REVIEW OF NEURONAL MULTIPLEXERS WITH BACK PROPAGATION ALGORITHM

Nwagbo, C. Lizzy and Ezekwe C. Genevra

[1] Nwafor Orizu College of Education, Nsugbe Anambra State, Nigeria[2]Electronic Development Institute, Awka, National Agency for Science and Engineering Infractructure, Presidency, Nigeria.

nwagbochioma2005@gmail.com and ezekwechinwe@eldi.org.ng

Abstract:

A multiplexer (switching device) is the most frequently used combinational circuit and it is an important building block in many digital systems which is commonly used in our day-to-day life in landline telephone networks and the Cable TV. Artificial Neural Network emulated multiplexers (neuronal multiplexers) are neural network model that mimics the way a human brain works by selecting the best conditions in the input signals of a physical multiplexers, train neurons to behave exactly like multiplexers. It is made up of a number of nodes that are organized in several layers that must follow the artificial neural network routing algorithm. The training of the neurons will base on the back propagation algorithm with several training parameters that will be presented to the neurons. The input layers of neurons feeds the input variables into the network through the hidden layers or processing elements which send the processed signal to the output layers. This can be implemented on a real television cable network with billions of nodes[neural netwok input nodes]. If this is achieved, we will experience a more reliable television cable networks on reliable secured protocols. This network will be more efficienct than the conventional network. There will not be signal lost on the network because of various signal routes of the same class of nodes. Network breakdown might be impossible with this because any breakdown on the network presents the next possible route on the same network[artificial intelligent learning]

Key Words: neuronal multiplexers, multiplexers, artificial neural networks, back propagation algorithm

I. Introduction

In order to explore the historical background of artificial neural network it is necessary to understand the origins and historical advancements of both logic devices and Artificial Neural Network. Gottfried Wilhelm Leibniz enhanced the binary number system, which he published in 1705, drawing inspiration from the binary system found in the ancient I Ching (Nylan, 2001)(Perkins, 2004). Leibniz recognized that employing the binary system allowed for the integration of both arithmetic and logical principles.

Between 1934 and 1936, Akira Nakashima, an engineer from NEC, along with Claude Shannon and Victor Shestakov, independently introduced the concept of switching circuit theory through a series of papers they demonstrated that two-valued Boolean algebra could effectively describe the functioning of switching circuits (Stanković & Astola, 2008). This discovery laid the groundwork for utilizing electrical switches to implement logical operations, forming the fundamental principle that underlies all electronic digital computers. The introduction of switching circuit theory

revolutionized digital circuit design, gaining widespread recognition among the electrical engineering community during and after World War II. Theoretical rigor replaced the previously prevalent ad hoc methods, solidifying the importance of this new approach (Stanković & Astola, 2008).

Bell Labs engineers Mohamed M. Atalla and Dawon Kahng showcased metal-oxide-semiconductor (MOS) devices, specifically PMOS and NMOS, in 1960(Lojek, 2007) Subsequently, these two types of MOS devices were merged and transformed into complementary MOS (CMOS) logic by Chih-Tang Sah and Frank Wanlass at Fairchild Semiconductor in 1963 (Wanlass & Sah, 2021).

II. HISTORICAL BACKGROUND OF ARTIFICIAL NEURAL NETWORK

The most basic type of feed forward neural network is a linear network, consisting of a single layer of output nodes. Inputs are directly connected to outputs through weighted connections, and the node's output is calculated by summing the products of the weights and inputs. The weights are adjusted to minimize the mean squared errors between the calculated outputs and the target values, a technique known as the method of least squares or linear regression. This method has been used for over two centuries in various applications, such as predicting planetary movement(Baker & Kuttler, 2014)(Schmidhuber, 2022).

In 1925, Wilhelm Lenz and Ernst Ising developed the Ising model, which can be considered a non-learning artificial recurrent neural network. Shun'ichi Amari later made this architecture adaptive in 1972, and it gained popularity through John Hopfield in 1982(Schmidhuber, 2022).

In 1943, Warren McCulloch and Walter Pitts(Matsumura et al., 2019) proposed a non-learning computational model for neural networks. In the late 1940s, D. O. Hebb introduced the learning hypothesis known as Hebbian learning, based on neural plasticity. In 1958, Frank Rosenblatt invented the perceptron, the first implemented artificial neural network (Haykin, 2008) (Werbos & Foundation, 2016), which was funded by the United States Office of Naval Research. However, research encountered obstacles when Minsky and Papert discovered the limitations of basic perceptrons in processing complex circuits and the lack of computational power for effective neural networks(Minsky & Papert, 2019).

Despite these challenges, methods for training multilayer perceptrons (MLPs) were already known at the time of Minsky and Papert's findings. The first deep learning MLP, known as the Group Method of Data Handling(Schmidhuber, 2015b), was published by Alexey Grigorevich Ivakhnenko and Valentin Lapa in 1965. The first deep learning MLP trained by stochastic gradient descent was introduced by Shun'ichi Amari in 1967(Schmidhuber, 2022). Experiments conducted with multilayer perceptrons demonstrated the ability to learn useful internal representations for classifying non-linearly separable patterns(Schmidhuber, 2022).

In 1982, Teuvo Kohonen introduced self-organizing maps (SOMs), which are neural networks inspired by neurophysiology (Chemic & Received, 2020). SOMs learn low-dimensional representations of high-dimensional data while preserving the data's topological structure. They employ competitive learning for training (Kohonen, Teuvo; Honkela, n.d.).

The architecture of convolutional neural networks (CNNs), comprising convolutional layers and downsampling layers, was introduced by Kunihiko Fukushima in 1980 and was called the

neocognitron. In 1969, Fukushima also introduced the rectified linear unit (ReLU) activation function (Schmidhuber, 2022). ReLU has since become the most popular activation function for CNNs and deep neural networks in general (Zoph & Le, 2018). CNNs have become essential for computer vision tasks.

The back propagation algorithm, an application of the Leibniz chain rule, was developed in the 1970s. Seppo Linnainmaa coined the term "back propagation" in 1970, although Frank Rosenblatt mentioned the concept earlier in 1962(Schmidhuber, 2022). In 1982, Paul Werbos applied back propagation to multilayer perceptrons (MLPs) in a standard manner(Schmidhuber, 2015a). In 1986, Rumelhart, Hinton, and Williams demonstrated that back propagation could learn meaningful internal representations of words when trained for language prediction (Schmidhuber, 2022).

In 1987, Alex Waibel introduced the time delay neural network (TDNN), which combined convolutions, weight sharing, and back propagation(John Wesley et al., 2020). In 1988, Wei Zhang et al. applied back propagation to a CNN for alphabet recognition, and in 1989, Yann LeCun et al. trained a CNN to recognize handwritten ZIP codes (VanHook, 2014).

In the 1990s, back propagation faced challenges with deep feed forward neural networks (FNNs) and recurrent neural networks (RNNs). Juergen Schmidhuber proposed a solution in 1992 with a hierarchy of RNNs pre-trained using self-supervised learning. This approach enabled learning at multiple self-organizing time scales. Schmidhuber also introduced the linear Transformer in 1992, an alternative to RNNs(Schmidhuber, 2022), which has become widely used in natural language processing and computer vision tasks.

The modern Transformer model was introduced in 2017 (Vaswani et al., 2017), combining self-attention mechanisms with softmax operators and projection matrices(Schmidhuber, 2022). Transformers have become the model of choice for many language processing tasks and are increasingly utilized in computer vision applications.

In 1991, Juergen Schmidhuber published adversarial neural networks, introducing the concept of artificial curiosity. The principle of adversarial learning was later employed in generative adversarial networks (GANs) (Mahdizadehaghdam et al., 2019), allowing the creation of realistic deep fakes.

In 1991, Sepp Hochreiter addressed the vanishing gradient problem and proposed recurrent residual connections, leading to the development of long short-term memory (LSTM) networks in 1997. LSTM networks are capable of learning complex tasks(Schmidhuber, 2015b) with long credit assignment paths. LSTM has been widely cited and utilized in various applications. Later advancements led to the creation of the Highway network and the Residual neural network (Srivastava et al., 2015)(He et al., 2016), both achieving significant success in image recognition.

III. HISTORICAL BACKGROUND OF ARTIFICIAL NEURAL NETWORK EMULATION OF LOGIC DEVICES

The historical progression of replicating logical functions through the use of Artificial Neural Networks (ANNs) spans several decades and is characterized by significant milestones and valuable contributions.

The origins of ANNs can be traced back to the pioneering work of Warren McCulloch and Walter Pitts in 1943, when they introduced the McCulloch-Pitts neuron model. This model aimed to mimic the behavior of biological neurons by employing logical operations. This foundational concept laid the groundwork for subsequent advancements in neural networks(Schmidhuber, 2022).

In 1957, Frank Rosenblatt introduced the perceptron, a basic neural model capable of binary classification tasks(Schmidhuber, 2022). However, it soon became apparent that single-layer perceptrons had limitations when it came to solving complex logical problems, sparking discussions about the potential of ANNs in handling logic functions.

The 1980s marked a significant breakthrough with the development of multilayer perceptrons (MLPs) and the introduction of the back propagation algorithm(Schmidhuber, 2022). These innovations empowered ANNs to tackle more intricate logic and pattern recognition tasks, extending their capabilities beyond simple binary classifications.

As research progressed, the 1990s witnessed the emergence of neural logic networks (NLNs), specifically designed to replicate complex logic operations and functions(Schmidhuber, 2022). NLNs represented a pivotal step toward harnessing ANNs for logic emulation.

In recent years, ANNs and NLNs have found practical applications across various domains. They have been leveraged for hardware acceleration, fault-tolerant computing, and the design of digital circuits(Schmidhuber, 2022). These applications underscore the growing significance of ANNs in addressing the challenges associated with traditional logic devices.

In conclusion, the historical development of ANNs for emulating logical devices reflects a persistent pursuit of more efficient, flexible, and adaptable solutions. From the early days of the McCulloch-Pitts neuron to the advanced neural logic networks in use today, this journey has significantly influenced the digital technology landscape and continues to drive innovation in various fields.

IV. RELATED LITERATURE

a. Logic Devices

A thorough coverage will be shed on the definition, operation, application, evolution and challenges of logic devices. It is necessary to explore this facet of knowledge as it is relevant in providing a comprehensive understanding on the concept of logic devices which in turn will provide a back bone in comprehending its further application in being emulated and the reason for such emulation

Definition and Types of Logic Devices

A logic gate is an idealized or physical device that performs a Boolean function, a logical operation performed on one or more binary inputs that produces a single binary output. In the real world, the primary way of building logic gates uses diodes or transistors acting as electronic switches. They can also be constructed using vacuum tubes, electromagnetic relays with relay logic, fluidic logic, pneumatic logic, optics, molecules, or even mechanical elements. By utilizing amplification; it becomes feasible to link logic gates sequentially, akin to the composition of Boolean functions. This permits the creation of a tangible representation encompassing the entirety of Boolean logic,

and subsequently, the entirety of algorithms and mathematical concepts explicable through Boolean logic.

Types of Logic Devices

The types of logic gate explored will be limited in scope to those that will be emulated in this project. They include:

i. **AND GATE:** An AND gate receives 'A' and 'B', where A and B represent bits and produce output denoted by A.B or A∧B. In the scenario of this logic gate, it produces a high output or an output of one (1) only if the two inputs passed is one (1), other cases produce a low output or an output of zero (0). In a mathematical context, A.B=1 only if A=1 and B=1 else A.B=0. This situation takes the context of a two input AND gate.

ii. **OR GATE:** An OR gate receives 'A' and 'B', where A and B represent bits and produce output denoted by A+B or A∨B. In the scenario of this logic gate, it produces a high output or an output of one(1) only if at least one of the inputs passed is one(1) other cases produces a low output or an output of zero(0). In a mathematical context, A+B=1 if either A=1 or B=1 else A+B = 0. This situation takes the context of a two input OR gate.

iii. **NOT GATE:** A NOT gate (or inverter) receives input A, where A is a bit; and produces output. Basically, it produces the opposite of the input passed through it. For example, if A=1 then the output produced will be zero (0) and if A = 0 then the output produced will be one (1).

iv. **NAND GATE:** A NOT-AND gate can be likened to an AND gate succeeded by a NOT gate. All the outputs of NAND gates turn high if any of the inputs are low. The symbol for this gate resembles an AND gate, but with a small circle placed at the output. This small circle denotes inversion.

v. **NOR GATE:** A NOT-OR gate can be compared to an OR gate followed by a NOT gate. All the outputs of NOR gates become low if any of the inputs are high. The representation of this gate involves an OR gate symbol with a small circle positioned at the output. This small circle signifies inversion.

vi. **EXCLUSIVE-OR GATE (XOR or EOR or EXOR GATE):**The 'Exclusive-OR' gate is a circuit that produces a high output when one of its two inputs is high, but not both. An encircled plus sign (+) is utilized to represent the Exclusive-OR operation.

vii. **EXCLUSIVE-NOR GATE (XNOR or ENOR or EXNOR GATE):**The 'Exclusive-NOR' gate circuit functions in contrast to the EOR gate. It produces a low output when one of its two inputs is high, but not both. The representation of this gate involves an EXOR gate symbol with a small circle positioned at the output. This small circle signifies inversion.

viii. **2-TO-1 MULTIPLEXER:** Multiplexing is the process of combining one or more signals and transmitting on a single channel. The device which is responsible for Multiplexing is known as Multiplexer. A 2-to-1 multiplexer consists of two inputs D0 and D1, one select input S and one output Y. Depending on the select signal, the output is connected to either of the inputs. Since there are two input signals, only two ways are possible to connect the inputs to the outputs, so one select is needed to do these operations. If the select line is low, then the output will be switched to D0 input, whereas if select line is high, then the output will be switched to D1 input.

Symbols and Truth Table

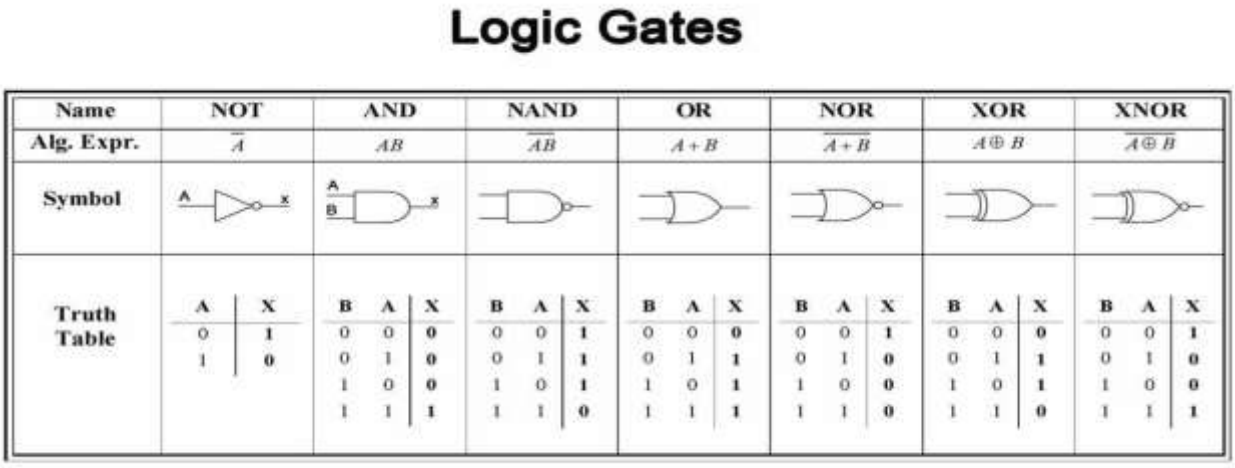

Figure 1: illustration of logic devices, their symbols and truth tables

Symbols and truth table multiplexers
Table 2.0 showing the truth table of 2-to-1 multiplexer

S	D0	D1	Y
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

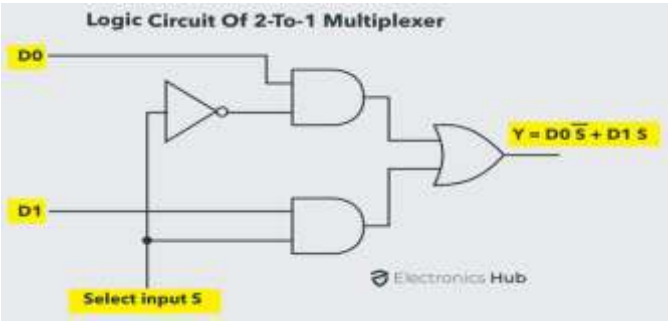

Figure 2: illustration of 2-to-1 multiplexer logic circuit

Applications of Logic Gates
The utilization of logic gates primarily depends on their truth table, which outlines their operational principles (Garg & Kaur, 2019).

AND GATE

AND gates find application in amalgamating numerous signals; the output is TRUE only when all input signals are TRUE. If any input signal is FALSE, the output becomes FALSE. AND gates are not as extensively employed as NAND gates, since NAND gates require fewer components and

can function as inverters. An instance of an AND gate application is in security alarms. For Example

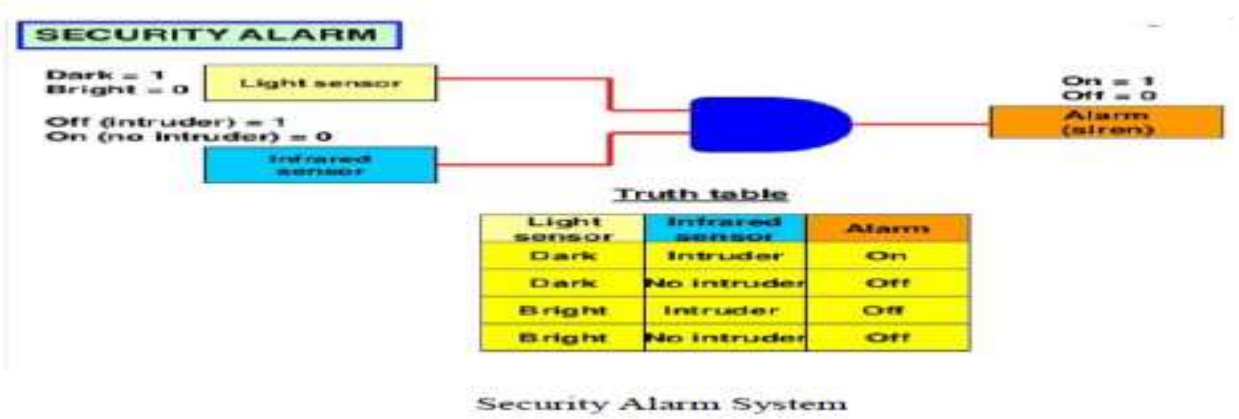

Figure 3: illustration of security alarm system using AND gate

OR GATE

Whenever the occurrence of any one or more than one event is needed to be detected or some actions are to be taken after their occurrence, in all those cases OR gates can be used. For Example

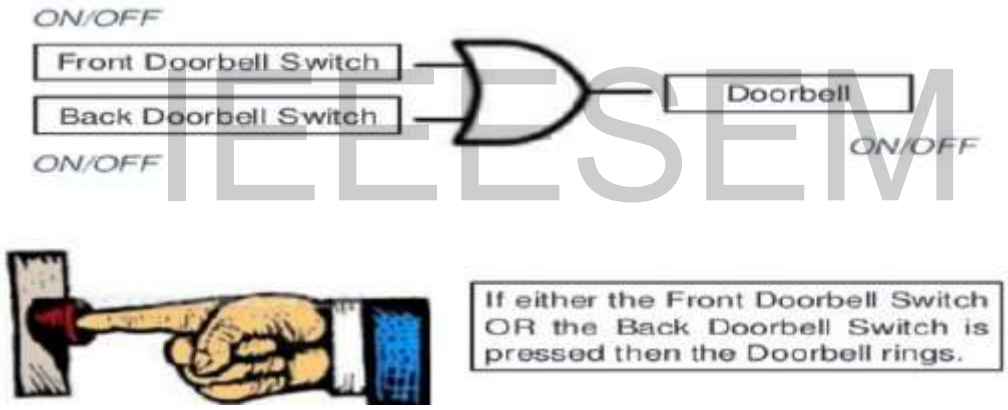

Figure 4: illustration of Doorbell system using OR GATE

NOT GATE

NOT gates, recognized as inverters, alter the provided input and display the opposite outcome. The benefit of utilizing these inverters is their minimal power consumption and simplified interfacing, distinguishing them from alternative logic gates. For Example

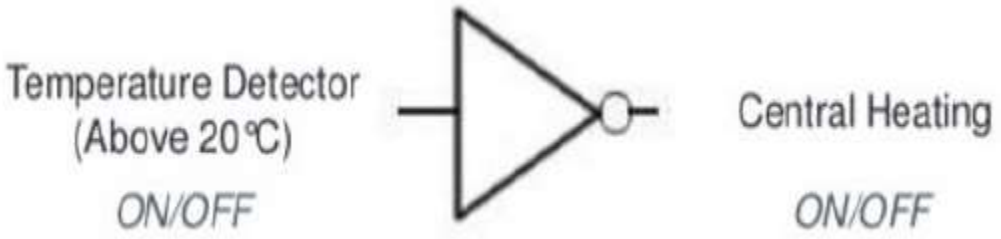

Figure 5: illustration of NOT gate based on heating concept

NAND GATE

It can be used for Device Failure Alarm System where if every fan is operational; the NAND gate receives an input of 111, leading to an output of 0. If any single fan ceases functioning, the output of the NAND gate shifts to 1, triggering the alarm.

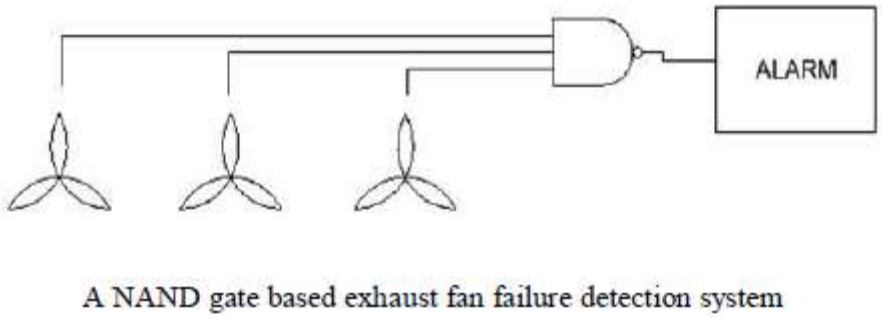

Figure 6: illustration of NAND gate based on exhaust fan failure detection system

NOR GATE

It can be used for Washing Machine Controller. A Washing Machine employs three sensors to monitor the status of its lid, the water level in the tub, and the weight of the laundry and water. If the machine's lid is open, the water level in the tub is below a specific threshold, or if the machine is overloaded (weight of water and clothes surpasses a certain limit), the respective sensor's output is marked as 1. This outcome results in a 0 at the NOR gate's output, which subsequently deactivates the machine.

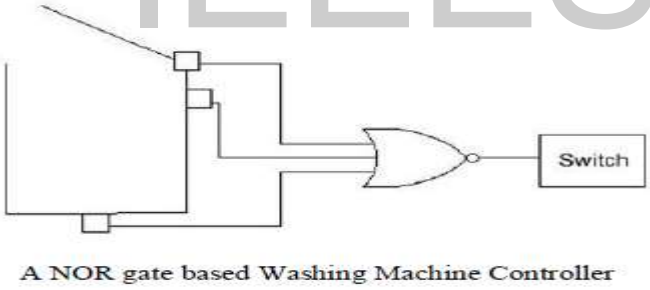

Figure 7: illustration of NOR gate based Washing Machine Controller

Multiplexers

In all types of digital system applications, multiplexers find its immense usage. Since these allow multiple inputs to be connected independently to a single output, multiplexers are found in variety of applications including:

Data Routing

Multiplexers are extensively used in data routing applications to route the data to a one particular destination from one of several sources. One of the applications includes the displaying of two multidigit BCD counters, one at a time. In such application, 74157 multiplexer ICs are used to select and display the content of either of two BCD counters using a set of decoder and LED displays.

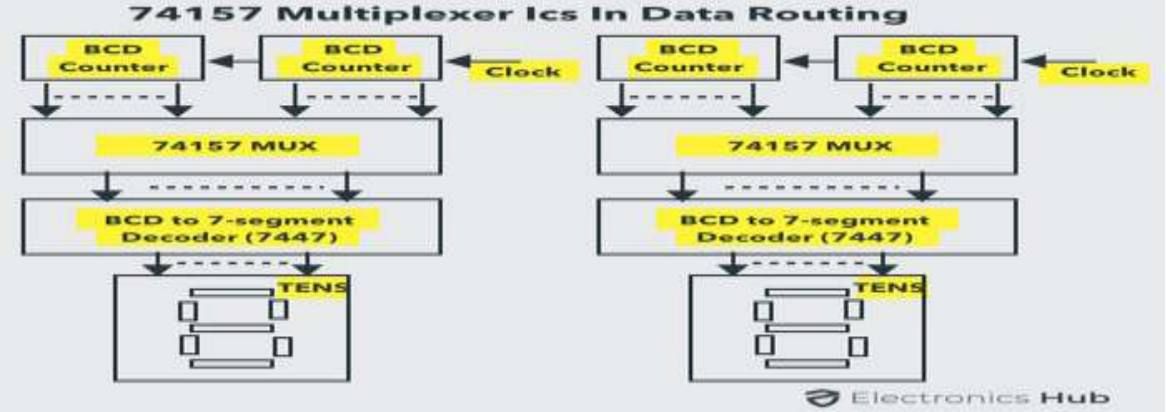

Figure 8: illustration of Multiplexers used in data routing applications

Challenges and limitations of conventional logic devices

Traditional logic devices, although integral to computing systems, present a range of issues that have prompted researchers to explore innovative alternatives.

i. **Power Consumption:** Traditional logic devices, particularly those reliant on power-intensive technologies like TTL, consume notable energy even during periods of inactivity due to static current flow. This results in escalated energy usage and heat generation, impacting both energy efficiency and heat management.
ii. **Speed Constraints:** As digital systems advance in complexity, signal propagation delays through conventional logic gates impose constraints on operational speed. This impedes the real-time processing requisites of contemporary applications.
iii. **Size and Integration:** Constructing intricate logic circuits through discrete components yields unwieldy circuitry, curtailing the potential for integration and miniaturization. This is of particular concern in scenarios where spatial limitations are a concern, such as portable devices.
iv. **Limited Flexibility:** Traditional logic gates possess fixed functions predetermined by their design. Adjusting their behavior to evolving requirements demands physical modifications, thereby curbing their adaptability.

Exploring Alternatives: Emulating Neural Networks

i. **Enhanced Power Efficiency:** In response, researchers are turning to neural network emulation as a means to address power consumption hurdles. Neural networks, with their parallel processing and adaptable structure, offer the prospect of reduced energy usage, especially when optimized for specific tasks.
ii. **Augmented Processing Speed:** By tailoring neural networks for specific tasks, researchers aim to potentially achieve swifter processing speeds compared to conventional logic gates. This adaptability proves particularly advantageous in scenarios necessitating real-time data processing.
iii. **Adaptive Learning Capability:** The inherent learning and adaptability of neural networks enable them to reconfigure according to shifting conditions. This dynamic quality mitigates the limitations of fixed-function logic gates.
iv. **Improved Integration and Size Efficiency:** The capacity to integrate neural networks within a single chip enhances integration capabilities, thus addressing size constraints. This has the potential to optimize spatial utilization in modern electronic systems.

The constraints posed by traditional logic devices, spanning power consumption, speed, and size challenges, have catalyzed the exploration of innovative avenues such as neural network emulation. These pathways offer the potential to surmount the limitations of conventional logic devices and open up novel prospects for energy-efficient and adaptive digital systems.

b. Artificial Neural Network

Artificial neural networks (ANNs), commonly referred to as neural networks (NNs) or neural nets, are computational structures that draw inspiration from the intricate networks of biological neurons found in animal brains.

The term "neural" is often used to refer to a particular category of statistical models that exhibit the following characteristics:

i. Contains sets of adjustable weights, which are numerical parameters adjusted by a learning algorithm.
ii. Is capable of approximating non – linear functions of their inputs

An ANN is built upon a cluster of interconnected units or nodes known as artificial neurons, which serve as a simplified representation of the neurons found in biological brains. Similar to synapses in the brain, each connection is capable of transmitting signals to other neurons. When an artificial neuron receives signals, it processes them and can transmit signals to the neurons it is connected to. The "signal" conveyed through a connection is a real number, and the output of each neuron is determined by a non-linear function applied to the sum of its inputs. These connections are referred to as edges. Neurons and edges typically possess weights that are adjusted during the learning process. These weights either strengthen or weaken the signal's intensity at a connection. Additionally, neurons may have a threshold, meaning a signal is only sent if the cumulative signal surpasses that particular threshold.

Usually, neurons are organized into layers, with each layer carrying out distinct transformations on the inputs it receives. The signals propagate through these layers, starting from the initial layer (known as the input layer) and progressing towards the final layer (known as the output layer), potentially passing through the layers multiple times before reaching the output layer.

A specific neural network model is determined by its topology, learning paradigm and learning algorithm.

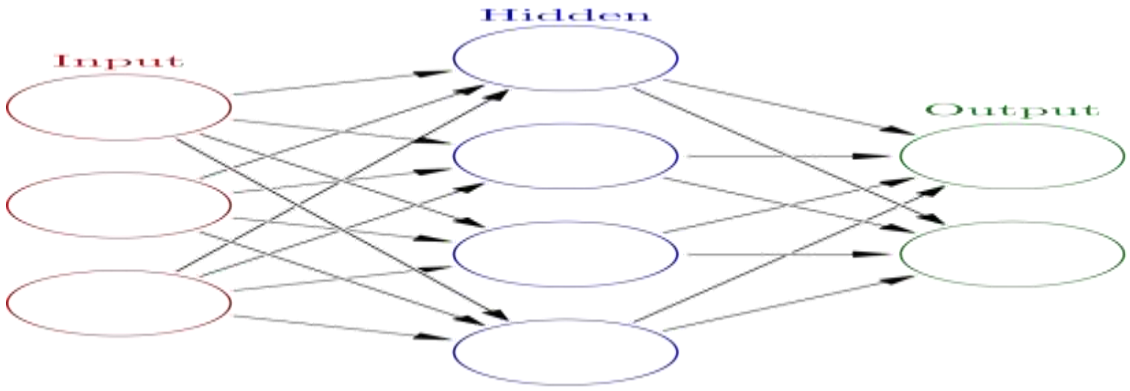

Figure 9: Neural Network model (one processing element region)

In Figure 9, each circular node symbolizes an artificial neuron, and the arrows indicate connections that transmit signals from the output of one artificial neuron to the input of another.

Artificial Neural Network can be deconstructed into models which will be fully explored. These models include:

i. The Biological model of Artificial Neural Network
ii. The Mathematical model
iii. The Artificial Neural Network model

Biological Model of Artificial Neural Network

Neurons serve as the elementary computing units in the nervous system. In the human brain, there are approximately 10^12 cells with 10^15 interconnections for information processing. Neurons are responsible for transmitting information and vary in their physical structure and function. Communication between distant neurons in the vertebrate nervous system is achieved through encoded pulse streams (Yellamraju et al., 2013).

Synapses, the junctions between neural cells, facilitate the transmission of impulses. Neurons are discrete cells and not connected to other parts of the body. Information flows from the axon to the dendrites through the cell body. Each neuron has around 10^3-10^4 synapses. Memories are formed by adjusting the connections between neurons, with neurotransmitters playing a vital role in synapses. Long-term memories are encoded within the synapses (Yellamraju et al., 2013).

The input to a neuron comes from the axon of the previous connecting neuron in the form of synaptic potentials. When an action potential reaches a presynaptic terminal, it triggers a series of events that lead to the release of neurotransmitters, resulting in ionic currents in the postsynaptic neuron's membrane. This current causes a change in the membrane voltage, known as the postsynaptic potential (PSP) (Yellamraju et al., 2013).

The dendrites of a neuron receive these synaptic potentials, which influence the action in the cell body. If multiple synaptic potentials are applied simultaneously and reach a sufficient threshold, their summation can generate an action potential or "spike." In other words, inputs from other neurons are summed, and the output is in the form of spikes, occurring only when a specific threshold is reached by a sufficient number of inputs (Yellamraju et al., 2013).

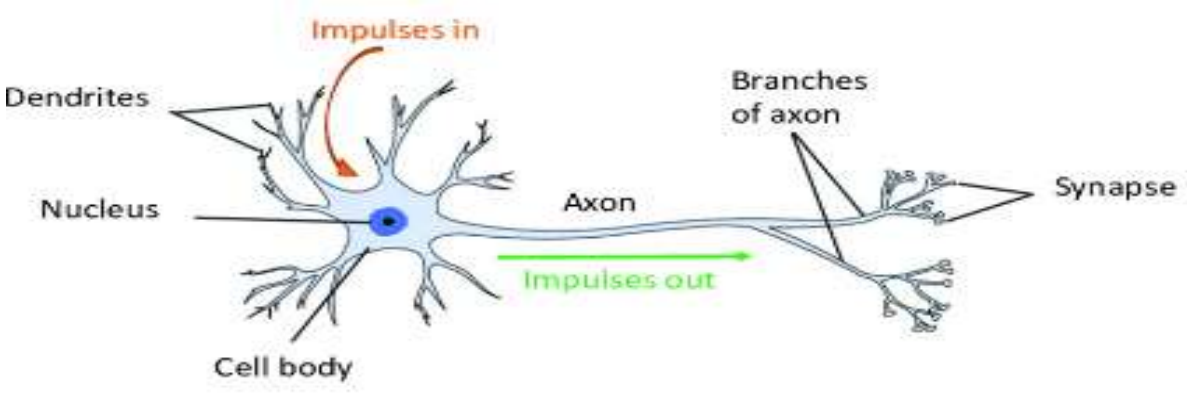

Figure 10: Diagram of biological neuron

Analogously, in artificial neural networks, there is a crude resemblance to biological neurons. The connections between nodes in artificial neurons can be likened to the axons and dendrites in biological neurons. The connection weights in artificial neurons represent synapses, while the threshold approximates the activity in the soma (cell body) of the receiving neuron. As a result, when a certain threshold is reached, the artificial neuron generates a new electrical signal that passes through the network (Mohammadhassani et al., 2013).

Figure 11 illustrates *n* biological neurons neurons with various signals of intensity *x* and synaptic strength *w* feeding into a neuron with a threshold of *b*, and the equivalent artificial neurons system (Mohammadhassani et al., 2013).

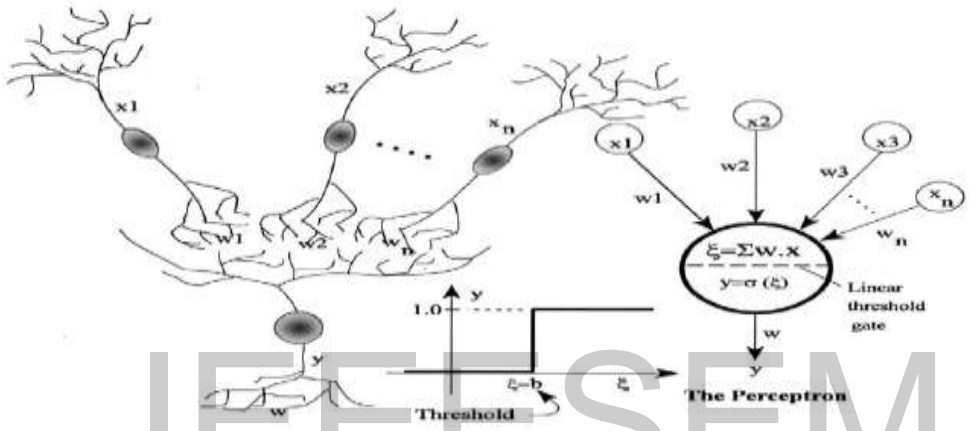

Figure 11: Signal interaction from *n* neurons and analogy to signal summing in an artificial neuron comprising the single layer perceptron (Mohammad hassani et al., 2013).

Comparison of biological to artificial neural network

Table 1: Comparison of biological to artificial neural network

Biological Neural Network	Artificial Neural Networks
Dendrites	Inputs
Cell Nucleus	Neurons
Synapse	Weights
Axon	Output

The Mathematical Model

When modeling an artificial functional model based on a biological neuron, there are three essential components that need to be considered. Firstly, the synapses of the biological neuron are represented as weights in the model. It is important to note that the synapse in the biological neuron refers to the interconnection within the neural network and determines the strength of the connection. In the artificial neuron, the weight is a numerical value that corresponds to the synapse.

Negative weights indicate inhibitory connections, while positive values indicate excitatory connections.

The next components of the model capture the actual activity of the neuron cell. All inputs are combined together and adjusted by the respective weights. This combination of inputs is referred to as a linear combination. Finally, an activation function is applied to control the magnitude of the output. For instance, the output may be constrained within an acceptable range, such as 0 to 1, or it could be between -1 and 1.

Mathematically, this process is described in the figure 2.10

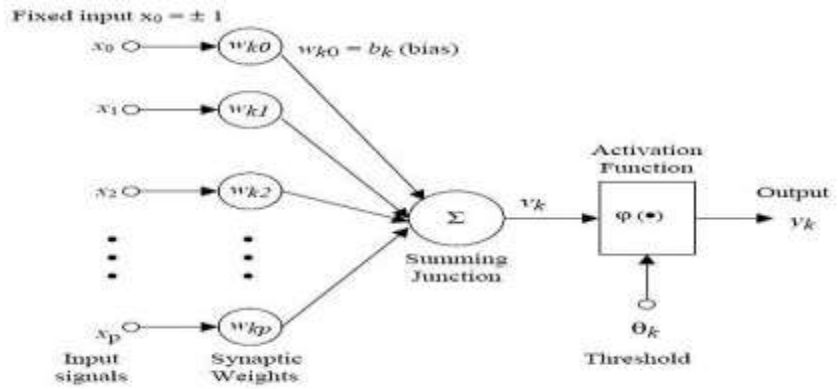

Figure 12: Diagram of mathematical model of ANN

From this model the interval activity of the neuron can be shown to be:

$$v_k = \sum_{j=1}^{p} w_{kj} x_j$$

The output of the neuron, y_k, would therefore be the outcome of some activation function on the value of v_k.

i. Activation functions

The activation function plays a crucial role as it acts as a function that compresses the output of a neuron within a specific range, typically between 0 and 1, or -1 and 1. Generally, there are three types of activation functions denoted as Φ (.). One of these types is the Threshold Function, which assigns a value of 0 if the total input is below a specified threshold value (v), and a value of 1 if the total input is equal to or greater than the threshold value.

$$\varphi(v)=\begin{cases}1 & \text{if } v \geq 0\\ 0 & \text{if } v < 0\end{cases}$$

Another type of activation function is the Piecewise-Linear function. This function, similar to the Threshold Function, can have values of 0 or 1. However, it can also have intermediate values within a specific range based on the amplification factor in a particular region of linear operation.

$$\varphi(v)=\begin{cases}1 & v \geq \frac{1}{2}\\ v & -\frac{1}{2} > v > \frac{1}{2}\\ 0 & v \leq -\frac{1}{2}\end{cases}$$

The third type of activation function is the sigmoid function. This function typically spans from 0 to 1, although it can also be adapted to the range of -1 to 1. One example of a sigmoid function is the hyperbolic tangent function.

$$\varphi(v)=\tanh\left(\frac{v}{2}\right)=\frac{1-\exp(-v)}{1+\exp(-v)}$$

ii. The network activation rate

During the process of supervised training in an artificial neural network, the activation function, also known as "alpha," plays a crucial role. This parameter is responsible for determining the initial network weights, which are referred to as the initial weights (wi1 to win). The final weights (wf1 to wfn) of the network are determined by the combination of the initial weights and the product of the network error vector and the chosen network input.

The final weights (w_{f1} to w_{fn}) = the initial weight (w_{i1} to w_{in}) + (the network error vector)(the chosen network input)(alpha).

The the network error vector is determinmined from the differene between the actual output and the network output.

The network error vector = the actual output - the network output(Ezekwe, n.d.).

c. The Artificial Neural Network Model

It is important to note that every Artificial Neural Network will primarily consist of:

i. The Input Layer
ii. The Processing element or Hidden Layer
iii. The Output Layer

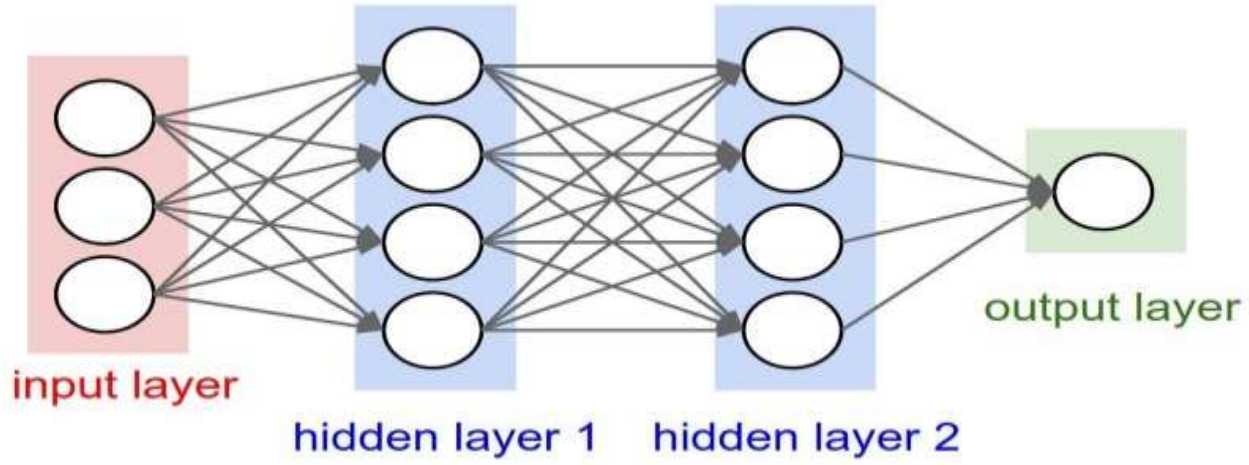

Figure 13: Artificial Neural Network model (two processing element region)

A neural network model consists of multiple neurons arranged in different layers: an input layer, one or more hidden layers, and an output layer. The input layer receives input variables and passes them to the network. The hidden layer acts as an intermediary between the input and output layers. The neurons in this layer are essentially hidden from view, and their arrangement and number are often treated as a black box by system operators. The primary function of the hidden layer is to process the input variables. This involves summing up the weighted inputs, checking if the sum meets a specific threshold, and applying a transformation function. The weights between the input neurons and hidden neurons determine when each unit in the hidden layer becomes active or remains inactive. By adjusting these weights, the hidden layer learns the relationship between inputs and outputs, similar to the way the human brain operates. The output layer serves a similar purpose to the hidden layer, with each neuron in this layer receiving inputs as in the hidden layer. The specific characteristics of a neural network model are defined by its topology (structure), learning paradigm, and learning algorithm.

Types of Artificial Neural Network

Neural networks can exist either in hardware form, where neurons are embodied by tangible components, or as software-based constructs represented by computer models. These networks encompass diverse configurations and employ a range of topologies and learning algorithms.

i. **Feed forward Network**: The initial and most basic form of neural network was the feed forward neural network. In this architecture, data flows exclusively from the input layer through possible hidden layers to the output layer without forming loops or cycles. Feed forward networks come in diverse variations, employing different unit types, including binary McCulloch–Pitts neurons. The most straightforward among these is the perceptron. In scenarios involving back propagation, continuous neurons, often activated with sigmoid functions, are commonly employed.

ii. **Deep belief network**: A deep belief network (DBN), comprising multiple hidden layers, is a generative model with a probabilistic nature. It can be seen as an amalgamation of basic learning components. This network can be employed for generative pre-training of a deep neural network (DNN) by using the acquired DBN weights as initial weights for the DNN.

iii. **Recurrent neural networks**: Recurrent neural networks (RNNs) not only transmit data in the forward direction but also in reverse, from subsequent processing phases to preceding ones. RNNs find utility as versatile processors of sequences in general.
iv. **Modular neural network**: Research in the field of biology has demonstrated that the human brain functions as an assembly of smaller networks. This understanding led to the emergence of the notion of modular neural networks, where multiple compact networks collaborate or contend to address challenges.
v. **Physical neural network**: A tangible neural network comprises materials with electrically modifiable resistance to mimic synthetic synapses. Instances encompass the ADALINE neural network based on memristors. An optical neural network is a real-world realization of an artificial neural network that employs optical components.
vi. **Dynamic neural networks**: Dynamic neural networks deal with nonlinear interactions among multiple variables and encompass time-varying behaviors, including learning, transient occurrences, and delays. Methods for deducing a system's functioning from observed data are categorized as system identification techniques.
vii. **Memory networks**: Memory networks integrate an extended memory capability. This memory can be both accessed and updated, primarily intended for predictive purposes. These models find application in question answering scenarios, where the prolonged memory operates as a dynamic knowledge repository, generating text-based responses.

Artificial Neural Network Training

Neural networks acquire knowledge through the analysis of examples, where each example consists of a known input and result. This process forms connections between the input and result, which are stored within the network itself. To train a neural network, it compares the output it generates (often a prediction) with the desired output and calculates the difference, known as the error. Using a learning rule and the error value, the network then adjusts its connections. By repeating this adjustment process, the neural network gradually improves its output to closely match the desired result. The training continues until certain criteria are met, at which point it can be concluded. This type of learning is referred to as supervised learning.

These systems acquire the ability to carry out tasks by examining examples, typically without the need for explicit programming of task-specific instructions. For instance, in the context of image recognition, they can learn to recognize images containing cats by analyzing labeled examples of "cat" or "no cat" and using that knowledge to identify cats in other images. Remarkably, they accomplish this without any prior knowledge about cats, such as their possession of fur, tails, whiskers, or cat-like faces. Instead, they autonomously generate distinguishing features based on the examples they process.

Learning

Learning is the process of enabling a network to enhance its performance in a given task by analyzing sample observations. It entails modifying the weights (and possibly thresholds) of the network to enhance the accuracy of its outputs. This adjustment is achieved by minimizing the errors observed during the learning process. Learning is considered finished when further examination of additional observations ceases to significantly decrease the error rate. However, even after learning, it is common for the error rate to remain nonzero. If the error rate remains unacceptably high post-learning, it is often necessary to redesign the network.

Supervised Learning

Supervised learning is a machine learning approach employed when dealing with data that is labeled, where each data point contains both features (covariates) and a corresponding label. The primary objective of supervised learning algorithms is to learn a function that can map input feature vectors to their respective labels (Nath & Levinson, 2014), using the available example input-output pairs. These algorithms deduce this function by examining a set of labeled training data, comprising various training examples. In supervised learning, each example consists of an input object (often represented as a vector) and an associated desired output value, also referred to as the supervisory signal. By analyzing the provided training data, a supervised learning algorithm generates an inferred function, which can subsequently be used to map new examples.

Unsupervised Learning

Unsupervised learning pertains to algorithms that acquire patterns and structures from unlabeled data. Unlike supervised learning, where models are trained to map input to target output (such as classifying images as "cat" or "fish"), unsupervised methods focus on learning condensed representations of the input data. These representations can be utilized for data exploration, analysis, or even generating new data. Within the spectrum of supervision, other levels include reinforcement learning, where the machine receives performance scores as guidance, and semi-supervised learning, where only a subset of the training data is labeled.

Use of Artificial Neural Network

ANN capabilities fall within the following broad categories:

i. Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling
ii. Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
iii. Data processing, including filtering, clustering, blind source separation and compression.
iv. Robotics, including directing manipulators and prostheses.

Application of Artificial Neural Network

Artificial neural networks (ANNs) have diverse applications across various disciplines due to their capability to replicate and model nonlinear processes. They include:

i. ANNs find application in areas such as system identification, control, and prediction, including vehicle control, trajectory prediction, process control, and natural resource management.
ii. They are also utilized in fields like quantum chemistry, general game playing, and pattern recognition, including radar systems, face identification, signal classification, 3D reconstruction, and object recognition.
iii. ANNs play a role in sensor data analysis, sequence recognition (such as gesture, speech, and text recognition), medical diagnosis, finance (e.g., financial long-run forecasts and artificial financial markets), data mining, visualization, machine translation, and social network and e-mail spam filtering.

iv. Furthermore, ANNs have demonstrated potential in diagnosing various types of cancers and distinguishing between highly invasive and less invasive cancer cell lines solely based on cell shape information.

Advantages of Artificial Neural Networks

Advantages of artificial neural networks include:

i. Parallel processing abilities mean the network can perform more than one job at a time.
ii. Information is stored on an entire network, not just a database.
iii. The ability to learn and model nonlinear, complex relationships helps model the real-life relationships between input and output.
iv. Fault tolerance means the corruption of one or more cells of the ANN will not stop the generation of output.
v. Gradual corruption means the network will slowly degrade over time, instead of a problem destroying the network instantly.

Disadvantages of Artificial Neural Networks

The disadvantages of ANNs include:

i. The lack of rules for determining the proper network structure means the appropriate artificial neural network architecture can only be found through trial and error and experience.
ii. The requirement of processors with parallel processing abilities makes neural networks hardware-dependent.
iii. The network works with numerical information, therefore all problems must be translated into numerical values before they can be presented to the ANN.
iv. The lack of explanation behind probing solutions is one of the biggest disadvantages in ANNs. The inability to explain the why or how behind the solution generates a lack of trust in the network.

V. RELATED WORKS ON NEURONAL MULTIPLEXERS WITH BACK PROPAGATION ALGORITHM

Not much focus has been placed on emulating logic devices with Artificial Neural Network but artificial intelligence has been making steady breakthrough in recent time thus it has been garnering more attention. Such case study will be explored in the paragraphs below

(Ferrer et al., 2004) work is about creating a special kind of computer chip called an FPGA(Field Programmable Gate Array) that can run a neural network. This neural network can be customized easily to work with different types of data or tasks. The chip's design allows you to change things like the number of layers and neurons in the network, how data is represented, and how calculations are done. It also lets you test different setups quickly to find the best balance between speed and accuracy. The chip uses a particular way of doing calculations, called fixed-point arithmetic, which helps make things faster. The researchers tried this chip on an older computer board and found that it worked really well. It was much faster than running the same neural network on regular software like MATLAB. This shows that using this chip can greatly speed up certain types of computations

(Mukkara & Ramanaiah, 2019)paper, the researchers came up with a special way to build a "neuronal half adder" using computer chips called VLSI (Very Large Scale Integrated circuit). They used a specific algorithm called "CSD" to make it work. Then, they tested this design on a chip called FPGA and compared the results with two other methods: "vedic multiplier" and a

regular method. They found that the CSD (Canonic Signed Digit) algorithm had the fastest response time (lowest delay) and used less power compared to the Vedic algorithm and the regular method. However, the drawback was that the CSD algorithm needed more space on the chip (area) compared to the other methods. So, it's a trade-off between speed and power efficiency on one side and chip space on the other.

VI. SUMMARY OF THE LITERATURE (IN A TABLE)

Table 2.2: Summary of the Literature

S/N	AUTHOR(S), DATE AND TITLE	RESEARCH OBJECTIVE	METHODOLOGIES	FINDINGS	INHERENT GAPS
1	(Garg & Kaur, 2019) A Review of Logic Gates and Its Applications	Identifying the types of logic gates and their applications	Systematic literature review	Outline of the types and application of simple logic gate	Only covered simple logic gates and not all application of the covered gates was described
2	(Yellamraju et al., 2013) Design of various logic gates in neural networks	Designing and implementing a biologically inspired neuron which will accept multiple synaptic inputs.	CMOS (Complementary Metal-Oxide-Semiconductor) technique	(I)The design and simulation of the pulsed output neuron and its simulations have been presented. (II)The design of the logical OR, AND & NOT gates have also been designed and presented.	The brain operates with very minimal voltage and frequency levels, but precisely replicating these parameters in S-Edit is difficult. Nevertheless, trying out various voltage settings in S-Edit has the potential to achieve maximum efficiency, even though exact outcomes might not be attainable.
3	(Mohammadhassani et al., 2013)	To familiarize the reader with ANN-based	Experimental data from eight high strength self-compacting	An historical summary of the evolution of the field of neurocomputing	Not stated in the research paper

	Application of artificial neural networks (ANNs) and linear regressions (LR) to predict the deflection of concrete deep beams	computing (neurocomputing)	concrete (HSSCC) deep- beams.	was presented along with a review of the basic issues pertaining to ANN-based computing and ANN de- sign.	
4	(Ferrer et al., 2004)	Creating a special kind of computer chip that can run a neural network.	Reprogramability of FPGAs to implement a whole family of multilayer perceptronneural networks modifying only a set of parametersin compilation time.	A flexible and scalable design was obtained for a whole family of multilayer perceptron neural networks.	Not stated in the research paper
5	(Mukkara & Ramanaiah, 2019)	VLSI implementation of neuronal half adder with CSD algorithm is proposed and implemented in FPGA.	Neuronal CSD Half adder is proposed and implemented on FPGA.	It is observed CSD half adder faster in performance with comparable low power consumption with that of other two multipliers. This work can be extended to implement any other combinational circuits.	Not stated in the research paper

VII. RESEARCH GAPS AND RESEARCH CONTRIBUTIONS

In the field of artificial neural network emulation of logic devices, certain research gaps have emerged. While considerable research has been conducted on simple logic devices, there remains a significant gap in exploring the emulation of more intricate and multifunctional logic devices using artificial neural networks. Additionally, the absence of standardized methodologies tailored for selecting neural network architectures, training parameters, and performance evaluation metrics specific to logic device emulation is notable

On the other hand, the research in this field contributes significantly to advancing the accuracy and efficiency of neural network emulation models. These contributions range from optimizing training strategies to enhance the speed and effectiveness of emulations, to proposing hybrid approaches that combine neural networks with other computational techniques for improved performance.

REFERENCES

Baker, R., & Kuttler, K. (2014). Linear algebra with applications. In *Linear Algebra with Applications*. https://doi.org/10.1142/9111

Chemic, B., & Received, F. R. G. (2020). Self-Organization of Orientation Sensitive Cells in the Striate Cortex. *Pattern Recognition by Self-Organizing Neural Networks*, *100*. https://doi.org/10.7551/mitpress/5271.003.0007

Ezekwe, C. G. (n.d.). *Artificial Neural Network Training*.

Ferrer, D., González, R., Fleitas, R., Acle, J. P., & Canetti, R. (2004). NeuroFPGA - Implementing artificial neural networks on programmable logic devices. *Proceedings - Design, Automation and Test in Europe Conference and Exhibition*, *May 2014*, 218–223. https://doi.org/10.1109/DATE.2004.1269233

Garg, B., & Kaur, S. (2019). a Review of Logic Gates and Its Applications. *Journal of Emerging Technologies and Innovative Research*, *6*(5), 124–129. www.jetir.org

Haykin, S. (2008). Neural Networks and Learning Machines. In *Pearson Prentice Hall New Jersey USA 936 pLinks* (Vol. 3). https://doi.org/978-0131471399

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90

John Wesley, R., Nayeemulla Khan, A., & Shahina, A. (2020). Phoneme classification in reconstructed phase space with convolutional neural networks. In *Pattern Recognition Letters* (Vol. 135, pp. 299–306). https://doi.org/10.1016/j.patrec.2020.05.002

Kohonen, Teuvo; Honkela, T. (n.d.). *Kohonen network - Scholarpedia*. http://www.scholarpedia.org/article/Kohonen_network

Lojek, B. (2007). History of semiconductor engineering. *History of Semiconductor Engineering*, 1–387. https://doi.org/10.1007/978-3-540-34258-8

Mahdizadehaghdam, S., Panahi, A., & Krim, H. (2019). Sparse generative adversarial network. *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, 3063–3071. https://doi.org/10.1109/ICCVW.2019.00369

Matsumura, R., Harada, K., Domae, Y., & Wan, W. (2019). Learning based industrial bin-picking trained with approximate physics simulator. *Advances in Intelligent Systems and Computing*, *867*, 786–798. https://doi.org/10.1007/978-3-030-01370-7_61

Minsky, M., & Papert, S. A. (2019). Perceptrons. In *Perceptrons*. https://doi.org/10.7551/mitpress/11301.001.0001

Mohammadhassani, M., Nezamabadi-Pour, H., Jumaat, M. Z., Jameel, M., & Arumugam, A. M. S. (2013). Application of artificial neural networks (ANNs) and linear regressions (LR) to predict the deflection of concrete deep beams. *Computers and Concrete*, *11*(3), 237–252. https://doi.org/10.12989/cac.2013.11.3.237

Mukkara, L. kiran, & Ramanaiah, K. V. (2019). Neuronal Logic gates Realization using CSD algorithm. *International Journal of Reconfigurable and Embedded Systems (IJRES)*, *8*(2), 145. https://doi.org/10.11591/ijres.v8.i2.pp145-150

Nath, V., & Levinson, S. E. (2014). Machine learning. In *SpringerBriefs in Computer Science* (Vol. 0, Issue 9783319056050). https://doi.org/10.1007/978-3-319-05606-7_6

Nylan, M. (2001). *The Five "Confucian" Classics*. Yale University Press.

Perkins, F. (2004). Leibniz and China: A commerce of light. In *Leibniz and China: A Commerce of Light*. Cambridge

University Press. https://doi.org/10.1017/CBO9780511519994

Schmidhuber, J. (2015a). Deep Learning. *Scholarpedia*, *10*(11), 32832. https://doi.org/10.4249/scholarpedia.32832

Schmidhuber, J. (2015b). Deep Learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Schmidhuber, J. (2022). *Annotated History of Modern AI and Deep Learning*. *22*, 1–75. http://arxiv.org/abs/2212.11279

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). *Highway Networks*. http://arxiv.org/abs/1505.00387

Stanković, R. S. ;, & Astola, J. T. (2008). *Reprints from the Early Days of Information Sciences: TICSP Series On the Contributions of Akira Nakashima to Switching Theory*.

VanHook, A. M. (2014). Metabolism OutFOXes circadian rhythm. *Science Signaling*, *7*(329), 541–551. https://doi.org/10.1126/scisignal.2005580

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Wanlass, F. M., & Sah, C. T. (2021). *1963: Complementary MOS Circuit Configuration is Invented | The Silicon Engine | Computer History Museum*. Computer History Museum. https://www.computerhistory.org/siliconengine/complementary-mos-circuit-configuration-is-invented/

Werbos, P., & Foundation, N. S. (2016). *Beyond regression : new tools for prediction and analysis in the behavioral sciences / C I) NEW TOOLS FOR PREDICTION ANO ANALYSIS IN THE BEHAVIORAL SCIENCES BEYONn REGRESSION : A thesis presented Paul John Werbos for the degree of by In partial fulfillme*. *January 1974*.

Yellamraju, S., Kumari, S., Girolkar, S., Chourasia, S., & Tete, A. D. (2013). Design of various logic gates in neural networks. *2013 Annual IEEE India Conference, INDICON 2013*, 13–17. https://doi.org/10.1109/INDCON.2013.6725879

Zoph, B., & Le, Q. V. (2018). Searching for activation functions. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 1–13.