural networks (ANNs) implemented in digital electronics to handle big data sets, integrating and analysing massive volumes of information quickly without the need for explicit instructions. These ANNs have been modified and augmented in many ways, leading many domain specific techniques, most notable one being Convolutional Neural Networks (CNNs) which is primarily used in image analysis.

1.1 Artifical Neural Networks

Artificial neural networks refers to those algorithms which are inspired by the biological neural networks that constitute animal brains. Such systems learn to perform tasks by considering examples, generally without being programmed with any task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They have found most use in applications difficult to express with a traditional computer algorithm using rule-based programming.

Traditionally ANNs have been described as a black box of sorts. It has a number of input variables and output variables using simple arithmetic connections, gives output from the input.

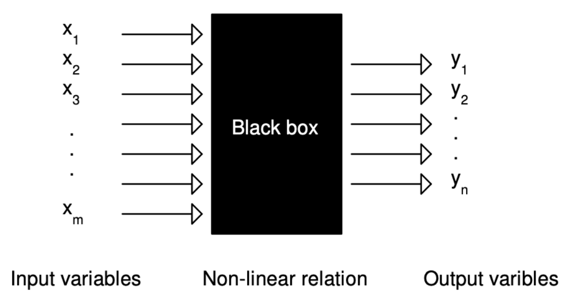

Fig. 1.1: Blackbox representation of a ANN

ANNs Traditionally tend to excel where the relations between inputs and outputs are non-linear. It is practically better and more efficient at classifying or identifying non-linear relationships rather that linear ones, where it might perform worse than a more statistical approach.

Nowadays, the most common type of ANN is the feedforward neural network, which consists of a group of neurons (called a layer) that transfer data to another group of neurons in a feedforward manner. The first layer is called the input layer, and the last layer is called the output layer. The layers in between are called hidden layers. The input data travels through the layers in a feedforward manner, and the output is the result of the last layer.

The output is then compared to the expected output, and the error is calculated. The error is then backpropagated through the network, and the weights are adjusted accordingly. This process is repeated until the error is below a certain threshold.

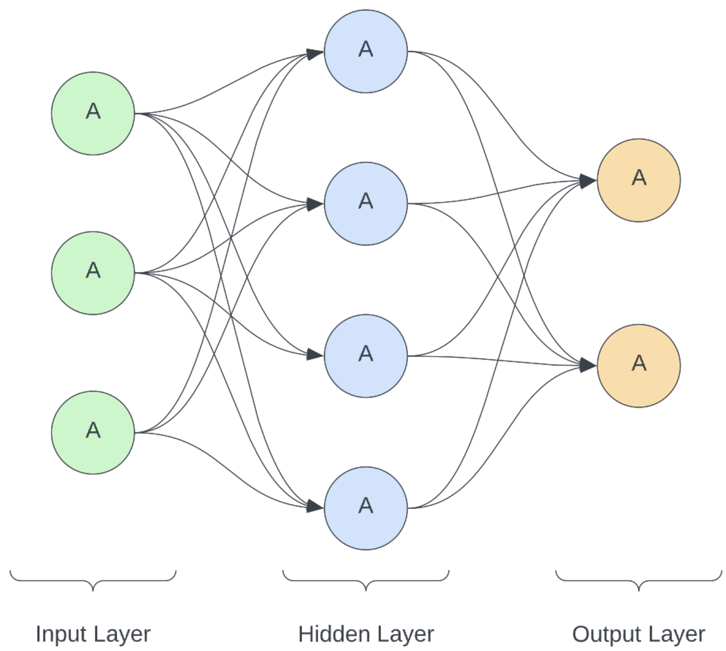

Fig. 1.2: Feedforward ANN

1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs usually apply convolution operation over the image (hence the name) by using a filter, called the kernel. This produces a smaller image but where features can be extracted easily. Diiferent filters are applied to the same image in the form of multiple channels resulting in diverging features in each channel which can be detected. This is called feature mapping. The feature maps are then flattened and fed into a fully connected ANN which gives the final output. The application of kernel is illustrated in figure 1.3.

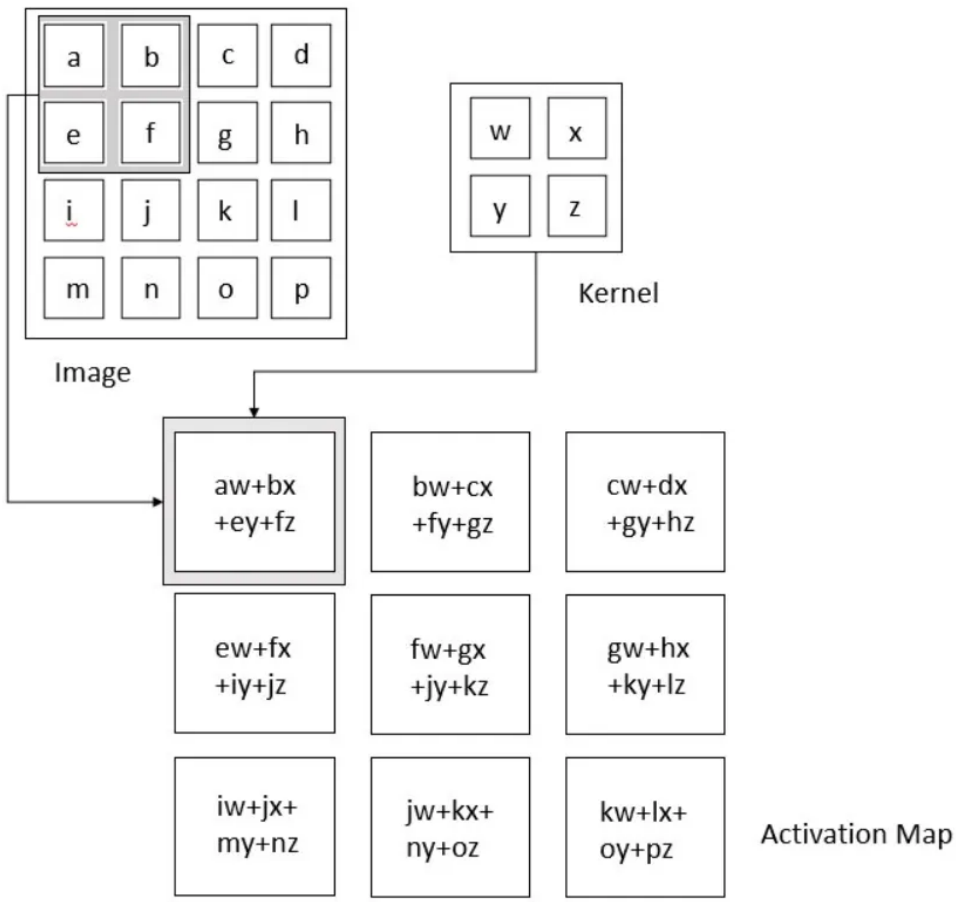

Fig. 1.3: Working of kernel and convolution in CNNs [1]

1.3 Conventional Implementation of Neural Networks

Neural networks, for the most part, have been software based. The algorithms are implemented in code and is executed in high-power GPUs for fast parallel computations. This form is extremely flexible and can be used for a variety of applications. The model can be changed as and when the need arises and is reletively commitment-less. The hardware itself is also largely independent of the software and changed/replaced whenever deemed necessary.

There has been a recent growth in certain of Application Specific Integrated Circuits (ASICs) specifically for neural networks, some notable examples being from popular manufactors like NVIDIA [4] and Apple. These chips are designed to perform neural network computations and are extremely fast and efficient. They are also extremely expensive and are not easily available.

Even with the advent of ASICs in Machine learning space, the field still suffers from the fact that the hardware is not really coupled with the software. By increasing the specificity of the hardware, the efficiency of the system can be increased by a large margin.

1.4 Disadvantages of Conventional Implementation

The conventional implementation of neural networks has a number of disadvantages. The most prominent one being the fact that the hardware is not really coupled with the software. The hardware is not really designed to perform neural network computations and is not really efficient at it. Large amounts of power are required to train models for prolonged amounts of time. This leads to a lot of inefficiencies in the system.

1.5 Photonic Neuron

As discussed, the conventional implementation of neural networks has a number of disadvantages. The hardware is not really designed to perform neural network computations and is not really efficient at it. This sparks the question "What can be a proper hardware implementation of neuron, the basic structure of a neural network, that can be used to perform neural network computations efficiently?". The answer to this question that this thesis proposes is the Photonic Neuron.

The photonic neuron exploits the fact that multiplication can be done essentially for free in the photonics domain through the use fo hardware like Mach-Zehnder Interferometers or Micro-Ring Resonators. Photonics is known to be inherently very fast since most of the operations are done in speeds close to the speed of light. This makes it a very good candidate for the speed up of existing hardware implementation.

Many prospective applications become blaringly obvious when we consider the speeds that photonics really has to offer. When we also consider the power efficiency of such a implementation, we can see the possibilities. Easily identifiable applications include on-device deployments of trained models for applications like self-driving cars, drones, etc. These require highly power efficient and latentcy-less neural network implementation which photonics can offer.

1.6 Diiferent Architectures of Photonic Neurons

There are a number of different architectures of photonic neurons that have been proposed. Some of them are discussed below.

One possible implementation is cascading a number of Mach-Zehnder Interferometers (MZIs) to perform the multiplication operation [5]. MZIs can be efficiently used to achieve multiplication. Thus a series of MZIs can be cascaded as per the requirement of the model the achieve an efficient implementation of the neural network.

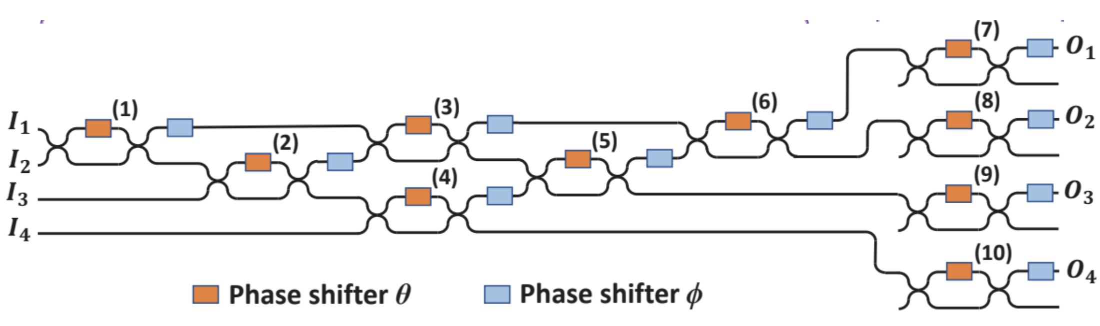

Fig. 1.4: Schematic diagram of a 4x4 MZI based photonic neuron

As is evident from the picture, the hardware complexity grows exponentially as the number of parameters increase, which is not really practical as the number of parameters in a neural network is usually very large, even reaching billions at times [6]. This makes the implementation of such a system very difficult.

Another approach to making neural networks in photonics domain is using MicroRing Resonators (MRRs). This uses multiple MRRs in a cascaded fashion to implement a complete spiking neural network [2]. Although cascading MRR is somewhat better than cascading MZIs, it still suffers from the same problem of exponential growth of hardware complexity.

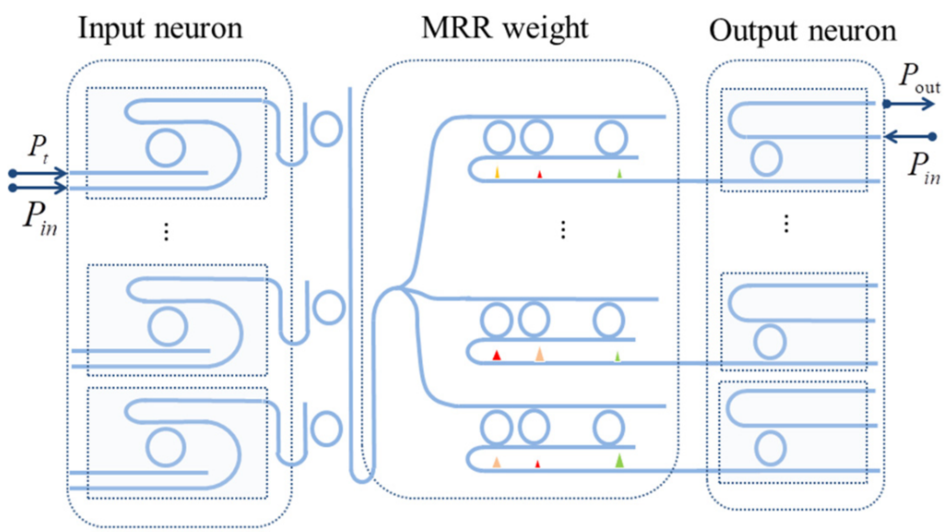

Fig. 1.5: Schematic diagram of a all MRR photonic neuron [2]

One way to overcome this issue is to use one resuable component for the repetitive operations of the neural network. Just like how GPU were employed for neural net calculations because they were efficient at parallel and repeptetive operations, we can use a small reusable neuron and reuse for the entire neural network.

Following in this path, the structure known as PEMAN was introduced [3]. PEMAN stands for Photonic Electronic Multipication Accumulation Neuron. It proposes a hybrid photonic electronic neuron that can be reused many times to implement a complete neural network. The structure of PEMAN is shown in figure 1.6.

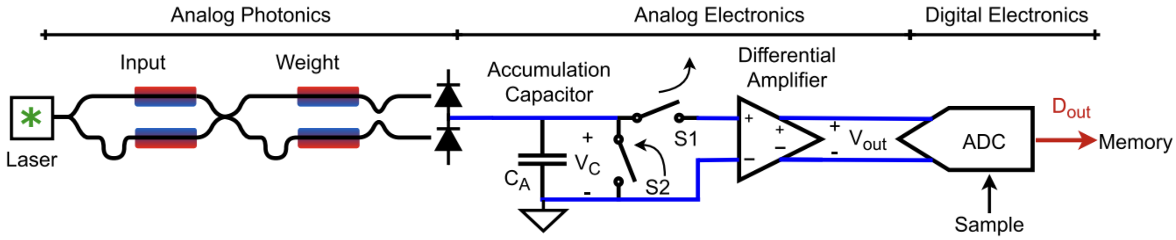

Fig. 1.6: Schematic diagram of a PEMAN [3]

1.7 Motivation

The motivation behind this thesis is to explore the possibility of using a PEMAN like structure to implement a complete neural network. The PEMAN structure is a very good candidate for the implementation of a neural network in the photonic domain. It is highly reusable and can be used to implement a complete neural network. The PEMAN structure is also highly power efficient and can be used to implement a neural network that is highly power efficient.

In particular, this thesis aims to explore the creation of a artifical neural network and convolutional neural network architecture using the PEMAN structure. It also tries to explore the process of training on the said hardware implementation and study the accuracy, ENOB and time efficiency of the system.

Chapter 2

The PEMAN Structure

The PEMAN structure is a hybrid photonic electronic structure which uses photonic operation for multiplication and electronic hardware for accumulation. It also has inbuilt non-linearity with the use of an ADC.

2.1 Working Principle

The PEMAN structure consists of three distinct sections: the analog photonics section, analog electronics section and digital electronic ssection. Each section applies one part of the ANN algorithm and is thus resuable inside a neural network. Each of the section is discussed in detail in the upcoming sections.

2.1.1 Analog Photonics Section

The analog photonics section is responsible for the multiplication of the input vector with the weight matrix. The multiplication of inputs and weights have been identified as the most computationally intensive part of the ANN algorithm and thus we can exploit photonics to the maximum extent by using it in this context.

In order to study how the multiplication works, a single MZI is first studied. Figure 2.1 shows the structure of an MZI.

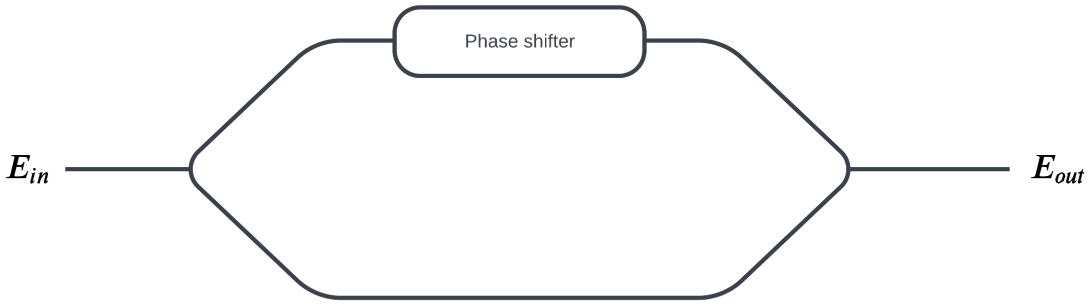

Fig. 2.1: Schematic diagram of a 1x1 MZI

The structure is made up oftwo Y junctions on both sides with a phase shifter in-between. Lets say the input field is E_{in}, output field is E_{out} and the phase shifter has a shift of is ϕ. After applying the transfer matrices for each of the components, we get

$$E_{out} = E_{in} \cos \phi \tag{2.1}$$

This result can be interpreted as the multiplication of the input field with a value of $\cos \phi$. In PEMAN, the laser is generally given a constant current as it is reused for many operations. This means that the phase shifter needs to be changed dynamically to achieve multiplication.

In the actual PEMAN structure, two MZIs are used, one for input setting and one for weight setting, thereby giving the effect of multiplication of input and weight together.

Figure 2.2 depicts the structure that is used in this thesis as the PEMAN structure.

2.1.2 Analog Electronics Section

The analog electronics domain consists of a capacitor and a differential amplifier. These two components together is used as an accumulator to achieve the addition part in a neuron. Once the capacitor stores all the output produced by the photonics part, it is discharged into the next stage.

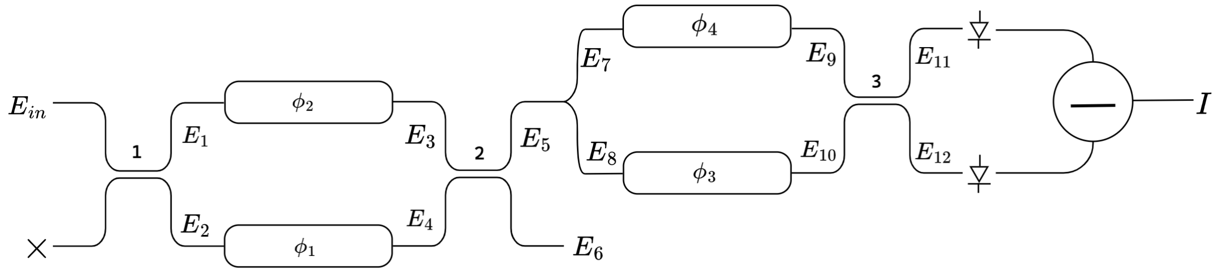

Fig. 2.2: Schematic diagram of the PEMAN structure

2.1.3 Digital Electronics Section

The digital electronics section consists of an ADC. This ADC is responsible to convert the analog current coming from the capacitor to digital domain. WHile converting, it is also implemented to apply a non- linear function to account for the activation function in a neuron. The ADC is also responsible for quantizing the output to a certain number of bits. Quantizing the values reduces the accuracy but boosts the overall speed.

2.2 Timing Diagram

Figure 2.3 shows the timing diagram of the PEMAN structure. The timing diagram shows the working of the PEMAN structure for a single neuron. It is necessary to study the time complexities and the under-workkings of the PEMAN structure in order to properly utilize its advantages.

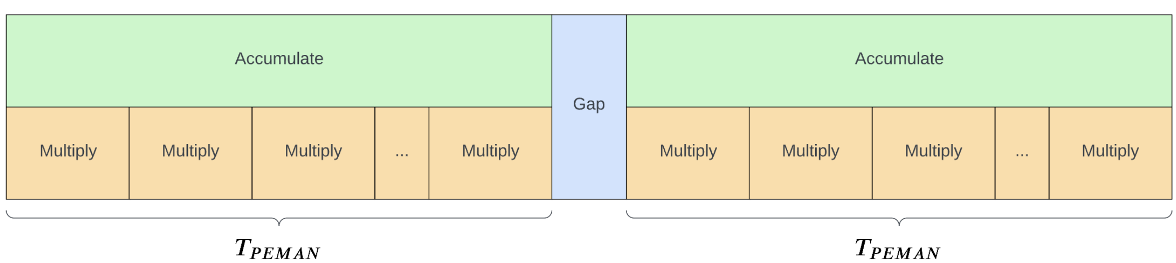

Fig. 2.3: Timing diagram of the PEMAN structure

From the figure, we can see that accumulation and the consecutive functions do not need to be as fast as multiplication. This is because multiplication is done a number of times before it has to be accumulated. This can also be leveraged by the fact that multiplication is done in the photonics domain and accumulation is done in the analog domain. This means that the multiplication can be done relatively fast while the accumulation can be done at a much slower speed.

2.3 PEMAN Structure and Neural Networks

The PEMAN structure is made to be reused such that an entire neural network can be implemented using the same structure. While it is implemented for the entire structure, we also need to consider training and inference.

2.3.1 Conventional Method

The conventional method to use a neural network built with the PEMAN structure involves using a GPU to train the model as normally done. Once the model has been trained to a satisfactory state, the weights are extracted. These weights are then matched with a premade lookup table to extract the necessary phase shifts for the weights. Now, when actual input is sent for inference, the inputs are converted to phases using the same lookup table, and set. The corresponding weight phases are set simultaneously. The output is then passed through the remainder of the structure and the final output is obtained.

This method is the traditional way to use PEMAN as we do not have to deal with reduced bit operation during training and can use effective quantization techniques to quantized the extract the weights without much drop in accuracy.

2.3.2 Training on PEMAN

Training on PEMAN is a relatively new concept. The idea is to train the model directly on the PEMAN structure. This means that the weights are not extracted from a GPU but are instead trained on the PEMAN structure itself. This is done by essentially using the same feedforward and backpropogation methods which are slightly modified to adapt to the new structure.

This approach can provide a significant speed boost to both training and inference as the weights do not need to be extracted and the entire model can be trained on the PEMAN structure itself. This also means that the model can be trained with reduced bit operation and thus can be trained faster. However, this approach is still in its infancy and needs to be studied in detail to understand its advantages and disadvantages.

One disadvantage might be that the reduced bit operation might degrade the accuracy more that necessary. But, upon testing for the handwritten digit recognition problem, it was found that the accuracy drop was not significant. This means that the reduced bit operation can be used to train the model faster without much drop in accuracy.

2.4 Algorithm for training on PEMAN

The process of training on PEMAN directly is fairly straightforward and bears a lot of resemblance to the original algorithm. The only real change is most of the operations now have an additional lookup step, which is necessary to convert the digital data to values of physical significance.

Figure 2.4 shows the algorithm for training on PEMAN. The algorithm is divided into two parts: the feedforward part and the backpropogation part. The feedforward part is responsible for calculating the output of the network for a given input. The backpropogation part is responsible for calculating the gradients of the weights and biases and updating them.

Through using this technique, we employ the same backpropogation technique that has been proven to be successful while also modiying it so that we can train the phases of the PEMAN directly. This means that we can train the model directly on the PEMAN structure and thus can achieve a significant speed boost.

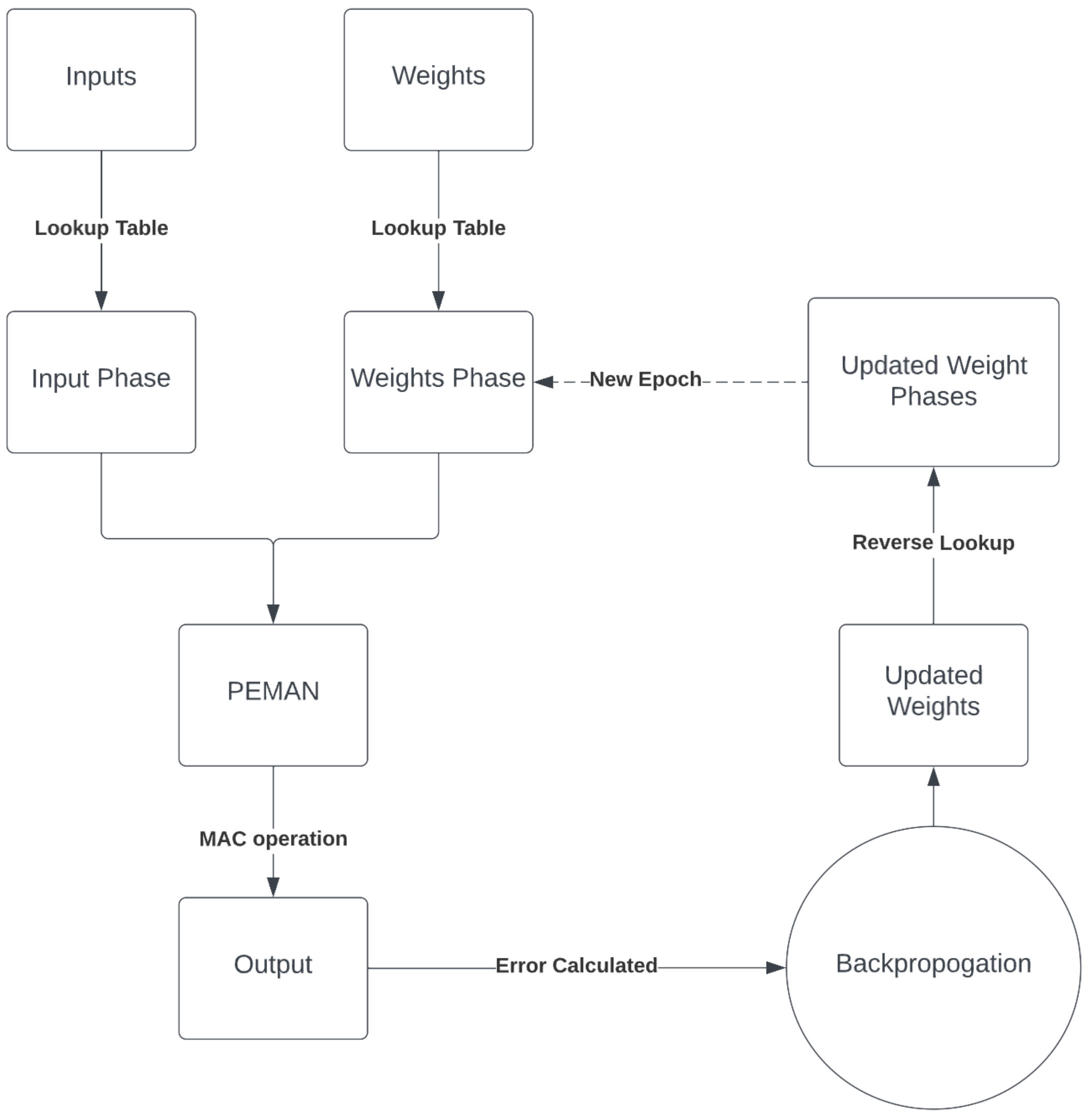

Fig. 2.4: Algorithm for training on PEMAN

2.5 PEMAN Structure and Handwritten Digit Recognition

In order the study the practical implications of using such an approach, the handwritten digit recognition problem is used. The handwritten digit recognition problem is a classification problem where the goal is to classify a given handwritten digit into one of the 10 classes. The dataset used for this problem is the MNIST dataset. The MNIST dataset consists of 60,000 training images and 10,000 testing images. The images are grayscale images of size 28x28. The dataset is split into training and testing sets with a ratio of 6:1.

The model used for this problem is a simple feedforward neural network with 1 hidden layer. The input layer has $28 \cdot 28 = 784$ parameters. The hidden layer has 100 parameters. The output layer has 10 parameters. The activation function used is the ReLU function. The loss function used is the cross entropy loss function. The optimizer used is the Adam optimizer. The model is trained for 10 epochs with a batch size of 64.

The model was trained in the same architecture but two different times. The first time, post training quantization technique was applied to quantize the wieghts to 8 bits and then the inference was run. In the second case, all the inputs and weights were quantized to various bit resolution while training itself. This helps us simulate the difference between training on PEMAN versus only inference through PEMAN.

The model when only done post-training quantization achieved an accuracy of 98.10%. Figure 2.5 shows the accuracy values achieved by the same model when trained with pre-quantized parameters.

From this plot, we can infer that at 8 bit resolution, we have an approximate accuracy of 96.50 %, which is lower that 98.10% achieved by the post-training quantization technique. However, we can also see that the accuracy is still quite high and the drop is not significant. This means that we can train the model with reduced bit operation and still achieve a high accuracy. This also means that we can train the model faster as the reduced bit operation is faster than the full bit operation.

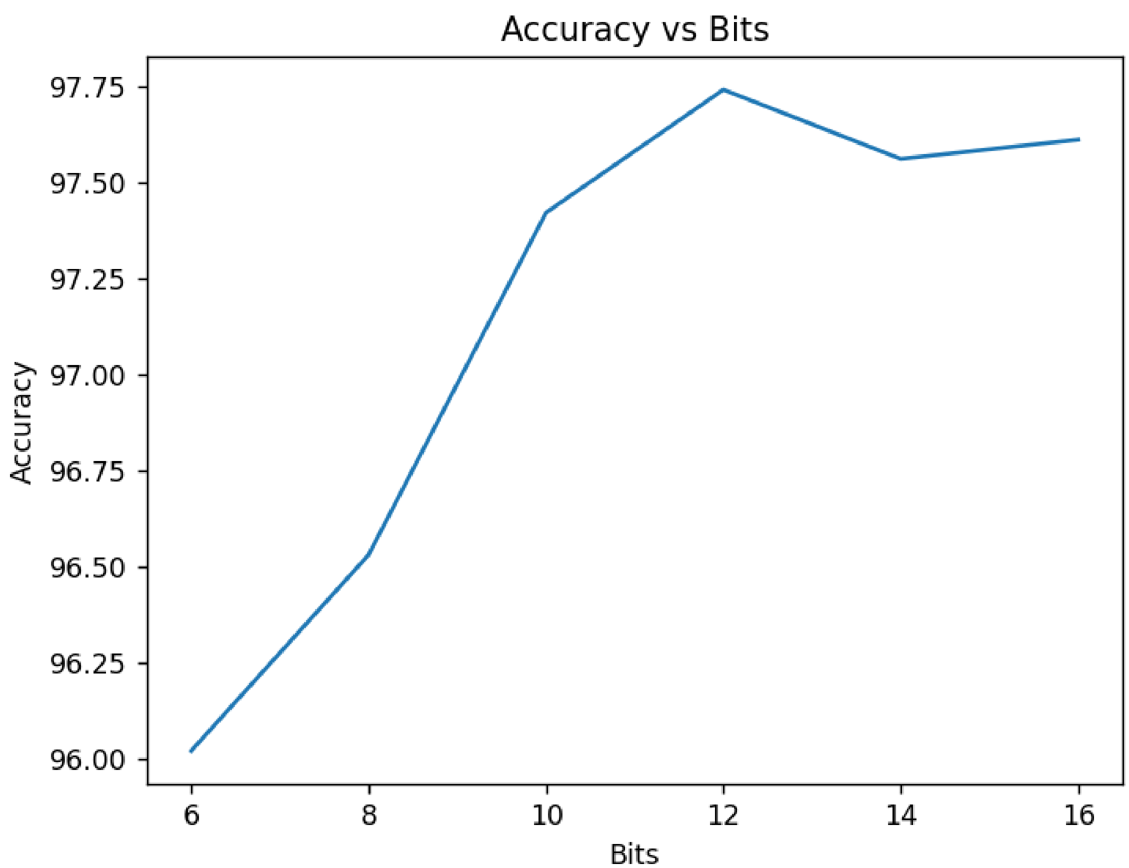

Fig. 2.5: Accuracy values achieved by the model when trained with pre-quantized parameters

Chapter 3

Algorithm I

give details of your algorithm

3.1 Conclusion

In this chapter, we proposed a distributed algorithm for construction of xyz. The complexity of this algorithm is $O(n \log n)$. Next chapter presents another distributed algorithm which has linear time complexity based on xyz.

Chapter 4

Algorithm II

The algorithm presented in previous chapter has $O(n)$ time complexity. We further propose another distributed algorithm in this chapter based on xyz which has linear time complexity.

4.1 Construction

Write ...

4.2 Improved Method

Write...

4.3 Conclusion

In this chapter, we proposed another distributed algorithm for XYZ. This algorithm has both time complexity of $O(n)$ where n is the total number of nodes. In next chapter, we conclude and discuss some of the future aspects.

Chapter 5

Conclusion and Future Work

write results of your thesis and future work.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press.

[2] Y. Han, S. Xiang, Y. Zhang, S. Gao, A. Wen, and Y. Hao, "An all-MRR-based photonic spiking neural network for spike sequence learning," vol. 9, no. 2. [Online]. Available: https://www.mdpi.com/2304-6732/9/2/120

[3] L. De Marinis, A. Catania, P. Castoldi, G. Contestabile, P. Bruschi, M. Piotto, and N. Andriolli, "A codesigned integrated photonic electronic neuron."

[4] NVIDIA T4 Tensor Core GPU for AI Inference — NVIDIA Data Center. [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-t4/

[5] F. Shokraneh, M. S. Nezami, and O. Liboiron-Ladouceur, "Theoretical and experimental analysis of a 4 × 4 reconfigurable MZI-Based linear optical processor," vol. 38, no. 6, pp. 1258–1267.

[6] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat,

M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "PaLM: Scaling language modeling with pathways."