A High-Performance, Hardware-based Deep Learning System for Disease Diagnosis

Ali Siddique[1], Muhammad Azhar Iqbal[2], Muhammad Aleem[3], and Jerry Chun-Wei Lin[4]

[1]**National University of Computer and Emerging Sciences, Lahore, Pakistan**
[2]**School of Computing and Communications, Lancaster University, UK**
[3]**National University of Computer and Emerging Sciences, Islamabad, Pakistan**
[4]**Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Norway**

Corresponding author:
Jerry Chun-Wei Lin[4]

Email address: jerrylin@ieee.org

ABSTRACT

Modern deep learning schemes have shown human-level performance in the area of medical science. However, the implementation of deep learning algorithms on dedicated hardware remains a challenging task because modern algorithms and neuronal activation functions are generally not hardware-friendly and require a lot of resources. Recently, researchers have come up with some hardware-friendly activation functions that can yield high throughput and high accuracy at the same time. In this context, we propose a hardware-based neural network that can predict the presence of cancer in humans with 98.23% accuracy. This is done by making use of cost-efficient, highly accurate activation functions, Sqish and LogSQNL. Due to its inherently parallel components, the system can classify a given sample in just one clock cycle, i.e., 15.75 nanoseconds. Though this system is dedicated to cancer diagnosis, it can predict the presence of many other diseases such as those of the heart. This is because the system is reconfigurable and can be programmed to classify any sample into one of two classes. The proposed hardware system requires about 983 slice registers, 2655 slice look-up tables, and only 1.1 kilo-bits of on-chip memory. The system can predict about 63.5 million cancer samples in a second and can perform about 20 Giga operations per second. The proposed system is about 5 to 16 times cheaper and at least four times speedier than other dedicated hardware systems using neural networks for classification tasks.

1 INTRODUCTION

Deep learning is a subset of machine learning that does not involve much human effort and does not require handcrafting of features Guo et al. (2019). In fact, by using deep learning techniques, machines and systems learn by themselves. It is important to note that deep learning and neural networks are not two separate ideas or techniques; any neural network that has two or more layers is considered 'deep'. Neural networks find applications in stock market prediction, agriculture, medical sciences, document recognition, and facial recognition, among others Awais et al. (2019); Nti et al. (2021); Zhou et al. (2019); Guan (2019); Chen et al. (2020); Kim et al. (2020); Lammie et al. (2019). The process of learning is usually carried out using 'backpropagation', a supervised learning technique in which the parameters of a neural network are adjusted according to a predefined error function. The parameters that give the lowest error at the output are selected as the optimal parameters.

It must be noted that hardware throughput is directly dependent on underlying algorithms. Therefore, efficient ANN algorithms and activation functions need to be devised if real-time neural processing is required. Sometimes, accuracy has to be sacrificed to support low-delay classification at low costs. The required level of accuracy, latency, speed, etc. depends on the underlying application, as shown in Table 1.

A major challenge facing deep learning researchers is the growing complexity of neural networks, which makes them unsuitable for execution on general-purpose processors. It is a fact that deep learning has traditionally been carried out on general-purpose computers. However, with time, neural networks

Table 1. Features and requirements of various deep-learning application areas.

Application	Required Latency	Required Accuracy	Cost
Military	Low	High	High
Medical Sciences	Medium	High	Medium-High
Video Surveillance	Low	Medium	Medium-High
Agriculture	High	Low	Low
Digit Classification	Medium-High	Low-Medium	Low
Stock Market	Low	High	High

have grown extremely large and deep. Therefore, modern neural networks cannot be efficiently trained and/or executed on a general-purpose computer Lacey et al. (2016); Merolla et al. (2014). For efficient processing and training, specialized hardware-based neural networks are required. Since dedicated hardware platforms such as field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) offer a low-power, high-speed alternative to conventional personal computers (PCs), they are becoming popular day by day. However, such platforms come with their own set of challenges: these platforms and costly and inflexible, and their cost-efficiency is highly dependent on the underlying algorithms. Therefore, it is of utmost importance to develop algorithms and activation functions that are friendly to the hardware.

Conventional activation functions such as sigmoid, softmax, and hyperbolic tangent (TanH) yield high accuracy but are not suitable for hardware implementations. This is because they involve division and many other hardware-inefficient operations Wuraola et al. (2021). Though rectified linear unit (ReLU) Nair and Hinton (2010) is an extremely powerful activation function that does not require any costly elements and is the most hardware-friendly function to date, sometimes it does not produce good results. This is because it suffers from *dying neurons*, since it cancels out all the negative input values Lu (2020). If output neurons receive negative inputs only, the system will always produce *zero* for all the output neurons, and no sample will be correctly classified. This is why scientists have come up with functions that are not only accurate but are friendly to hardware platforms. Such activation functions do not involve any costly functions such as exponentials or long divisions Wuraola et al. (2021). Two of these functions are *squish* and *square logistic sigmoid (LogSQNL)*. Both these functions do not require any storage element or division operation. This is the reason why we adopt these functions for neuronal implementation in the proposed system. The Sqish and LogSQNL are shown in Fig. 1 (a) and Fig. 1 (b), respectively.

In this article we present a system based on *sqish* and *square logistic sigmoid (Log_SQNL)* functions Wuraola et al. (2021) for breast cancer classification. The system is described in Verilog language at the register-transfer level (RTL), is implemented on a low-end Virtex 6 FPGA, and can classify a given sample, with 98.23% accuracy, into one of two classes: *benign* and *malignant*. Since most of the non-image datasets regarding disease recognition have two classes and less than 30 features, the proposed system can be used for the diagnosis of almost all diseases. The system is programmable; to diagnose a different disease, all that a user has to do is reprogram the device and use a new set of weights. The proposed system consumes only 983 slice registers, 2655 slice look-up tables, 234 DSP48 elements, and 33 block random access memories (BRAMs). The system is about **5-16 times cheaper** and **at least four times speedier** than many modern systems such as Sarić et al. (2020); Farsa et al. (2019); Shymkovych et al. (2021); Thanh et al. (2016); Ortega-Zamorano et al. (2016); Zhang et al. (2020); Tiwari and Khare (2015). These excellent results can be attributed to the following features:

- High Degree of Parallelism: all the required operations can be completed in a single clock cycle.

- Pipelining: use of pipeline registers at appropriate places in the system to improve throughput.

- Cost-Efficient Functions: use of Sqish in hidden layers and LogSQNL at the output layer. None of these functions require costly operations such as exponentials. Both these functions can be realized in hardware using combinational MAC computers. Since FPGAs contain a lot of DSP48 elements, multiplications can be efficiently performed.

- Proper HyperParameter Tuning: Hyperparameter tuning is extremely important for high network accuracy. The proposed network has been carefully tuned using the so-called 'grid search' Zheng, Alice (2015).

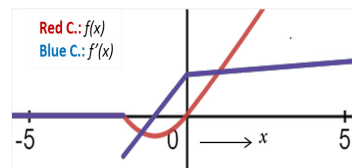
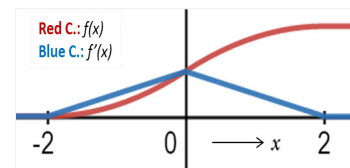
The rest of this paper is organized as follows. Section 2 presents a critical review of various high-performance activation functions and inference systems. The proposed scheme along with its hardware implementation is given in detail in Section 3. The test conditions and performance metrics are mentioned in Section 4. The results obtained by using the proposed scheme are given in Section 5; the system is also compared against other state-of-the-art systems to prove that the proposed scheme outperforms other schemes when it comes to classification accuracy, precision, recall, and hardware efficiency. The discussion is concluded in Section 6.

(a) Sqish function and its derivative.

(b) Log SQNL function and its derivative.

Figure 1. Sqish and Log SQNL functions along with their derivatives.

2 LITERATURE REVIEW

Since high-accuracy hardware systems are in demand, various hardware-friendly algorithms and high-performance hardware ANN systems have been presented in the literature. It is to be mentioned, however, that only a few hardware-based systems for disease diagnosis are presented in the literature. Most of the algorithms concerning medical sciences, healthcare, and disease diagnosis are not intended for any hardware implementation. For example, the algorithm presented in Kilicarslan et al. (2020) can perform cancer classification but the authors test their scheme only on software.

A recently-developed activation function is 'swish' Ramachandran, Prajit, and Barret Zoph, and Quoc V. Le. (2018). According to available reports, swish is more accurate than ReLU, especially when the network is very deep. Unlike ReLU, it is universally differentiable, i.e., the function has a valid derivative at all points on the real line. Like ReLU, the swish activation function solves the gradient vanishing problem. Swish allows negative values to backpropagate to the input side, which is impossible in the case of ReLU, since ReLU completely cancels out the negative values Nair and Hinton (2010). However, swish is not a hardware-friendly function since it involves division and a lot of other costly elements Wuraola et al. (2021).

In Sarić et al. (2020), the authors present an FPGA-based system that can predict two types of epileptic seizures. Moreover, the system can predict whether a seizure is present in the first place. The overall accuracy of the system is around 95.14%. The system in Shymkovych et al. (2021) implements a simple neural network that has 4-5 synapses. The system has four Gaussian neurons, which are radial basis functions (RBFs). The system has not been tested on any well-known dataset and the purpose of the system is to demonstrate the hardware efficiency of the proposed scheme.

A high-performance activation function based on exponential units is proposed in Clevert et al. (2015) that obviates the need for batch normalization. Batch normalization is an extremely costly process that requires big storage elements as well as large computational elements. Therefore, ELU is a good function in that context. However, ELU suffers from the same problems that many other activation functions do: ELU is still a cost function that is not as hardware-efficient as ReLU.

Another recently-proposed activation function is ReLTanH Wang et al. (2019). According to its developers, it has all the nice qualities possessed by hyperbolic tangent (TanH) and at the same time, it solves the problem of gradient vanishing. A big flaw in their work is that they apply the proposed function only to the diagnosis of rotating machinery faults. They do not perform any extensive tests. Moreover, they do not implement their scheme on any dedicated hardware platform, due to which it is quite hard to determine the hardware efficiency of their algorithm and functions. However, one thing that can certainly be said about their function is that the function is not friendly to the hardware because it, like TanH, requires division and other costly operations.

A hardware system for weed classification is proposed in Lammie et al. (2019). The system finds applications in agricultural robots. The system they design uses binary weights (±1). Due to this property, the system can operate with 98.83% accuracy while having small computational units and storage elements.

134 Eyeriss is another system that relies on extensive data reuse to reduce energy consumption Chen et al.
135 (2016). The system uses convolutional neural networks (CNNs) along with row- and column-wise data
136 reuse techniques. In this way, the system achieves both high accuracy and low energy consumption.
137 In Tiwari and Khare (2015), the researchers first implement the sigmoid function using 'Coordinate
138 Rotation Digital Computer (CoRDiC)' technique and then implement a complete neural network having
139 35 synapses using such CoRDiC neurons. The minimum value of the root mean squared error (RMSE)
140 between the CoRDiC sigmoid and the original sigmoid is $1.67E$-11.

141 A hardware-based radial basis function neural network (RBF-NN) capable of online learning is
142 proposed in Thanh et al. (2016). The network has 20 synapses and uses stochastic gradient descent (SGD)
143 for on-chip learning. To increase hardware efficiency, the exponential terms are approximated using
144 Taylor series expansion and look-up tables. The system has been implemented on a Cyclone-IV FPGA.
145 The forward computation component consumes 14,067 logic elements and the SGD learning algorithm
146 component consumes 17,309 logic elements. A comprehensive comparison of various modern works is
147 presented in Table 2. In Table 2, *C&R* stands for *classification and regression*.

Table 2. Summary of the related work.

	Neuron	Algo.	Learning Platform	Implem. Platform	Accuracy	Synapses	H/W Efficiency	Application
Ramachandran, Prajit, and Barret Zoph, and Quoc V. Le. (2018)	Swish	BP	Software	Software	Extr. High	-	Extr. Low	C&R
Sarić et al. (2020)	Sigmoid	BP	Software	FPGA	High	-	Mod. High	Epil. Seizure C.
Shymkovych et al. (2021)	Gaussian	BP	Software	FPGA	High	4-5	Mod. High	Classification
Clevert et al. (2015)	ELU	BP	Software	Software	Extr. High	-	Moderate	C&R
Wang et al. (2019)	ReLTanH	BP	Software	Software	High	-	Low	Fault Diagnosis
Lammie et al. (2019)	Binary	BP	Software	FPGA	High	-	High	Weed Classif.
Chen et al. (2016)	Mixed	BP	Software	ASIC	High	-	High	Classification
Tiwari and Khare (2015)	Sigmoid	-	-	FPGA	High	35	Low	C&R
Thanh et al. (2016)	Radial	SGD	FPGA	FPGA	-	20	Low	C&R

3 PROPOSED METHODOLOGY

149 In order to understand how the proposed system works, it is extremely important to get familiarized with
150 a few basic concepts regarding ANN operation. Therefore, we first explain the basic ANN operation and
151 then explain the proposed ANN topology, learning scheme, and the proposed hardware system along with
152 its constituent components.

3.1 Basic ANN Operation
Network Inputs
155 The input values are first *standardized* in order to make them *zero-centered*. The process of standardization
156 follows Equation 2. In Equation 1, *X* represents the input vector, μ represents the average, and σ represents
157 the standard deviation of data samples. The process of standardization is visually represented in Fig. 2. It
158 is important to note that standardization is sometimes referred to as 'normalization' in literature, though
159 normalization is, in reality, different from standardization.

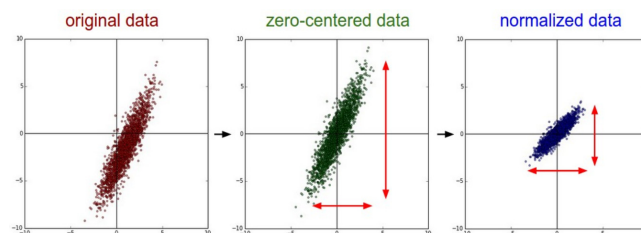



Figure 2. Standardization of input data Stanford University (2022).

$$X_{\text{standard.}} = \frac{X_{\text{original}} - \mu}{\sigma} \tag{1}$$

160 ***Accumulation and Activation***

161 These normalized/standardized inputs are multiplied by the corresponding weights and the resulting
162 products are then summed up (accumulated). A neuron is activated or deactivated based on the value of
163 this weighted sum.

164   The activation of a neuron is dictated by a so-called 'activation function'. Some of the various popular
165 activation functions are rectified linear unit (ReLU) Nair and Hinton (2010), swish Ramachandran, Prajit,
166 and Barret Zoph, and Quoc V. Le. (2018), exponential linear unit (ELU) Clevert et al. (2015), among
167 others. Every activation function has its own merits and demerits. ReLU, for example, is used to solve the
168 'gradient vanishing' problem that occurs in hidden layers during learning. However, ReLU completely
169 cancels out the negative region, due to which functions like Swish were developed. A detailed discussion
170 on this topic can be found in Ramachandran, Prajit, and Barret Zoph, and Quoc V. Le. (2018).

171   The output values produced by the activated neurons are then multiplied by the corresponding weights
172 of the next layer and the process is repeated. To offset a neuron's value for better learning, a bias term b_j
173 is added to the weighted sum.

174 ***Classification and Backpropagation***

175 At the output layer, the neuron that is activated the most corresponds to the predicted class. The prediction
176 of an input sample corresponds to the completion of a single iteration of the forward pass.

177   In the backward pass, synaptic weights are modified according to an algorithm called 'backpropaga-
178 tion'. The basic idea is that the magnitude of synaptic weight updates is dictated by the magnitude of
179 output error. If a wrong prediction is made, the error (such as *mean squared error*) is computed and the
180 synaptic weights corresponding to that (wrong) neuron are decreased. At the same time, the synapses
181 corresponding to the correct output neuron are increased. With time, the network improves itself and
182 eventually achieves convergence. This algorithm will be explained at length in the coming sections.

183 **3.2 Proposed Network Topology and Actuators**

184 In this work, we have chosen *sqish* activation function for the hidden layer, and *square logistic sigmoid*
185 *(LogSQNL)* for the output layer. The Sqish function is morphologically similar to Swish, and LogSQNL
186 is similar in behavior to the traditional sigmoid. The beauty of these functions is that both of these can
187 be implemented in a single cycle using arithmetic and logic units (ALUs) only. As shown in Fig. 3, the
188 proposed system can take 30 (or less) input features, has five hidden neurons, and two outputs.

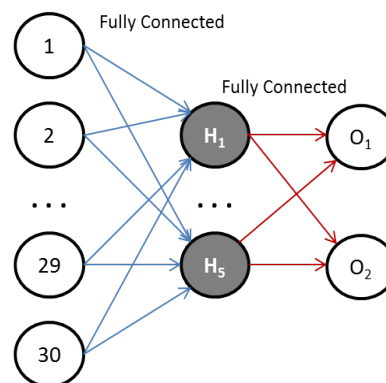



Figure 3. The proposed ANN topology.

189   The RTL schematic diagrams of the complete system and the predictor are shown in Fig. 4 and Fig.
190 5, respectively. The rectangular yellow and red boxes shown in this figure represent storage elements,
191 and white rectangular/square boxes represent computational blocks. As shown in Fig. 5, the activation
192 values corresponding to the two output neurons are given as input to the predictor. The predictor then
193 calculates the maximum value and the class corresponding to this maximum value is the predicted class.
194 The top level diagram of the proposed system is shown in Fig. 6. In Fig. 6, I_M represents M^{th} input. The
195 multiplier units (MUs) are responsible for multiplying weights with incoming inputs and the accumulator
196 is responsible for adding these products. The output of the multiplier-accumulator (MAC) unit is sent to
197 the appropriate actuator for neuronal activation.

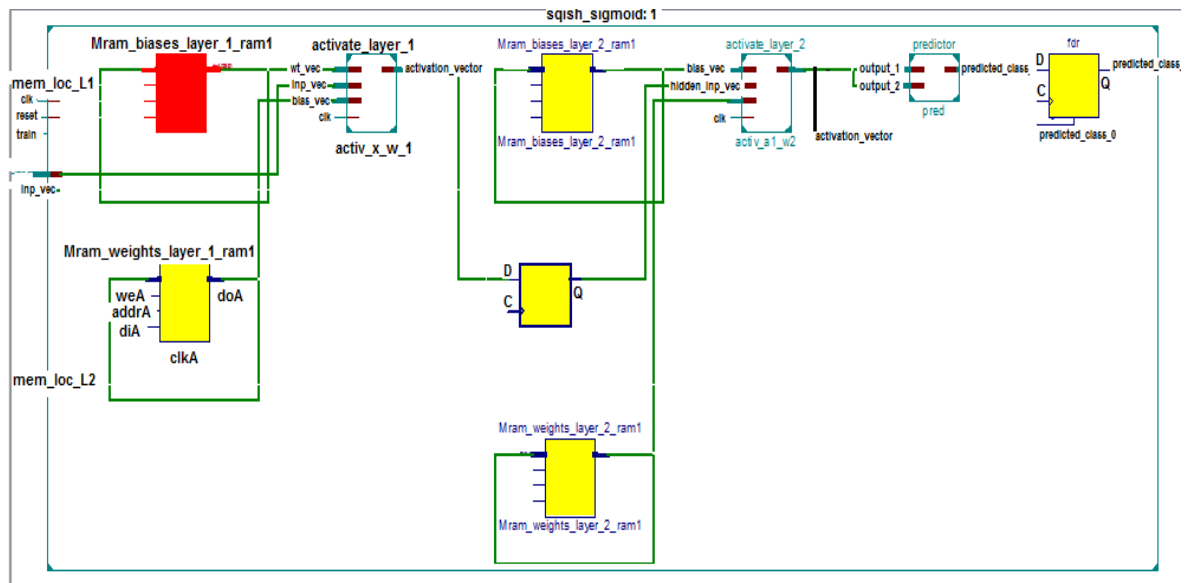



Figure 4. RTL Schematic of the complete system

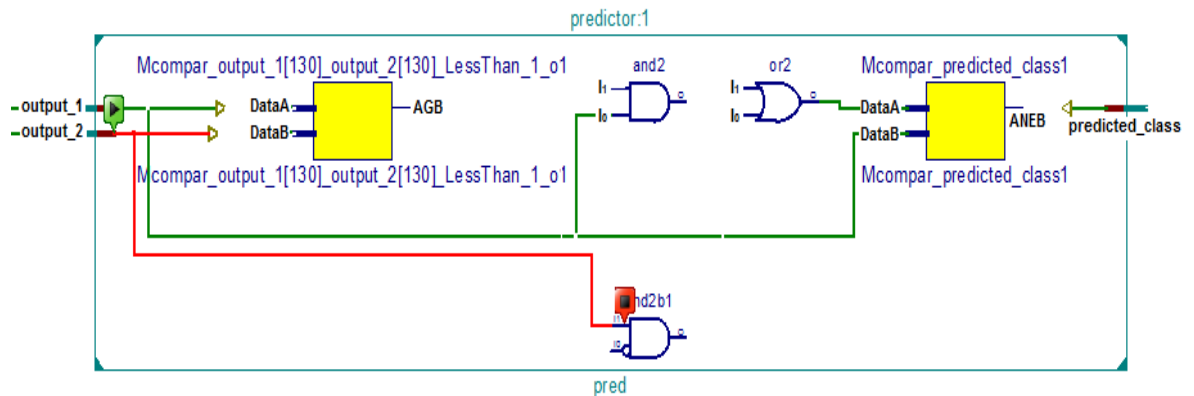



Figure 5. RTL schematic of the predictor.

3.3 Mathematical Setup

Now we give details of the complete mathematical setup. Here, weights are denoted by W_i and inputs are denoted by X_i. Biases are represented by b_j, and the weighted sum is represented by Z_j. Finally, activation values are represented by A_j. Here, the subscript j represents the postsynaptic neuron and the subscript i represents the presynaptic neuron. The following equations represent the complete mathematical process.

$$Z_1 = \sum_i (W_i \cdot X_i) + b_1$$

Since Sqish is used as the activation function for Layer 1 in the proposed scheme, $A_1(Z_1)$ is given by the following equation:

$$A_1 = \begin{cases} Z_1 + \dfrac{Z_1^2}{32} & Z_1 \geq 0 \\[2mm] Z_1 + \dfrac{Z_1^2}{2} & -2 \leq Z_1 < 0 \\[2mm] 0 & Z_1 < -2 \end{cases}$$

The Layer 1 activation vector is then passed as input to Layer 2 in order to obtain the weighted sum Z_2, as shown in the following equation.

$$Z_2 = \sum_i (W_i \cdot A_1) + b_2$$

207   The LogSQNL neurons in the output layer are activated according to the following rule:

$$A_2 = \begin{cases} 1 & Z_2 > 2 \\ \left(Z_2 - \frac{Z_2^2}{4}\right)\frac{1}{2} + \frac{1}{2} & 0 \leq Z_2 \leq 2 \\ \left(Z_2 + \frac{Z_2^2}{4}\right)\frac{1}{2} + \frac{1}{2} & -2 \leq Z_2 < 0 \\ 0 & Z_2 < -2 \end{cases}$$

208   The derivative of LogSQNL function is given by Equation 2 and the dependence of the loss function
209 on Layer 2 weight vector and bias vector is given by Equation 3 and Equation 4. The derivative of Sqish
210 function is given by Equation 5 and the dependence of the loss function on Layer 1 weight vector and
211 bias vector is given by Equation 6 and Equation 7.

$$\frac{\partial A_2}{\partial Z_2} = \begin{cases} \frac{2-Z_2}{4} & 0 \leq Z_2 \geq 2 \\ \frac{2+Z_2}{4} & -2 \leq Z_2 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\frac{\partial L}{\partial W_2} = \begin{cases} (A_2 - y) \cdot A_1 \cdot \frac{2-Z_2}{4} & 0 \leq Z_2 \geq 2 \\ (A_2 - y) \cdot A_1 \cdot \frac{2+Z_2}{4} & -2 \leq Z_2 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\frac{\partial L}{\partial b_2} = \begin{cases} (A_2 - y) \cdot \frac{2-x}{4} & 0 \leq Z_2 \geq 2 \\ (A_2 - y) \cdot \frac{2+x}{4} & -2 \leq Z_2 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$\frac{\partial A_1}{\partial Z_1} = \begin{cases} 1 + \frac{Z_1}{16} & Z_1 \geq 0 \\ 1 + Z_1 & -2 \leq Z_2 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\frac{\partial L}{\partial W_1} = \begin{cases} \frac{(A_2 - y) \cdot A_1 \cdot (2 - Z_2) \cdot W_2 \cdot (16 + Z_1) \cdot X_1}{64} & \\ & 0 \leq Z_1 \leq 2; 0 \leq Z_2 \leq 2 \\ \frac{(A_2 - y) \cdot A_1 \cdot (2 + Z_2) \cdot W_2 \cdot (1 + Z_1) \cdot X_1}{4} & \\ & -2 \leq Z_1 < 0; -2 \leq Z_2 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$\frac{\partial L}{\partial b_1} = \begin{cases} \frac{(A_2 - y) \cdot A_1 \cdot (2 - Z_2) \cdot W_2 \cdot (16 + Z_1)}{64} & \\ & 0 \leq Z_1 \leq 2; 0 \leq Z_2 \leq 2 \\ \frac{(A_2 - y) \cdot A_1 \cdot (2 + Z_2) \cdot W_2 \cdot (1 + Z_1)}{4} & \\ & -2 \leq Z_1 < 0; -2 \leq Z_2 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

3.4 Proposed Hardware System

As mentioned before, there are two layers in the proposed system: Layer 1 and Layer 2. Both these layers have their memories to store weights. It is pertinent to mention that all the weights are stored in the chip. The total number of weights in the system is 160 since there are 160 synapses. The on-chip weight memory consumes 1.1 kilobits. The required weights that are fetched from the corresponding memory are then multiplied by the respective layer inputs using the multiplier units (MUs) shown in Fig. 6. The multiplication is carried out using the built-in DSP48 elements. The resulting products are then summed up using an accumulator that contains an array of adders. This process is carried out for all the neurons in a layer. In the end, N weighted sums are obtained, where N represents the number of neurons in a layer. These N weighted sums are passed to their respective actuators for neuronal activation. The actuator then passes these calculated values onto the next layer and the same process is repeated. The complete structure of the proposed hardware system is shown in Fig. 6.

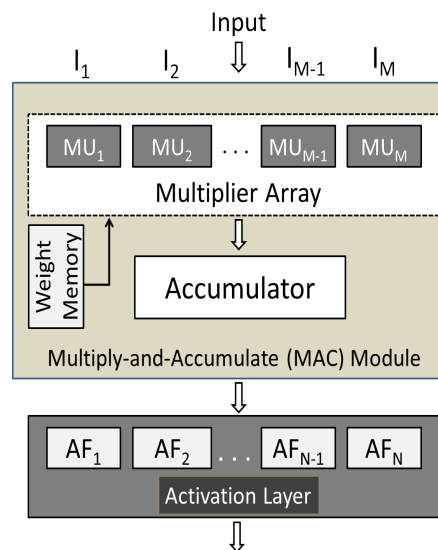



Figure 6. Top-level view of the proposed hardware system.

As mentioned earlier, Sqish neurons are used in Layer 1 and LogSQNL neurons are used in Layer 2. The structure of a Sqish neuron is shown in Fig. 7 and that of a LogSQNL neuron is shown in Fig. 8. A sqish neuron can be implemented using two multiplexers (MUXes), two adders, two shifters and two multipliers. A LogSQNL neuron, on the other hand, consumes more resources than Sqish. A LogSQNL neuron can be implemented in hardware using three MUXes, four shifters, four adders, and two multipliers. However, since there are only two outputs in the proposed system for binary classification (one-hot encoding), the implementation of LogSQNL neurons is not a big deal.

Interestingly, the distribution of data in the dataset under consideration, i.e., *Wisconsin Breast Cancer (WBC)* University of California (2022) is highly non-uniform. The standard deviation of the data is extremely large. The final input values obtained after standardization require 11 bits, where 4 bits are reserved for the integer part and 7 bits are reserved for the mantissa (fractional part). As per our observations and calculations, the classification accuracy is more sensitive to the fractional part than the integer part. Therefore, more bits are reserved for the mantissa.

4 TEST CONDITIONS AND PERFORMANCE METRICS

In this section, we mention the test conditions under which the evaluation and comparisons are carried out. We also mention the philosophy behind the performance metrics used for evaluation.

4.1 Test Conditions

Since the system has been developed for cancer diagnosis, the dataset used for experimentation is *Wisconsin Breast Cancer (WBC)* University of California (2022). This dataset has 30 features, 569 samples and 2 classes (*benign* and *malignant*). As per rules, about 80% samples have been used for

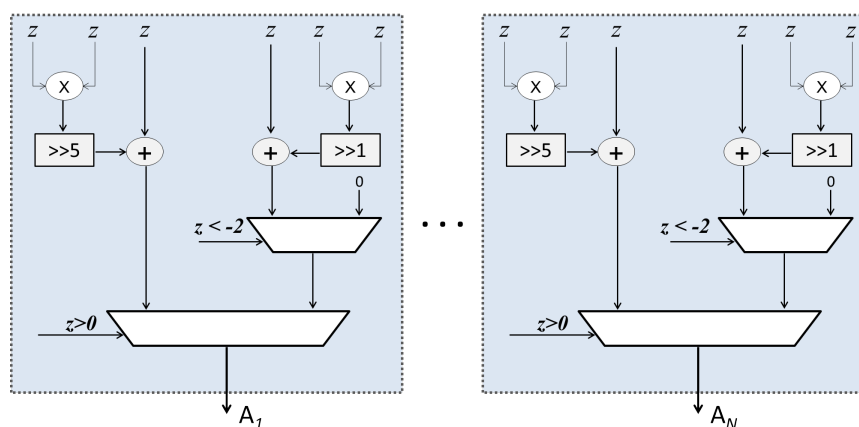



Figure 7. Internal structure of the Sqish (hidden) layer.

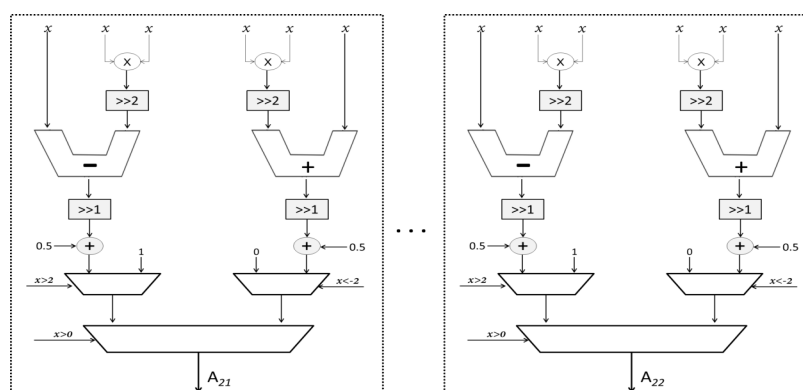



Figure 8. Structure of the output LogSQNL array.

training and 20% have been used for evaluation. To achieve class balance, some samples are picked up
from the 'benign' class and others are picked up from 'malignant' class. We use Python for evaluation of
the proposed scheme. The hardware is described in Verilog language at the register transfer level (RTL).
The learning rate is kept equal to $\frac{1}{3}$. The momentum is equal to 0.9. The data is processed in batches to
achieve high accuracy; the batch size used in the proposed system is 100. The network is trained for 4300
epochs. All these hyperparameter values have been found through empirical tuning using the so-called
'grid search' Zheng, Alice (2015). The specifications of the platform on which all the tests are carried out
are given in Table 3.

Table 3. Specifications of the platform used for performance evaluation.

Processor	Intel Core i7-5500 (4 CPUs)
Memory	8.00 GB
Operating System	Windows 8.1
System Type	x64 (64-bit OS)

4.2 Performance Metrics

The metrics used for the evaluation of the proposed scheme are classification accuracy, precision, recall,
hardware implementation cost, and system throughput. The classification accuracy is simply defined as
the number of correctly-classified samples out of the total number of samples. In the context of disease
diagnosis, accuracy is not a good measure of system performance. Therefore, we use *precision* and *recall*
in order to properly quantify performance. The precision and recall are defined in Equation 8 and Equation
9 respectively. Since these metrics are very common, we believe there is no need to discuss them in detail
here. In Equation 8 and Equation 9, TP stands for 'true positive', TN stands for 'true negative', FN stands
for 'false negative', and FP stands for 'false positive'.

$$Precision = \frac{TP}{TP + FP} \qquad (8)$$

$$Recall = \frac{TP}{TP + FN} \qquad (9)$$

To evaluate hardware efficiency, we use two metrics: the number of resources (number of slice registers, number of slice look-up tables, number of block memories, and DSP elements) consumed by the system, and system throughput. The throughput is defined in two ways: *the number of multiply-and-accumulate (MAC) operations that can be performed in a second*, and *the number of input samples that can be processed by the system in a second.*

5 RESULTS AND DISCUSSION

Here, the proposed system is compared with other state-of-the-art systems such as Sarić et al. (2020); Farsa et al. (2019); Shymkovych et al. (2021); Thanh et al. (2016); Ortega-Zamorano et al. (2016); Zhang et al. (2020); Tiwari and Khare (2015) in terms of classification accuracy, throughput, and implementation cost. We demonstrate how the proposed scheme is better than other traditional as well as contemporary schemes, especially for disease diagnosis.

5.1 Classification Accuracy, Precision, and Recall

As per obtained results, the system can predict the type of cancer with 98.23% accuracy. Moreover, the average precision of the proposed system is 97.5% and recall is around 98.5%. The classification report and the confusion matrix for the proposed system are given in Table 4 and Table 5, respectively. Moreover, the proposed system is compared with many other state-of-the-art systems in terms of classification accuracy in Table 6. The classification accuracy as a function of epochs is presented in Fig. 9a, and the confusion matrix is visually shown in Fig. 9b.

Table 4. Classification report: The proposed system.

	Precision	Recall
0 (benign)	0.95	1.00
1 (malignant)	1.00	0.97
	0.975	0.985

Table 5. Confusion Matrix: The proposed system.

TP = 38	**FP** = 0
FN = 2	**TN** = 73

Table 6. Classification Accuracy Comparisons

	Wuraola et al. (2021)	Aljarah et al. (2018)	Sarić et al. (2020)	Zhang et al. (2020)	Farsa et al. (2019)	Ortega-Zamorano et al. (2016)	**Proposed**
Features	784	5	8	4	25	4-35	30
Classes	10	3	2	3	2	2-6	2
Synapses	102k	96	153	144	130	≥ 84	160
Samples	70k	822	699	1000	8	<1000	569
Accuracy (%)	96.71	95.14	98.32	96	73-89	88.26	$\approx$**98.23**

5.2 Implementation Cost and Throughput Comparisons

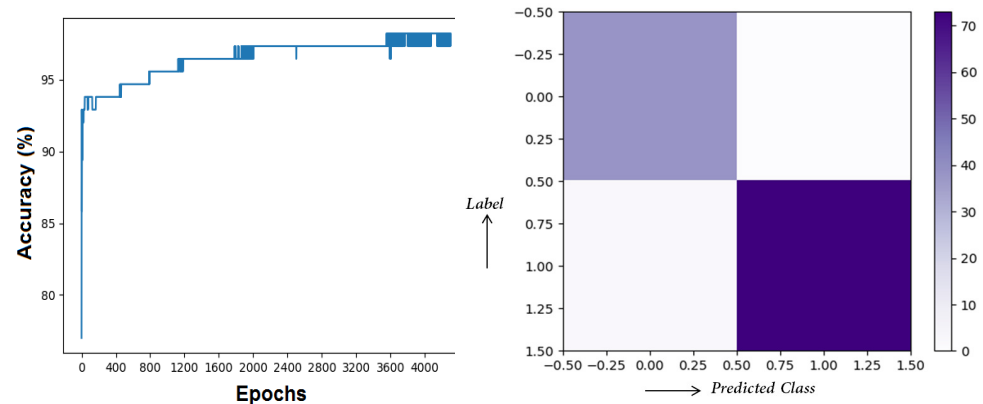
The use of Sqish and Log_SQNL Wuraola et al. (2021) allows the processing of one sample in one clock cycle. In a single cycle, the system can perform all MAC operations and can activate all the neurons

(a) Accuracy as a function of Epochs.   **(b)** Confusion Matrix - The proposed Scheme.

Figure 9. Accuracy, Precision, and Recall yielded by the Proposed System

without using any divider or storage element. The multiplication operations can be performed using DSP48 multipliers that are abundantly available in an FPGA.

There are 160 synapses in the proposed neural system that operates at 63.487 MHz. The number of synaptic multiplications and additions to be performed are 160 and 153 respectively. Therefore, the system can perform 20 Giga operations in a second (GOPS). Since the system can classify one cancerous sample in one cycle ($\approx$15.75 ns), the system can classify about 63.5×10^6 (63.5 million) samples in a second. Since a sample contains 30 inputs, about 1.91×10^9 1-input samples can be classified by the proposed system in one second. The system is compared with other state-of-the-art systems in terms of implementation cost and throughput in Table 7 and Table 8 respectively.

Table 7. FPGA Implementation Cost Comparisons

System	Acc.	Synapses	S. Regs.	S. LuTs	Max. Freq.	Mults.	Platform	Learning
Farsa et al. (2019)	89%	130	1023	11,339	189 MHz	-	Virtex 6	Offline
Ortega-Zamorano et al. (2016)	88.3%	84	6766	13,062	Variable	12	Virtex 5	Online
Sarić et al. (2020)	95.14%	96	114	12,960	50 MHz	116	Cyclone IV	Offline
Tiwari and Khare (2015)	-	35	1898	3,124	-	154	Virtex 5	Offline
Shymkovych et al. (2021)	-	5	790	1195	10 MHz	14	Spartan 3	Offline
Prop.	$\approx$98.23%	160	983	2655	63.49 MHz	234	Virtex 6	Offline

Table 8. Throughput (TP) Comparisons

System	Synapses	Sample Size	NTP
Farsa et al. (2019)	130	25	4.73×10^9
Sarić et al. (2020)	96	5	0.25×10^9
Shymkovych et al. (2021)	5	4	0.04×10^9
Proposed	160	30	1.91×10^9

6 CONCLUSION

This paper presents a high-throughput, hardware-efficient training scheme that uses Sqish neurons in the hidden layer and sigmoid-like LogSQNL neurons in the output layer. Since these functions do not require multiple cycles to process, the proposed system—based on these functions—does not consume a lot of hardware resources and yields high throughput. With only 160 synapses, the system can classify a cancerous sample into one of the two classes: *benign* and *malignant*. The proposed hardware system requires only 1.1 kilobits of on-chip memory, and can process about 1.91×10^9 1-input samples in a second. In just one second, the system can process 63.5 million cancer samples, and can perform 20×10^9 MAC operations. The system is about 5-16 times cheaper and at least four times speedier than most state-of-the-art hardware solutions designed for similar problems. Moreover, the system is way more

accurate than most contemporary systems. An important thing worth mentioning here is that to improve accuracy even by 1%, a lot of extra hardware resources are required. Therefore, the improvement in accuracy obtained by using the proposed scheme must not be undermined. Though the proposed system is specifically designed for cancer classification, the system can perform binary classification on any data sample that has 30 features or less. This is because the proposed system uses reconfigurable memory that can be programmed using an external computer. In future, convolutional neural networks can be applied to high-resolution mammograms (and/or ultrasound images) for diagnosing CoViD-19, cancer, and other ailments.

REFERENCES

Aljarah, I., Faris, H., and Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, 22(1):1–15.

Awais, M., Iqbal, M. J., Ahmad, I., Alassafi, M. O., Alghamdi, R., Basheri, M., and Waqas, M. (2019). Real-time surveillance through face recognition using hog and feedforward neural networks. *IEEE Access*, 7:121236–121244.

Chen, K., Lu, S. M., Cheng, R., Fisher, M., Zhang, B. H., Di Maggio, M., and Bradley, J. P. (2020). Facial recognition neural networks confirm success of facial feminization surgery. *Plastic and reconstructive surgery*, 145(1):203–209.

Chen, Y.-H., Krishna, T., Emer, J. S., and Sze, V. (2016). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Farsa, E. Z., Ahmadi, A., Maleki, M. A., Gholami, M., and Rad, H. N. (2019). A low-cost high-speed neuromorphic hardware based on spiking neural network. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(9):1582–1586.

Guan, J. (2019). Artificial intelligence in healthcare and medicine: promises, ethical challenges and governance. *Chinese Medical Sciences Journal*, 34(2):76–83.

Guo, K., Zeng, S., Yu, J., Wang, Y., and Yang, H. (2019). A survey of fpga-based neural network inference accelerators. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 12(1):1–26.

Kilicarslan, S., Adem, K., and Celik, M. (2020). Diagnosis and classification of cancer using hybrid model based on ReliefF and convolutional neural network. *Medical Hypotheses*, 137:109577.

Kim, S., Park, S., Na, B., and Yoon, S. (2020). Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11270–11277.

Lacey, G., Taylor, G. W., and Areibi, S. (2016). Deep learning on fpgas: Past, present, and future. *arXiv preprint arXiv:1602.04283*.

Lammie, C., Olsen, A., Carrick, T., and Azghadi, M. R. (2019). Low-power and high-speed deep fpga inference engines for weed classification at the edge. *IEEE Access*, 7:51171–51184.

Lu, L. (2020). Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814.

Nti, I. K., Adekoya, A. F., and Weyori, B. A. (2021). A novel multi-source information-fusion predictive framework based on deep neural networks for accuracy enhancement in stock market prediction. *Journal of Big Data*, 8(1):1–28.

Ortega-Zamorano, F., Jerez, J. M., Urda Muñoz, D., Luque-Baena, R. M., and Franco, L. (2016). Efficient implementation of the backpropagation algorithm in fpgas and microcontrollers. *IEEE Transactions on Neural Networks and Learning Systems*, 27(9):1840–1850.

Ramachandran, Prajit, and Barret Zoph, and Quoc V. Le. (2018). Searching for activation functions.

Sarić, R., Jokić, D., Beganović, N., Pokvić, L. G., and Badnjević, A. (2020). Fpga-based real-time

355   epileptic seizure classification using artificial neural network. *Biomedical Signal Processing and*
356   *Control*, 62:102106.

357 Shymkovych, V., Telenyk, S., and Kravets, P. (2021). Hardware implementation of radial-basis neural
358   networks with gaussian activation functions on fpga. *Neural Computing and Applications*, pages 1–13.

359 Stanford University, C. (2022). CS231n Convolutional Neural Networks for Visual Recognition.

360 Thanh, N. P., Kung, Y.-S., Chen, S.-C., and Chou, H.-H. (2016). Digital hardware implementation of a
361   radial basis function neural network. *Computers & Electrical Engineering*, 53:106–121.

362 Tiwari, V. and Khare, N. (2015). Hardware implementation of neural network with sigmoidal activation
363   functions using cordic. *Microprocessors and Microsystems*, 39(6):373–381.

364 University of California, I. (2022). UCI Machine Learning Repository.

365 Wang, X., Qin, Y., Wang, Y., Xiang, S., and Chen, H. (2019). ReLTanh: An activation function with
366   vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault
367   diagnosis. *Neurocomputing*, 363:88–98.

368 Wuraola, A., Patel, N., and Nguang, S. K. (2021). Efficient activation functions for embedded inference
369   engines. *Neurocomputing*, 442:73–88.

370 Zhang, G., Li, B., Wu, J., Wang, R., Lan, Y., Sun, L., Lei, S., Li, H., and Chen, Y. (2020). A low-cost and
371   high-speed hardware implementation of spiking neural network. *Neurocomputing*, 382:106–115.

372 Zheng, Alice (2015). *Evaluating machine learning models: a beginner's guide to key concepts and*
373   *pitfalls*, chapter Hyperparameter Tuning. O'Reilly Media.

374 Zhou, J. T., Du, J., Zhu, H., Peng, X., Liu, Y., and Goh, R. S. M. (2019). Anomalynet: An anomaly
375   detection network for video surveillance. *IEEE Transactions on Information Forensics and Security*,
376   14(10):2537–2550.

Figure 1

Figure2

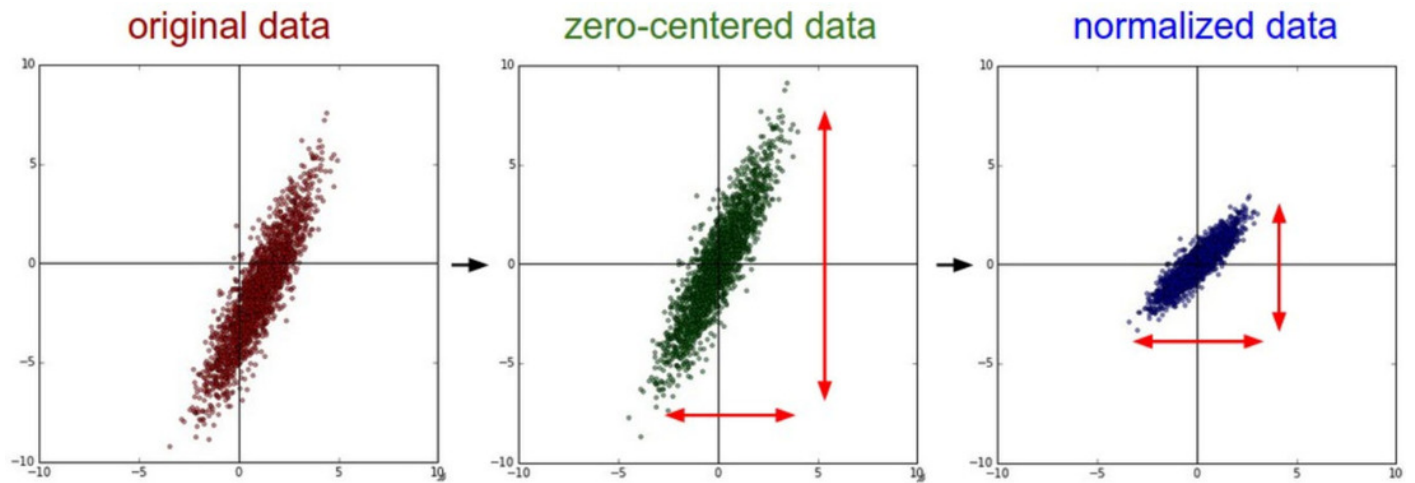



Figure 2

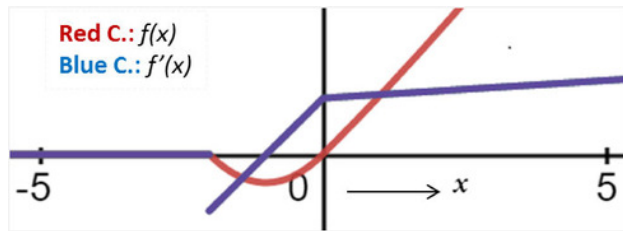
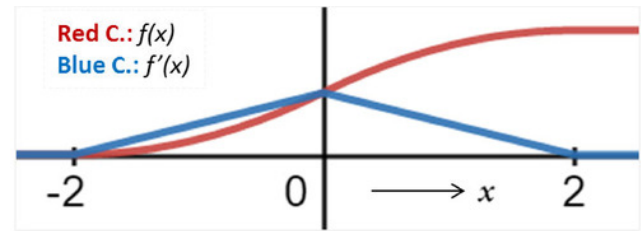
Figure1

(a) Sqish Function and its Derivative

(b) Log SQNL Function and its Derivative

Figure 3

Figure5

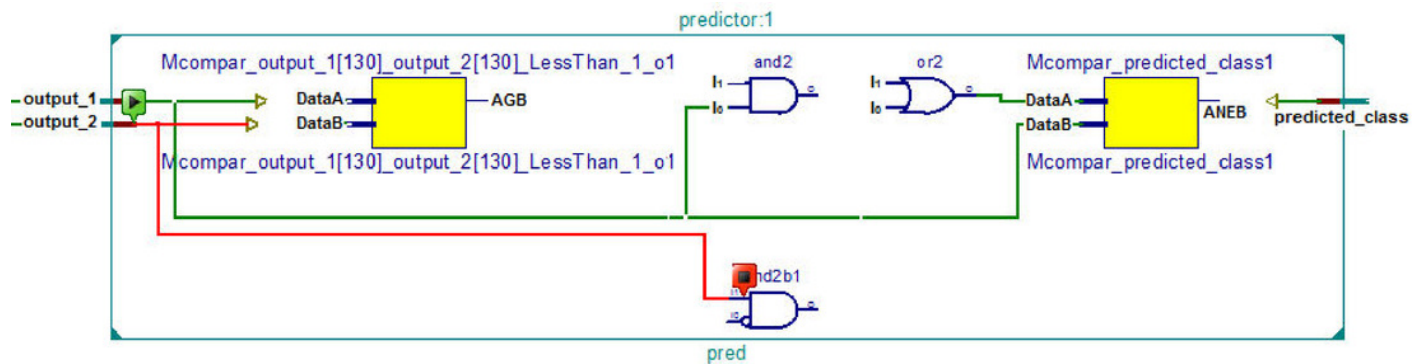



Figure 4

Figure3

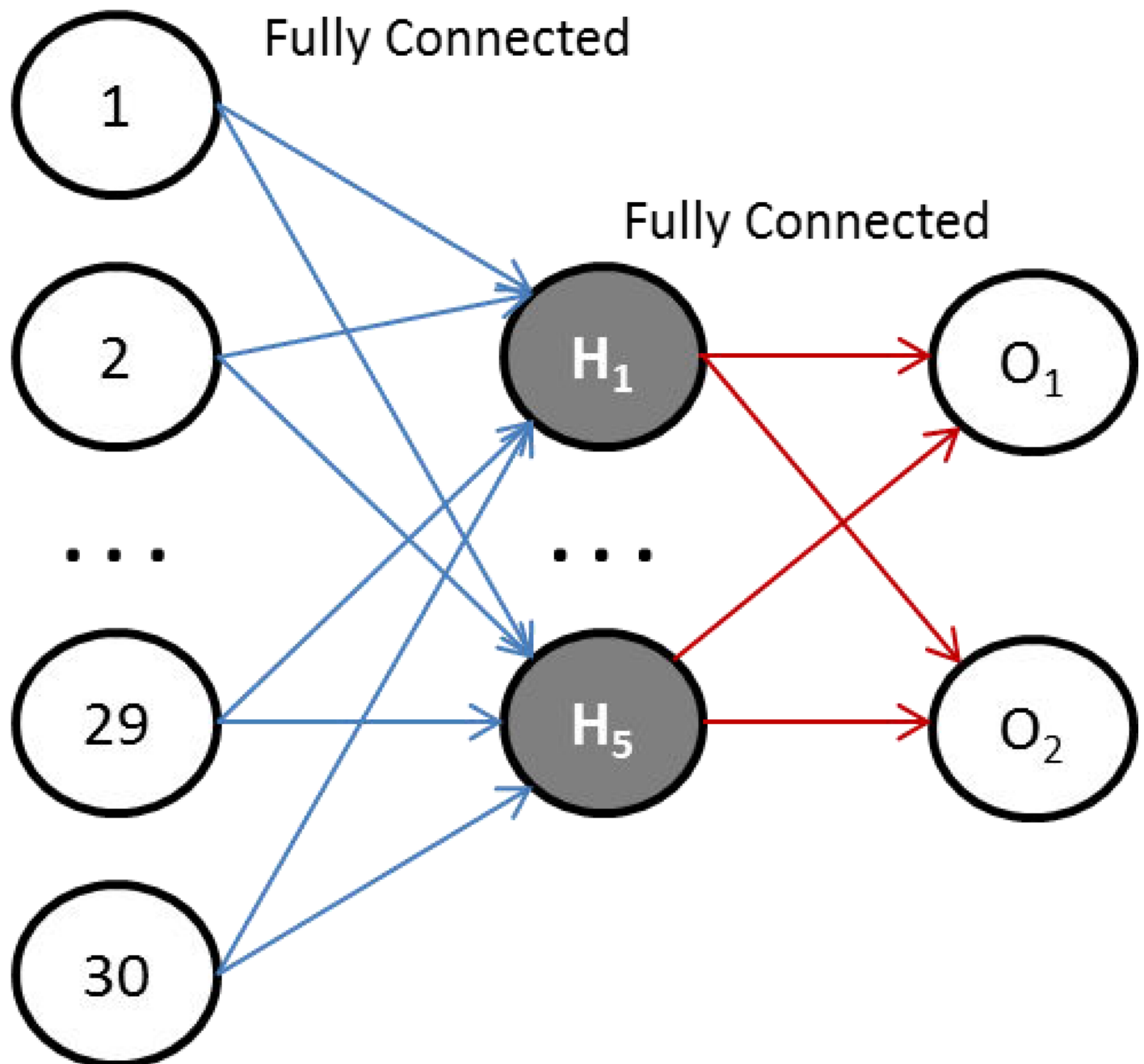



Figure 5

Figure7

Figure 6

Figure9

(a) Accuracy as a function of Epochs.   (b) Confusion Matrix - The proposed Scheme.

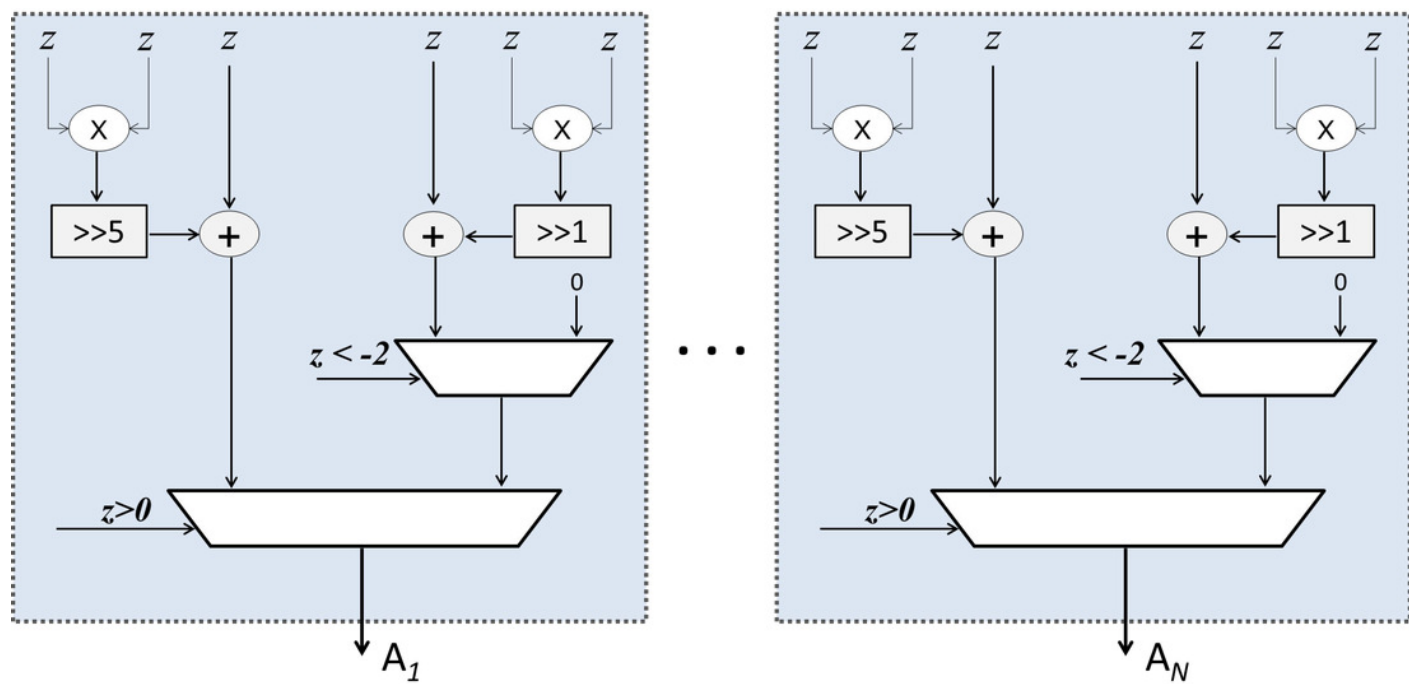
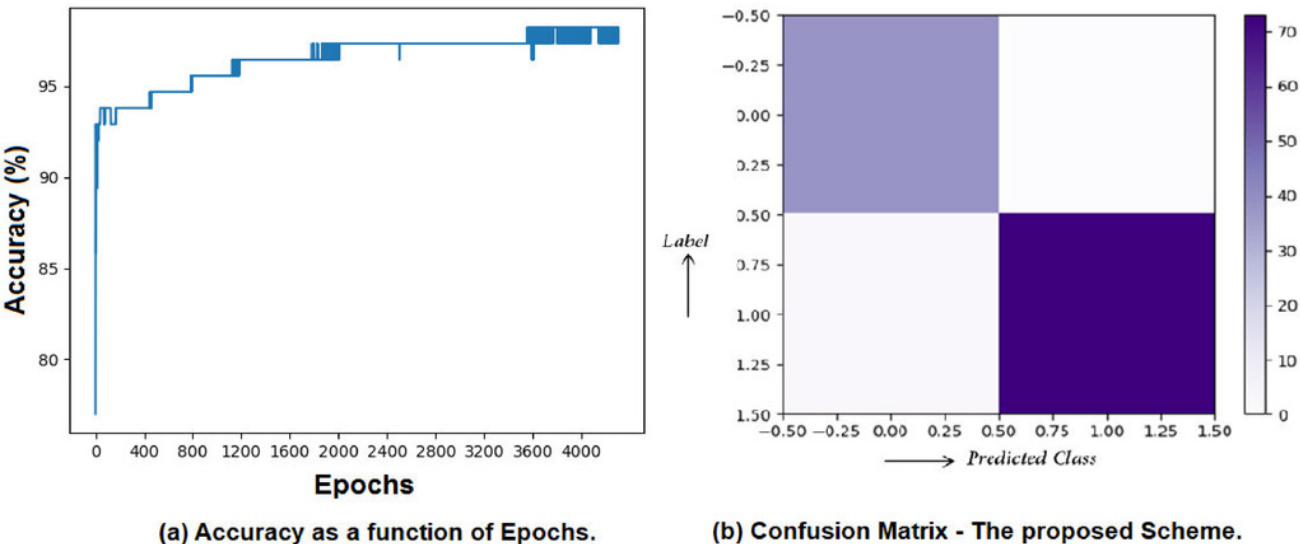
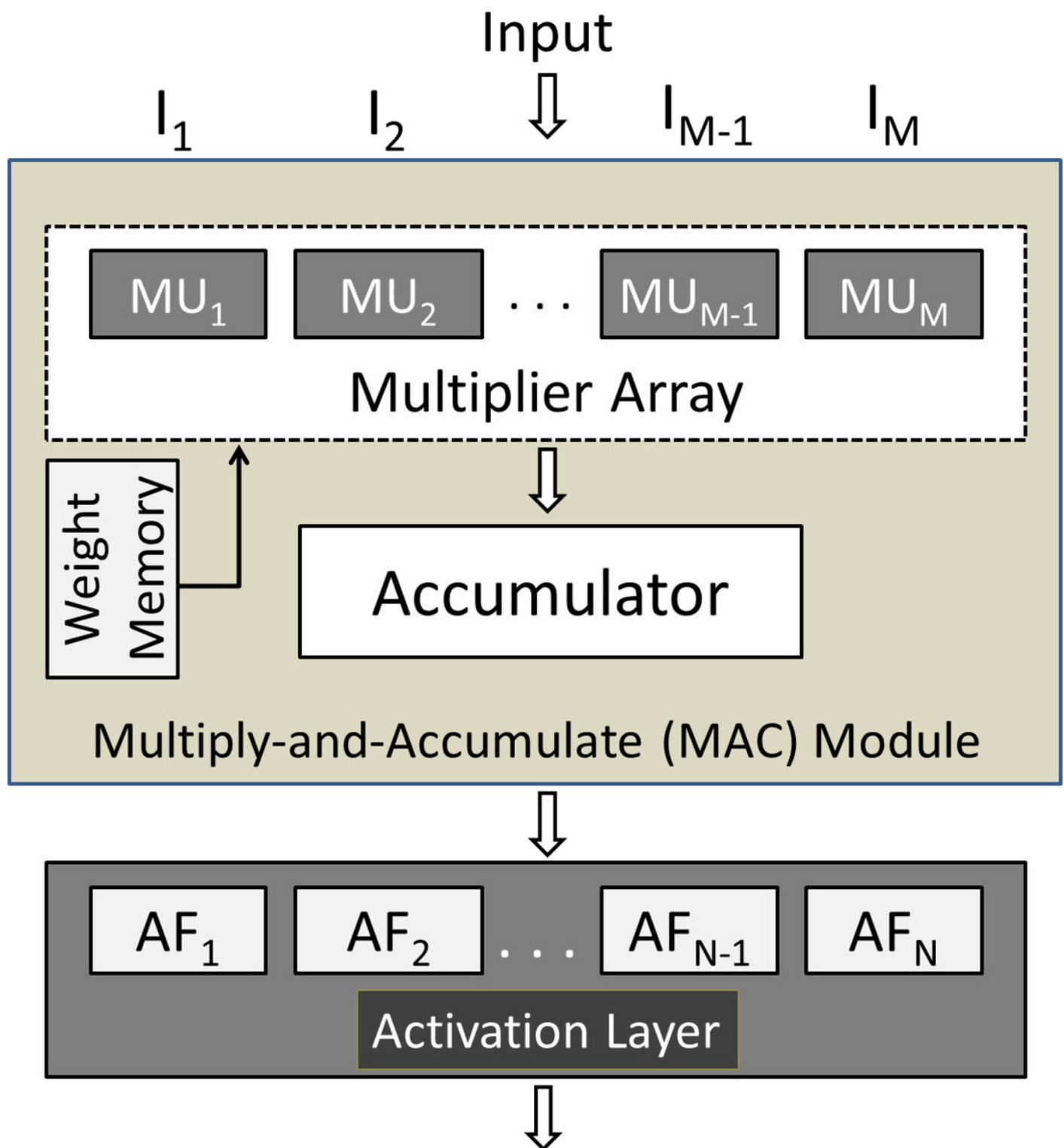



Figure 7

Figure6

Figure 8

Figure8

Figure 9

Figure4