**Republic of Iraq**

**Ministry of Higher Education**

**and Scientific Research**

**University of Babylon**

**College of Engineering**

**Department of Electrical Engineering**


# Design and Simulation of Hybrid Neural Network System

Submitted to the Council of the college of Engineering, University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master of Science (M.Sc.) in Electrical Engineering / Industrial Electronics.

**By**

**Zainab Faris Talib Hashem**

**Supervised by**

**Prof. Dr. Ibrahim Abdullah Murdas**

**2023 A.D** **1444 A.H**

بِسْمِ اللَّـهِ الرَّحْمَـٰنِ الرَّحِيمِ

﴿قَالَ الَّذِي عِنْدَهُ عِلْمٌ مِنَ الْكِتَابِ أَنَا آتِيكَ بِهِ قَبْلَ أَنْ يَرْتَدَّ إِلَيْكَ طَرْفُكَ فَلَمَّا رَآهُ مُسْتَقِرًّا عِنْدَهُ قَالَ هَذَا مِنْ فَضْلِ رَبِّي لِيَبْلُوَنِي أَأَشْكُرُ أَمْ أَكْفُرُ وَمَنْ شَكَرَ فَإِنَّمَا يَشْكُرُ لِنَفْسِهِ وَمَنْ كَفَرَ فَإِنَّ رَبِّي غَنِيٌّ كَرِيمٌ﴾

صدق الله العلي العظيم

النمل (40)

Copyright © 2023. Without written permission from the author or the department of electrical engineering, faculty of engineering, university of Babylon, no portion of this thesis may be duplicated in any form, electronic or mechanical, including photocopy, recording, scanning, or any information.

# Examining certificate from the committee

We confirm that we have read and understood the thesis named ' **Design and Simulation of Hybrid Neural Network System'** and as an examining committee, examined the student **Zainab Faris Talib Hashem** in its substance, and that it, in our judgment, fits the requirements for a master's thesis engineering/electrical engineering / Industrial Electronic.

| | | |
|---|---|---|
| Signature: | Signature: | Signature: |
| Name: Prof. | Name: Prof. | Name: Asst. Prof. |
| Dr. Qais Kareem Omran | Dr. Musa H. Wali | Dr. Mustafa Rashid Ismael |
| (chirman) | (member) | (member) |
| Date: / /2023 | Date: / /2023 | Date: / /2023 |

| | |
|---|---|
| Signature: | Signature: |
| Name: Prof. | Name: Prof. |
| Dr. Ibrahim Abdullah Murdas | Dr. Qais Kareem Omran |
| (supervisor) | (Head of Electrical Department) |
| Date: / /2023 | Date: / /2023 |

Signature:

Name: Prof.

Dr. Hatem Hadi Obeid

(Dean of College of Engineering)

Date: / /2023

# Supervisor’s certification

I certify that this thesis entitled '**Design and Simulation of Hybrid Neural Network System**' was prepared by **Zainab Faris Talib Hashem** under my supervision as part of the prerequisites for a master's degree in engineering, department of electrical engineering, college of engineering, university of Babylon /electrical engineering /Industrial Electronic.

**Supervisor**

**Signature**:

Name**: Prof. Dr. Ibrahim Abdullah Murdas**

**Date: / /2023**

In light of the foregoing proposal, I am submitting this thesis to the Examination Committee for review.

Head of Electrical Department

Signature:

Name: Prof .Dr. Qais Kareem Omran

**Date: / /2023**

# Dedications

To the one who guides me to the way of God and God guided me guidance on his hands to my Shafi'i on the day of deen to the beloved Muhammad.

To my eternal love to the only one who will give me eternity if I die for him to the one who honored me by its Babylonian Origin to Great Iraq.

To those who did not leave me in my awakening and sleep to those who have great sorrow in my chest, and fire in my heart that does not calm down to those who spread hope despite pain and oppression to those who are the most generous of us to the martyrs of Iraq.

# Acknowledgement

First, I thank Allah (God) for His unlimited blessings that continue to flow into my life, and who gave me the power and determination to conclude my thesis successfully.

I would like to thank the guarding angel, the pure affection, one woman without her I would never be what I am, who provides me with love, strength and courage, the person to whom I am still indebted, the dearest person, (My Mother).

I would like to thank to the great man who offer his life for us to get to where we are now (my father).

I would like to thank the person who lit up my life. He was the main reason for completing this study. He bore the pressures and helped me to face the difficulties and gave me the factors of success (my beloved husband).

I would like to thank Prof. Dr. Ibrahim Abdullah Murdas, who award me the chance to do this work under his direction and guidance. I preferred to thank him for his support and advice throughout my thesis with infinite patience. His guidance and dedication gave me good experience throughout preparing this thesis.

I would like to thank my colleagues, brothers, sisters, and all the kind, helpful and lovely people. Who helped me directly or indirectly to complete this work and apologize to them for not being able to mention them by name here, but they are in my heart. May Allah reward them with the best reward.

# Abstract

Although Artificial Neural Networks (ANNs) are making great advances in large-scale and diverse fields of technology, the artificial neural network parameter training process requires a significant amount of time and energy. It is necessary to use new neural networks that differ from artificial neural networks known as Optical Neural Networks (ONNs). The ONNs can significantly improve the performance of the required tasks by taking advantage of their high characteristics such as high speed, high bandwidth, high balanced processing, low latency, and low power consumption. This study proposes to train an optical neural network through non-linear units by a Saturable Absorption (SA) (i.e. the SA as works a transducer that converts the electrical energy into optical energy) to perform image classification tasks. The neural networks are trained and tested with the help of (KMNIST, EMNIST, and MNIST) datasets, all datasets containing ($28 \times 28$) pixel grey scale images. A Convolution Neural Network (CNN) typically by processing an image through a series of layers to extract features. The output of the CNN is then fed into ONN, which further processes the features and makes a final classification. The mean absolute error is used to determine the error between actual and expected output. The error is then back-propagated across the network to update the weights and bias using the back-propagation algorithm. The training process occurs by repeating a process forward and backward in a direction that minimizes the loss function to the minimum value. After completing the training process and extracting the results, the test set was used to evaluate the performance of the model. The results of the system included a rise in the accuracy factor and a fall in the loss factor, so the proposed system is ideal for training the ONNs and performing image classification tasks for

the datasets. Finally, the performance of ONNs was compared with that of benchmark ANNs that employ ReLU activation functions. The results show that the accuracy of ANNs is almost close to the accuracy of ONNs. The value of the loss ONNs is much better than the value of the loss of ANNs, as well as for the processing speed in the implementation of ONNs is faster than the speed of the implementation of ANNs and the reason is due to the distinctive characteristics of ONNs compared to ANNs.

# Table of Contents


| Seq. | Contents | Page |
|---|---|---|
| **Chapter One: Introduction** | | |
| 1.1 | Introduction | 2 |
| 1.2 | Artificial Neural Network | 3 |
| 1.3 | Optical Neural Network | 6 |
| 1.3.1 | Hybrid Optic-electronic Neural Network | 7 |
| 1.3.2 | All-Optical Neural Network | 8 |
| 1.4 | Literature Review | 10 |
| 1.4.1 | Literature Review of Artificial Neural Networks | 10 |
| 1.4.2 | Literature Review of Optical Neural Networks | 13 |
| 1.5 | Problem Statement | 18 |
| 1.5 | Aims of the Study | 19 |
| 1.6 | Outline | 19 |
| **Chapter Two: Theory** | | |
| 2.1 | Introduction | 21 |
| 2.2 | Artificial Neural Network | 21 |
| 2.2.1 | Architecture of Artificial Neural Network | 21 |
| 2.2.2 | Training of Artificial Neural Network | 23 |
| 2.2.2.1 | Supervised Training | 23 |
| 2.2.2.2 | Unsupervised Training | 24 |
| 2.2.3 | Back-propagation Training Algorithm | 25 |
| 2.3 | Deep Learning | 27 |
| 2.4 | Dataset | 29 |

| | | |
|---|---|---|
| 2.4.1 | Training Set | 30 |
| 2.4.2 | Validation Set | 30 |
| 2.4.3 | Testing Set | 30 |
| 2.5 | Types of Dataset | 30 |
| 2.5.1 | Modified National Institute of Standards and Technology Dataset | 31 |
| 2.5.2 | Kuzushiji-MNIST Dataset | 31 |
| 2.5.3 | Extended MNIST Dataset | 32 |
| 2.6 | Mini-Batch | 33 |
| 2.7 | Convolution Neural Network | 33 |
| 2.7.1 | Basic Parts of Convolution Neural Network Architecture | 34 |
| 2.7.2 | Layer of Convolution Neural Network | 35 |
| 2.7.2.1 | Convolution Layer | 35 |
| 2.7.2.2 | Activation Layer (Non-Linear Layer) | 38 |
| 2.7.2.3 | Pooling Layer | 38 |
| 2.7.2.4 | Flattening Layer | 39 |
| 2.7.2.5 | Fully Connected Layer | 40 |
| 2.8 | Optical Neural Network | 41 |
| 2.8.1 | Spatial Light Modulator | 41 |
| 2.8.2 | General All-Optical Neural Network Structure | 43 |
| 2.8.3 | Linear Optical Operation | 44 |
| 2.8.4 | Activation Function | 46 |
| 2.8.4.1 | Sigmoid | 46 |
| 2.8.4.2 | Rectified Linear Unit | 47 |
| 2.8.4.3 | Hyperbolic Tangent | 47 |

| | | |
|---|---|---|
| 2.8.4.4 | Softmax Activation Function | 47 |
| 2.8.5 | Loss Function | 48 |
| 2.8.5.1 | Mean Square Error | 48 |
| 2.8.5.2 | Mean Absolute Error | 49 |
| 2.8.5.3 | Categorical Cross Entropy | 50 |
| 2.8.6 | Implementing Optical Back-propagation | 50 |
| 2.8.7 | Optimization Function | 57 |
| 2.8.8 | Type of Optimization Function | 58 |
| 2.8.8.1 | Stochastic Gradient Descent | 59 |
| 2.8.8.2 | Adam | 59 |
| 2.8.9 | Learning Rate | 61 |
| 2.8.10 | Evaluation Measures | 61 |
| 2.8.10.1 | Accuracy | 62 |
| 2.8.10.2 | Loss | 62 |
| **Chapter Three: The Proposed System** | | |
| 3.1 | Introduction | 64 |
| 3.2 | System Requirement | 64 |
| 3.3 | Proposed Methodology | 66 |
| 3.3.1 | Image Dataset | 67 |
| 3.3.2 | Reading Mini-Batch | 67 |
| 3.3.3 | Network Architecture | 67 |
| 3.3.3.1 | Convolution Neural Network/ Artificial Neural Network | 68 |
| A | First Convolution Layer + ReLU | 68 |
| B | First Pooling Layer | 69 |
| C | Second Convolution Layer + ReLU | 70 |

| | | |
|---|---|---|
| D | Second Pooling Layer | 70 |
| E | Flattening Layer | 71 |
| F | Fully Connected Layer | 71 |
| 3.3.3.2 | Convolution Neural Network/ Optical Neural Network | 73 |
| A | First Convolution Layer + SA | 73 |
| B | First Pooling Layer | 74 |
| C | Second Convolution Layer + SA | 75 |
| D | Second Pooling Layer | 75 |
| E | Flattening Layer | 76 |
| F | Fully Connected Layer | 76 |
| 3.3.4 | Training Network | 77 |
| **Chapter Four: Results and Discussion** | | |
| 4.1 | Introduction | 80 |
| 4.2 | Results and Discussion of Artificial Neural Networks | 80 |
| 4.3 | Results and Discussion of Optical Neural Networks | 83 |
| 4.4 | Comparison between Optical Neural Networks and Artificial Neural Networks | 93 |
| 4.5 | Comparing the Results with Related Work | 94 |
| **Chapter Five: Conclusions and Future Works** | | |
| 5.1 | Conclusions | 98 |
| 5.2 | Future Works | 99 |
| **References** | | 100 |

# List of Figures


| Figure | Figure Title | Page |
|---|---|---|
| 1.1 | The working principle of artificial neural networks. (a) A schematic drawing of biological neurons. (b) The structure of feed forward artificial neural networks. | 4 |
| 1.2 | The classification of ONN. | 7 |
| 1.3 | A structure of $D^2NN$. (a) The overall structure of $D^2NN$. (b) The usage of $D^2NN$ for image classifier. (c) The usage of $D^2NN$ for imaging. (d) The difference between diffractive ONN and electronic NN. | 9 |
| 1.4 | Development of ANN and ONN. | 18 |
| 2.1 | The block diagram of supervised training. | 24 |
| 2.2 | The block diagram of unsupervised training. | 25 |
| 2.3 | Example of the MNIST dataset. | 31 |
| 2.4 | Example of the KMNIST dataset. | 32 |
| 2.5 | Example of the EMNIST dataset. | 33 |
| 2.6 | The input image and kernel-matrix. | 36 |
| 2.7 | The convolution process. | 37 |
| 2.8 | The Max pooling and Average pooling with 4x4 cluster size. | 39 |
| 2.9 | Flattening process. | 40 |
| 2.10 | General perception for Fully Connected Layer. | 41 |
| 2.11 | Generally, AONN. (a) A usual two layer neural | 43 |

| | | |
|---|---|---|
| | networks. (b) Diagram of investigational realization of an optically neuron involving non-linear and linear processes. | |
| 2.12 | Optical setup, linear optical operation, characterization. Single-mode fiber (SMF), flip mirror (FM), mirror (M), optical lens (L1-L2), and camera (C1-C2). | 45 |
| 2.13 | ONN with all-optical forward and backward-propagation. (a) A single ONN and red color refers to forward and orange dashed refer to backward-propagation. (b) Error computation carried done optically or digitally at the output layer. | 52 |
| 2.14 | Response of a saturable absorber. (a) The transmission derivative of a SA unit with optical depths of 1 on the left and 30 on the right. In panel (b), these also approximate the derivatives with and without rescaling, which are the real probe transmissions from equation (20). | 54 |
| 3.1 | Method of import libraries. | 65 |
| 3.2 | Process of the proposed system or methodology. | 66 |
| 3.3 | Architecture of CNN/ANN. | 68 |
| 3.4 | Architecture of CNN/ONN. | 73 |
| 4.1 | The accuracy and loss of the KMNIST dataset in ANN. | 82 |
| 4.2 | The accuracy and loss of the MNIST dataset in ANN. | 82 |

| | | |
|---|---|---|
| 4.3 | The accuracy and loss of the KMNIST dataset in ONN (learning rate $1 \times 10^{-3}$). | 84 |
| 4.4 | The accuracy and loss of the KMNIST dataset in ONN (learning rate $1\times 10^{-4}$). | 86 |
| 4.5 | The accuracy and loss of the KMNIST dataset in ONN (learning rate $1\times 10^{-5}$). | 87 |
| 4.6 | The accuracy and loss of the KMNIST dataset in ONN ($\alpha_0 = 2$). | 89 |
| 4.7 | The accuracy and loss of the KMNIST dataset in ONN ($\alpha_0 = 31$). | 90 |
| 4.8 | The accuracy and loss of the EMNIST dataset in ONN. | 92 |
| 4.9 | The accuracy and loss of the MNIST dataset in ONN. | 93 |

# List of Tables


| Table | Table Title | Page |
|---|---|---|
| 4.1 | Results of ANN. | 81 |
| 4.2 | Results of the KMNIST dataset in ONN (learning rate $1 \times 10^{-3}$). | 84 |
| 4.3 | Results of the KMNIST dataset in ONN ( learning rate $1 \times 10^{-4}$). | 85 |
| 4.4 | Results of the KMNIST dataset in ONN (learning rate $1 \times 10^{-5}$). | 87 |
| 4.5 | Results of the KMNIST dataset in ONN ($\alpha_0 = 2$). | 88 |
| 4.6 | Results of the KMNIST dataset in ONN ($\alpha_0 = 31$). | 90 |
| 4.7 | Results of ONN. | 91 |
| 4.8 | A comparison between ONNs and ANNs. | 94 |
| 4.9 | Results of the suggested method and the paper [4]. | 95 |

# List of Abbreviations

| Abbreviation | Description |
|---|---|
| AE | Acoustic Emission |
| ANNs | Artificial Neural Networks |
| AONN | All-Optical Neural Network |
| BCNN | Binary Coding Neural Network |
| BNNs | Biological Neural Networks |
| BP-ANN | Back-Propagation Artificial Neural Network |
| CCE | Categorical Cross Entropy |
| CNN | Convolution Neural Network |
| CPUs | Central Processing Units |
| CS-ANN | Cuckoo Search-Artificial Neural Network |
| $D^2NN$ | Diffractive Deep Neural Network |
| EIT | Electromagnetically Induced Transparency |
| EMNIST | Extended Modified National Institute of Standards and Technology |
| FFNN | Feed-Forward Neural Network |
| FM | Flip Mirror |
| FMNN | Forward Multi-layer Neural Network |
| FPGA | Field programmable Gate Array |
| FPPGA | Field Programmable Photonic Gate Array |
| GA | Genetic Algorithm |
| GPUs | Graphical Processing Units |
| HCM | Hierarchical Clustering Method |
| HHT | Hilbert-Huang Transform |

| | |
|---|---|
| HONN | Hybrid Optic-electronic Neural Network |
| IER | Inland Environmental Resources |
| KMNIST | Kuzushiji Modified National Institute of Standards and Technology |
| LSTM | Long Short Term Memory |
| MAE | Mean Absolute Error |
| MNF | Minimum Noise Fraction |
| MNIST | Modified National Institute of Standards and Technology |
| MSE | Mean Square Error |
| NARMA | Non-linear Autoregressive Moving Average |
| $N - D^2NN$ | Non-linear Diffraction Deep Neural Network |
| ONC | Optical Neural Chip |
| ONNs | Optical Neural Networks |
| Opt-Conv | Optical Convolutional |
| PReLU | Parametric Rectifier Linear Unit |
| ReLU | Rectifier Linear Unit |
| ROI | Region Of Interest |
| RReLU | Randomized Rectifier Linear Unit |
| SA | Saturable Absorption |
| SGD | Stochastic Gradient Descent |
| SLM | Spatial Light Modulator |
| SMF | Single Mode Fiber |
| Tanh | Hyperbolic Tangent |
| VLSI | Very Large Scale Integration |

# CHAPTER ONE

# Introduction

# CHAPTER ONE

# Introduction

## 1.1 Introduction

Recently, technological advancements and the development of novel mathematical optimization methods have revitalized the interest in neural networks. ANNs are one of the most crucial fields of artificial intelligence, having a tight association and playing a significant part in many different artificial intelligence. ANNs analyze data in a manner that is similar to how the human brain does so, by taking advantage of the huge breakthroughs in computer technologies [1] and advances in neuroscience to understand the mechanisms of the mind in the processes of logical production and information processing, and all the types of intelligent behavior that characterize the human race. The complexity of the human brain exceeds that of any other known structure in the cosmos, it consists of 86 billion neurons connected by trillions of synapses [2], this takes up a minimal amount of space and uses relatively little energy, furthermore, the human brain is being the smartest and fastest operating system on the planet. Many experiments are currently being carried out to create neural network circuits to simulate the brain. However, traditional electronics tend to obstruct the incredibly complex wiring required for neural networks, but optical energy can reduce electric charge interference and allow information to travel far away. Optics will have distinct advantages over electronics in terms of lower power usage and lower latency [3], it also has large bandwidth, high connectivity, and inner parallel processing capabilities, allowing it to accelerate partial program execution the light speed. These make ONNs a potential alternative to ANNs. Although

neural networks may approximate any function, their actual strength lies in their ability to "learn" that ascension. The back-propagation algorithm is almost universally used for neural network training. This approach is difficult to implement optically because it requires the network's non-linear parts to respond differently to forward and backward light propagation. Existing ONNs are trained with, or heavily assisted by digital computers when faced with these issues. As a result, optics' significant advantages are frequently underutilized. Developing an all optically trained ONN to take advantage reaches advantages is still a work in progress [4]. In addition to studying biological neural networks (BNNs), research on ANNs aims to mimic human intelligence depending on streamlined controlled mathematical models have also been extensive [2].

## 1.2 Artificial Neural Network

By connecting neurons in different layers of the neural network and giving them significant generalization and robustness, ANNs imitate the structure of the nervous system. This branch is one of the most active fields of computer science. The artificial neural network lookup work has improved significantly since the 1980s. In the disciplines of pattern recognition, economics, biology, robotics, autonomous control, estimation, and others. It has effectively resolved numerous practical issues that are difficult to tackle with cutting-edge computers [2].

The basic units of BNNs are biological neurons, which are biological counterparts to artificial neurons. Countless neurons of all types and sizes make up BNNs. Simplified neuron components are dendrites, soma, axons, and synapses, as illustrated in figure (1.1.a). ANNs are a streamlined version

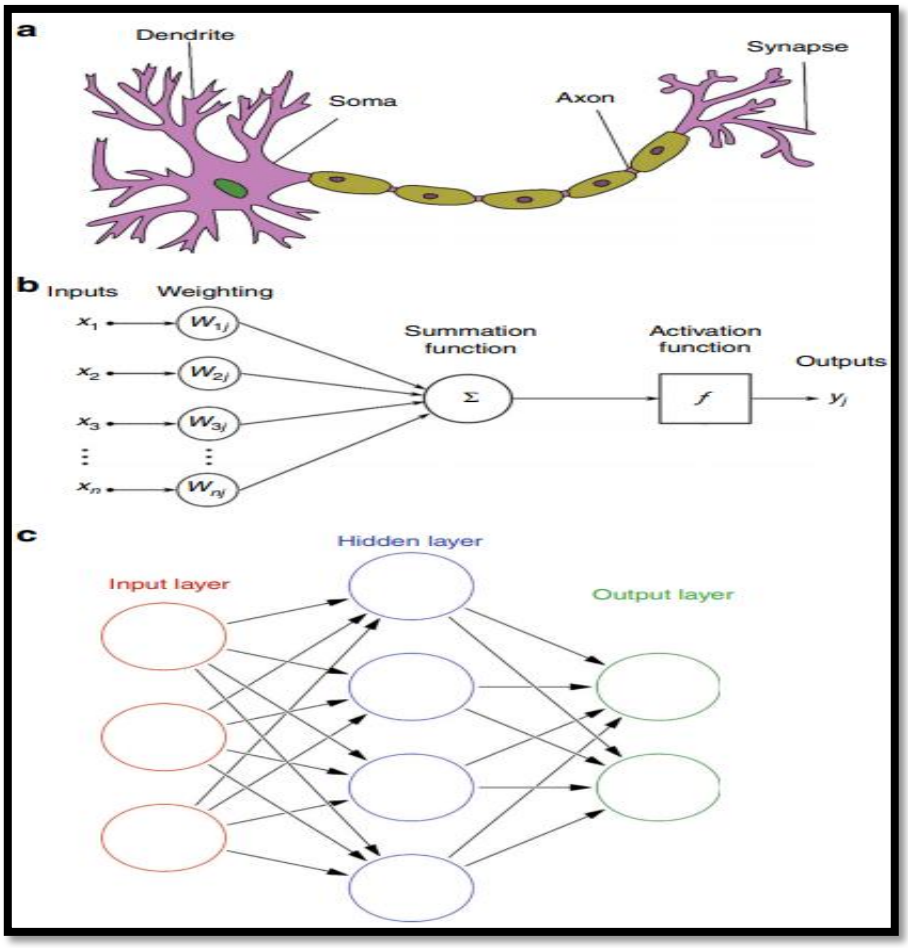


Figure (1.1): The working principle of artificial neural networks. (a) A schematic drawing of biological neurons. (b) A schematic drawing of artificial neurons. (c) The structure of feed forward artificial neural networks [5].

inspired by BNNs. McCulloch and Pitts-invented simple formalized artificial neuron that serves as the basis of this model which functions as a computational component of ANN. A lot of artificial neurons are simultaneously activated when a task is being carried out. As seen in figure (1.1.b), artificial neurons received a single or several inputs ($x_i$) from other neurons, which are summed up to form outputs ($y_i$), which were then delivered to the other neurons via axons. A transfer function or activation function ($f$) known as a non-linear function that may be a Hyperbolic Tangent

(Tanh) function, Rectifier Linear Unit (ReLU) function, etc., is separated by weight ($w_{ij}$). A mathematical model of the artificial neuron is expressed in equation (1.1) [5]:

$$y_i = f(\sum_{i=0}^{n} w_{ij} x_i) \quad (1.1)$$

The feed-forward ANN, a popular architecture, is illustrated in figure (1.1.c). The information in this network only flows in a forward direction. The input layer is made up of a collection of artificial neurons. In the ANNs, the input signal is directed utilizing a single or several hidden layers. The output layer offers the outcomes at the end of this construction. Unlike today's von Neumann computers, which do tasks based on the program's pre-design, through a series of training sessions and sample tasks, ANN may learn how to carry out a task. For instance, the supervised mastery method involves adjusting, weights using back-propagation errors between target values and output values [5].

The Very Large-Scale Integration (VLSI) systems were used to validate the ANNs that are certainly based on electronics. The chip's digital wires can function as digital axons and dendrites. In VLSI, a collection of transistors is utilized to approximate digital synapses and soma. Very large-scale integration based ANNs with a lot of artificial neurons were produced. But, due to the intricacy of architecture, the built-in degree of such an ANN is still modest. The majority of today's ANNs are software programs that run on digital von Neumann computers. Although this strategy has yielded many wonderful results, it does have some drawbacks: First, as the number of transistors in CPUs and GPUs grows exponentially, current leakage in nanometric nodes starts to account for a sizeable portion of power usage,

putting an end to the increase in microprocessor clock rates at around 4 GHz. Second, because of the breakdown of Dennard scaling, even slight increases in CPUs or GPUs performance may result in significant increases in energy usage and heat production. For example, inside the current von Neumann architecture, a supercomputer would need to be extremely powerful and consume at least 500 MW of power to simulate an ANN software system at the size of the human brain in 100% real time [5].

The two aforementioned drawbacks suggest that it is challenging to train a data hungry ANN (where "training" refers to an improvement system depended entirely on modifying ANNs parameters to allow it to perform precise functions) on a von Neumann computer which requires a lot of time and a lot of power will not be greatly reduced soon [5].

## 1.3 Optical Neural Networks

Photons are perfect information carriers due to their distinctive characteristics, which include their naturally high parallelism, quick propagation speed, and lack of adverse effects from reciprocal interference. Optical technology may be able to address some of the problems that electronics have by multiplying optical signals at the time, space, polarization, angular momentum, and wavelength domains. The research and advancement of optical connector technology have led to the use of optical waveguides or fibers instead of copper connections in computer chips and data centers that could improve the performance of ANNs using electronics and software modeling [5].

An optical Neural Network (ONN) is a type of artificial neural network that uses optical (electrical) devices to accomplish physical processes. The

physical implementation approach could be used to classify ONN [6], as illustrated in figure (1.2).

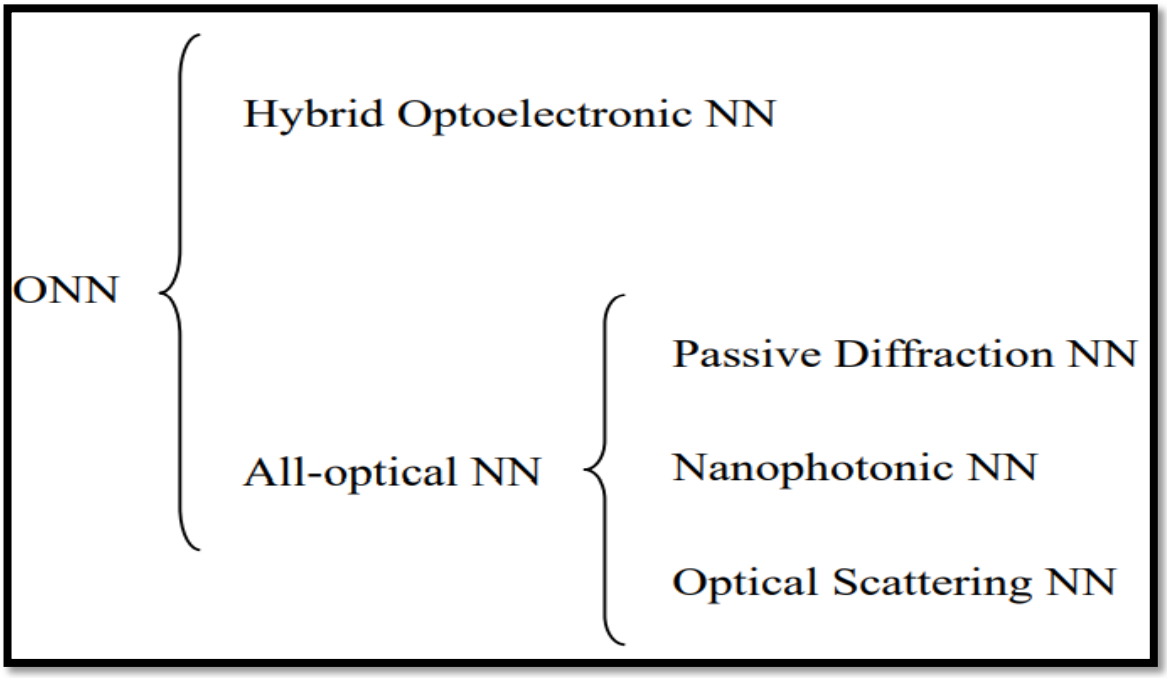


Figure (1.2): The classification of ONN [6].

According to different physical implementation approaches, ONN can be subdivided into Hybrid Optic-electronic Neural Network (HONN) and All-Optical Neural Network (AONN) [6].

### 1.3.1 Hybrid Optic-electronic Neural Network

HONN is a type of ONN that uses photoelectric conversion devices to send data while computing. Its key characteristic is the use of optical devices for partial layer manipulation. By photoelectric conversion, the results of the optical computation are fed into the traditional neural network of the remaining layers [6].

### 1.3.2 All-Optical Neural Network

AONN is a type of ONN that does not use photoelectric conversion devices to transmit data during the computation process. Optical devices are used to carry out all of the computations. AONN can be classified into three types of neural networks based on how optical devices are used: passive diffraction neural networks, nano photonic neural networks and optical scattering neural networks [6].

Figure (1.3.a) shows the structure of the Diffractive Deep Neural Network ($D^2NN$). It's made up of numerous diffractive surface layers. It is conceivable to achieve computing functions in the form of photons by interacting with these diffraction surfaces. The training $D^2NN$ is conducted using an electronic computer. After training is complete, Poisson Surface Reconstruction (PSR) is used to create 3D models of each diffraction layer of the network using $D^2NN$ parameters. The models are then printed using a 3D printer. Figure (1.3.b) and figure (1.3.c) show the produced neural network diffraction layers, every printed layer could be thought of as either projective or reflecting. The points on a layer represent neurons that can reflect and transmit light waves. These points are linked to successive layers via optical diffraction, allowing for a forward propagation method that can perform a variety of complex function processes at the speed of light [6].

In D²NN, neuron inputs are complex values. Diffraction in free space determines the weighting parameters, which are related to the actual distance between layers. As demonstrated in figure (1.3.d), multiplier biases are determined by the transmissions or reflections coefficient of each neuron. Each neuron in D²NN contributes to the modulation of the amplitude and

phase of input optical waves, as well as the production of a secondary wave. Through wave propagation and coherent or partially coherent interference coupling, the output of every neuron in the previous layer gives a unique inter-connectivity to the network. Though a secondary wave created by the neurons diffracts in all directions, the wave's intensity decreases according to the propagating distance. Waves could only be obtained within a finite radius for each neuron in the following layer. Light leaking beyond the limiting radius will reason optical power loss in the complete network. The network layer spacing, incident beam light intensity, detection SNR, coherence length, and radius of the light source are all factors that influence D²NN interconnections [6].

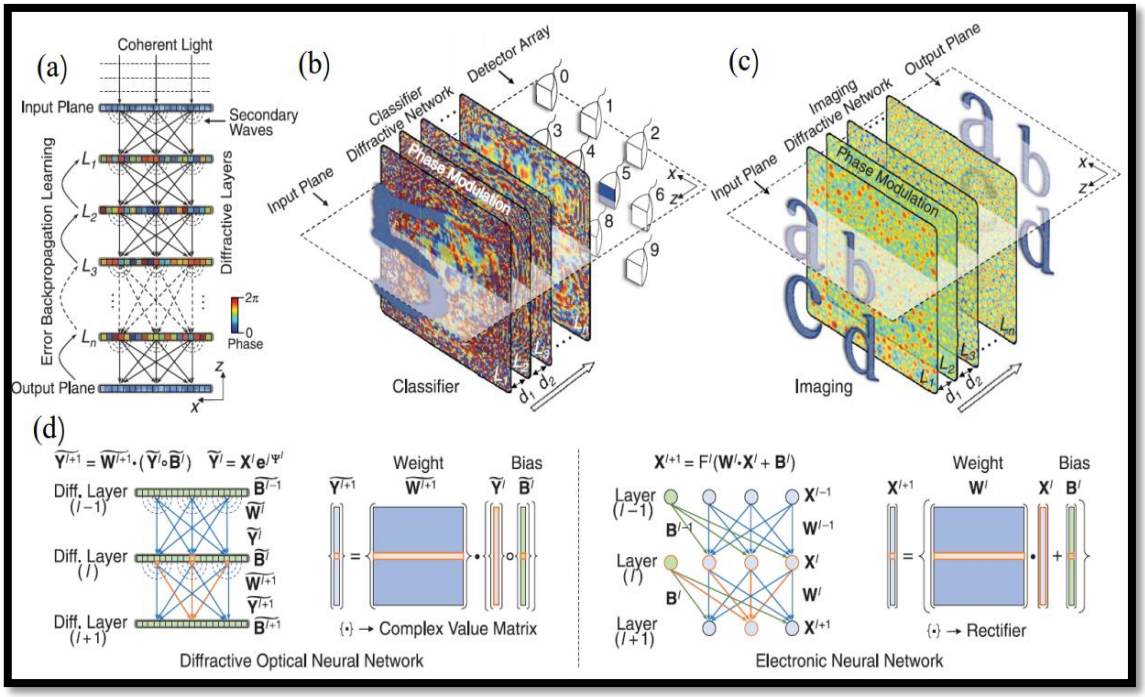


Figure (1.3): A structure of D²NN. (a) The overall structure of $D^2NN$. (b) The usage of $D^2NN$ for image classifier. (c) The usage of D²NN for imaging. (d) The difference between diffractive ONN and electronic NN [6].

## 1.4 Literature Review

The motivation behind this work, which there is an excellent technique that can be applied to increase the working performance of a neural network which is a very challenging task in any electronic system. To study neural network technologies, it is necessary to review most of the related previous work as follows:

### 1.4.1 Literature Review of Artificial Neural Networks

**Y. guo Wang and H. Peng Li, (2010)** [7] provided a technique for sample purification depending on statistical analysis theory that may purify training samples for better wetlands remote sensing categorization depending on ANN. For complex areas, BP-ANN with a non-linear mapping function can produce good classification results.

**N. Sharma and T. Gedeon, (2013)** [8] provided a computational model that can discriminate between the electrocardiogram, galvanic skin reaction, and blood pressure signals that are acquired from sensors and find corrupted signals that may happen as a result of hardware problems, like sensor failure. The study also examines the effects of signal modeling at various time scales and decides which signal time scale is most appropriate for categorization. With this technology, distorted signals can be automatically identified during the collection of signal data and used as a support tool when gathering real-time sensor data.

**M. Jabardi and H. Kaur, (2014)** [9] offer two artificial neural network classifications for handwritten digit recognition (from 0 to 9), using a feed-forward multi-layer neural network application with two separate classifiers:

1. Forward Multi-Layer Neural Network FMNN., 2. Binary Coding Neural Network BCNN. In terms of handwritten digit recognition, the highest recognition reliability and the lowest mistake rate have been attained.

**A. Graves and N. Jaitly, (2014)** [10] offered a speech recognition system that transcribes voice data directly with text, without the need for an intermediate voice representation. The system depends on a combination of Long Short Term Memory (LSTM) deep bidirectional recurrent neural network architecture and temporal classification target function. A modification has been made to the objective function of network training to reduce the expectation of an arbitrary copy loss function. This allows for a direct improvement in the rate of word errors, even when there is no lexicon or language model.

**K. Malathi and R. Nedunchelian, (2016)** [11] suggested a unique algorithm for accurately identifying blood vessels. The Region Of Interest (ROI) approach and Kirsch's templates methodology are used to extract the vein for segmenting the fundus retina image. An artificial neural network-based firefly clustering algorithm is employed for illness classification. The diameter of the vein and the size of the cotton wool patches are utilized to categorize the affected area. The technique has produced sufficient results to justify the diagnosis of diabetic retinopathy in four phases. Regardless of whether the retina is in normal or bad condition, accurate detection is successfully established.

**E.G. Silva Júnior, et al., (2017)** [12] presented a classification of 20 genotypes by using the fiber quality index, showing that artificial neural networks outperformed when utilizing fiber length paired with the micronaire

index, short fiber index, and fiber maturation when using only fiber length and prior associations.

**M. B. Mehmandari, et al., (2019)** [13] suggested the development of Acoustic Emission (AE) method when it comes to rotor and stator wear detection and categorization. First, the recorded AE signals' features are extracted using time-domain signal analysis techniques. A Feed-forward Neural Network (FFNN) is then used to categorize AE signals using retrieved features. The Bayesian Regularization and Levenberg Marquardt training techniques are used to train the neural network. 450 signals are divided into two classes using this method: 300 with wear and 150 without wear. The outcomes showed that the Bayesian Regularization training algorithm was more effective than Levenberg Marquardt. This algorithm's performance was enhanced by Levenberg Marquardt and Genetic Algorithm (GA).

**M. Der Yang, et al., (2020)** [14] suggested using of an integrated strategy for Hyper Spectral Image (HIS) classification tasks in artificial neural networks employing the Minimum Noise Fractions (MNF) and Hilbert-Huang Transform (HHT) transformations. HHT and MNF serve as feature extractors and picture decomposers, respectively, to reduce the effects of noise and dimensionality and to enhance the performance of training data. Two benchmark datasets were employed in the study: Indian Pine and Pavia University (PaviaU) hyper spectral pictures (IP). To enhance the number of required neurons and training instances in ANN, the corresponding classification accuracy for 1 to 1000 neurons and four ratios of training samples are evaluated.

**S. M. K. Chaitanya and P. Rajesh Kumar, (2020)** [15] proposed a methodology where the kidney images are trained and evaluated to distinguish between normal and pathological pictures. By employing the Gabor feature extraction method, the images are segmented the features are extracted, and these retrieved features are then chosen using the CS-ANN approach.

**A. N. Ismael, et al., (2020)** [16] suggested the method groundwater quality of more than 60 wells in Basra's Safwan and Al-Zubayr areas are assessed using artificial neural networks with feedback. The physical and chemical parameters of groundwater are the elements employed in the suggested categorization and verification model. The taxonomy model successfully developed and implemented more efficient and sustainable groundwater management methods in the research area due to its strong performance and precise taxonomic capacity.

**D. Vujičić, et al., (2020)** [17] described ANN for classifying asteroids into families. The Hierarchical Clustering Method was utilized to gather the data for artificial neural network training and testing Hierarchical Clustering Method (HCM). The researchers have demonstrated that ANNs may be used to validate the HCM in families with a high number of members.

### 1.4.2 Literature Review of Optical Neural Networks

**A. Jha, et al., (2016)** [18] suggested silicon nitride create a photonic reservoir node on an integrated non-linear photovoltaic device. Its non-linear dynamics originate from the Kerr effect which is brought on by optical intensity dependence and cavities. Due to its completely passive optical construction, it processes information very quickly enabling time delay reservoirs with rapid virtual nodes and minimal latency. Show experimentally how the reservoir

node in time-delay architecture can complete two benchmark tasks: (1) Binary classification on a dataset of earthquake sensor time series. (2) Time series prediction using a Non-linear Autoregressive Moving Average (NARMA-10) with a normalized root mean square error of 0.183. These results pave the way for non-linear photonic node-based photonic computing with high throughput and minimal latency.

**G. Fennessy, (2018)** [19] created an innovative ONN framework that a degree of scalar invariance to picture classification estimates. Images are divided into several levels of different zoom based on a focal point, taking inspiration from the human eye which has superior resolution near the center of the retina. An identical CNN is used to process each level in a Siamese fashion, and the combined results yield a highly accurate estimate of the item class. Since ONNs act as a wrapper around CNNs, they can be used for a variety of a wide range of existing algorithms to generate noticeable accuracy gains without modifying the underlying architecture.

**J. Chang, et al., (2018)** [20] suggested a technique for utilizing the intrinsic convolution of a linear spatially invariant imaging system to produce an optical convolutional (Opt-Conv) layer with an optimizable phase mask. To test their architecture, they first employ two simulated image classification models. The study found that the primary use of the convolutional layer consists of a single convolutional layer that matches image templates, an optical link that has been studied for optical target recognition and tracking. By feeding the output of the convolutional layer into a digitally fully connected layer, they demonstrate how the supplied Opt-Conv layer may integrate into a bigger hybrid optoelectronic CNN. The study convincingly shows that, in both instances, the simulated optoelectronic implementation of

the identical network structure competes with an unrestricted electronic implementation in terms of classification accuracy.

**S. Geoffroy-Gagnon, et al., (2019)** [21] described the presented optical neural network, a synthetic dataset was produced. The data set consists of four Gaussian distributions, each of which is filled with a set of four-dimensional points, designated as Inland Environmental Resources (IER). The different Gaussian distribution classes are one-hot encoded so that the ground PH OER is the truth vector for a single sample and 1 is set as a different value for each class. A single-layer neural network that is created using a traditional computer accurately classifies this dataset. To better comprehend the benefits and constraints of this optical neural network, it is now possible to compare it to a perfect classifier.

**R. Hamerly, et al., (2019)** [22] suggested a new coherent detection based photonic accelerator with large (GHZ) speeds, little (subattojoule) energy per Multiplied And Accumulated (MAC), and scalable to enormous ($N \gtrsim 10^6$) networks. This accelerator makes use of the substantial spatial multiplexing that is offered by standard free-space optical components. Unlike earlier techniques, both inputs and weights are optically recorded, making it possible to quickly retrain the network. ONNs have a "typical quantum limit" that is determined by photo detector shot noise. That limit, which can be as low as 50 $ZJ$/MAC, demonstrates the possibility of this technology to execute thermodynamic (Landauer) constrained digital irreversible computing. The suggested accelerator supports both convolutional and fully connected networks. The study also offerS a back-propagation and training scheme that utilizes the same gear. This design will make it possible to develop new ultralow energy deep learning processors.

**D. Zhang, et al., (2019)** [3] introduced a silicon-based fully connected ONN, which can be used to classify and recognize images. It is built on a completely connected neural network. In chip design, the one-layer model has been employed. As long as photons are big enough, chip simulations demonstrate that precision will not be impaired. This structure has the potential to be used in deep learning.

**M. Y.-S Fang, et al., (2019)** [23] suggested training two ONN to categorize handwritten digits, one with a more customizable design (GridNet) and one with improved fault-tolerance (FFTNet). When replicated flawlessly, GridNet outperforms FFTNet in terms of accuracy. However, the more fault tolerant FFTNet surpasses Grid Net when there is a slight inaccuracy in their photonic components.

**M. Paraschiv, et al., (2020)** [24] suggested creating and improving deep learning for the categorization, or species identification of fish photos acquired by underwater camera arrays. The Symbiosis project, which is concerned with the identification and classification of pelagic fish in the Mediterranean Sea and the Atlantic Ocean is the context for our work. In this project, mooring is used to drop an autonomous system below the water's surface. The researchers used two camera arrays like sonars and other sensors, the system monitors the species of each fish within a specified radius in real time. A remote observation station receives aggregated data from the computation of real-time statistics.

**H. Zhang, et al., (2021)** [25] proposed to highlight an Optical Neural Chip (ONC) that uses neural networks with genuinely complicated values. The researchers evaluate their complex-valued ONC's performance in four different contexts: simple Boolean operations, the classification of species in

an Iris dataset, handwriting recognition, and the classification of non-linear datasets (Circle and Spiral). When it comes to powerful learning capacities, their complex-valued ONC performs better than there real-valued counterpart, for example (quick convergence, high accuracy, and the capacity to create non-linear decision boundaries).

**Y. Sun, et al., (2021)** [26] proposed to combine the ONN with complex-valued neural networks and add a non-linear activation function to the structure resulting in ($N - D^2NN$) that depends on a wavelength of 10.6 um. Leaky-ReLU, Parametric ReLU (PReLU), and Randomized ReLU (RReLU) are among the upgraded activation functions of ReLU, which is specifically chosen to represent the activation function of the $N - D^2NN$ model. The high representation capabilities of a 10.6 um wavelength-based $N - D^2NN$ models are demonstrated by numerical simulation. Using the MNIST handwritten digital dataset and the Fashion-MNIST dataset, respectively, it was able to complete classification learning tasks with success.

**A. RYOU, et al., (2021)** [1] presented a free-space optical ANN with linear weight summation based on diffraction and non-linear activation made achievable by saturable atom heat absorption. With a single layer, in both simulation and experiment, the researchers can categorize pictures of handwritten digits, they find that this is more accurate than using a linear model. Their technology keeps free-space optics huge parallelism even in the presence of physical non-linearity paving the road for creative designs and wider adoption of optical ANNs.

**T. Wang, et al., (2022)** [27] provided a method for image classification using on average <1 photon detected for each scalar multiplication and required

specialized experimental instrumentation to execute optical vector-vector dot products. This approach is in line with theoretical forecasts for ONNs' quantum-limited optimal efficiency. Figure (1.4) gives an overview of the history of ANNs and ONNs.

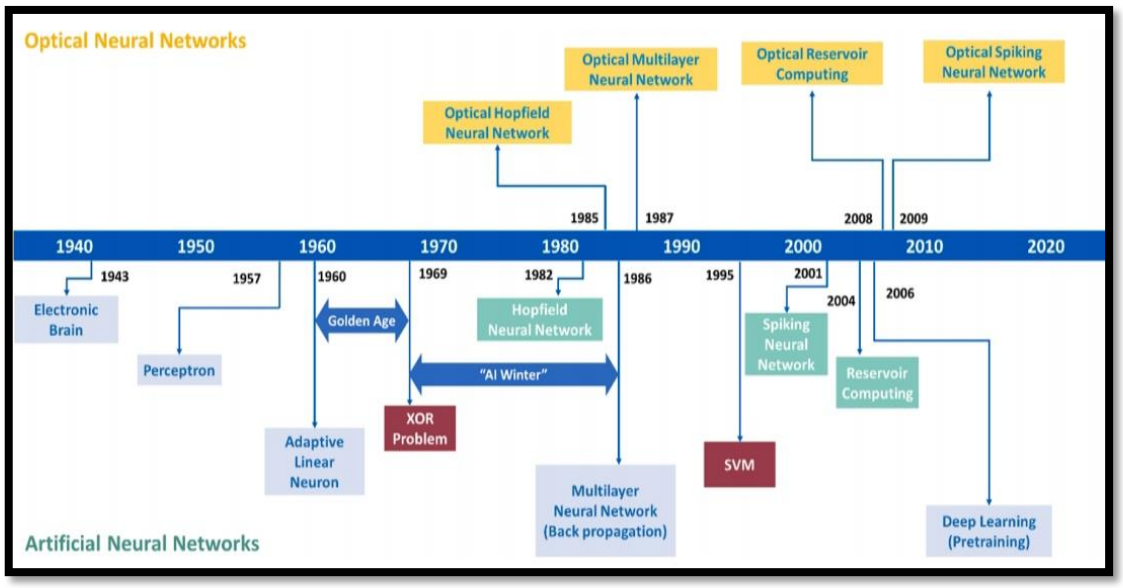


Figure: (1.4): Development of ANN and ONN [2].

## 1.5 Problem Statement

In complex circuits, electronics tend to prevent the signal from traveling too far, whereas the usage of optical power can remove interference caused by electrical charges and allow signals to travel farther. This work directly addresses the issue of signal processing problems at high speeds are solved by the usage of ONN through non-linear units in a manner that is both consistent with modern neural network architectures and compatible with leading ONN proposals.

## 1.6 Aims of the Study

The aims of this study are:

1. Design and implementation of ONNs.

2. Make a test to operate the ONNs.

3. Entering designed networks by practical application.

4. The simulation was carried out by applying a selection-based back-propagation algorithm by training an optical neural network to classify the collected dataset for image type recognition.

## 1.7 Outline

Chapter one: provides a summarized introduction of this thesis, literature review, aims of the study, and highlights the most important chapters of this thesis.

Chapter two: discusses all the concepts associated with the datasets, artificial neural networks and optical neural networks.

Chapter three: proposes a system of artificial neural networks, and optical neural networks.

Chapter four: demonstrates the results of the proposed method and discussion.

Chapter five: introduces the conclusions of the thesis and the future works.

# CHAPTER TWO

# Theory

# CHAPTER TWO

# Theory

## 2.1 Introduction

This chapter introduces the basic concepts that were used to implement neural networks from the side of theory and mathematical equations. The deep learning, CNN, dataset, and the types of datasets are described. After that, this chapter will be divided into two parts: the first part describes an ANN, some of its types, and common algorithms. The second part describes the methods of implementing an ONN, some types of activation functions, loss functions, and their algorithms.

## 2.2 Artificial Neural Network

ANNs are brain-inspired algorithms that are used to foresee problems and model complex patterns. ANN is a deep learning technique that was inspired by the idea of BNNs in the human brain. An effort to mimic the functions of the human brain led to the creation of ANN. Although they are not exactly the same, ANNs and BNNs have very similar workings. ANN algorithm only accepts structured and numeric data [28].

### 2.2.1 Architecture of Artificial Neural Network

The exchange of electrical pulses between cells, known as action potentials, occurs during the information-processing step of neural networks, which are formed by trillions of neurons (nerve cells). Artificial neural networks are the correct term to distinguish computer algorithms that mimic these biological structures from the squishy components inside animals [29].

Biological neural networks are used to create ANNs. Dendrites, axons, cell bodies, and synapses are the four basic units of biological neurons, as in figure (1.1.a). When a neuron receives a stimulus, it normally contains several dendrites that receive data. This neuron's axons send information to other neurons. Only one axon with a lot of terminals can transmit information to many additional neurons [30].

As seen in figure (1.1.b), axon terminals, which are connected to the dendrites of other neurons are responsible for signal transmission. Artificial neurons are information processing units with input signals, weight $w_i$, bias $b_i$, linear summation, and a threshold function $\Phi$ similar to biological neurons. Axons and dendrites are represented by the connections between signals (dendrite input and axon output), cell body activity is represented by the threshold and linear summation, synapses include weights and memories as illustrated in figure (1.1.c).

In feed-forward, the artificial neurons are given the inputs $x_i$, and each corresponds to a weight. Then weight and sum the input, carry out non-linear activation, and finally obtain the output $s$. It is getting the outputs as close to the genuine thing as feasible is the aim of training. Until the network is converged, it is regularly updated $w_i$ and $b_i$ in the training process. The form of $s$ is as follows in equation (2.1) [30]:

$$s = \Phi\left(\sum_{i=1}^{n} w_i * x_i + b_i\right) \tag{2.1}$$

Depending on the network architecture, there are several different types of ANNs: feed-forward neural network, recurrent neural network, radial basis function neural network, CNN, and self-organizing neural network [31].

### 2.2.2 Training of Artificial Neural Network

A network is prepared to be taught once it has been set up for a specific purpose. The initial weights are picked randomly to begin this process. The training or learning process then starts. Training can be divided in two ways: supervised and unsupervised. In supervised training, the network is given the required output either by having its performance "graded" manually or by giving it the desired outputs along with the inputs. The network must understand the inputs on its own during unsupervised training [32]. There are two types of training algorithms:

#### 2.2.2.1 Supervised Training

In supervised training, as illustrated in figure (2.1), the outputs and inputs are both available. The network compares the generated outputs to the desired outputs after processing the inputs. As a result of the errors propagating back through the system, the system then adjusts the weights that regulate the network. The process of propagating errors across the network is repeated over and over until the weights are adjusted. The datasets that are trained are called the “training set”. The training process can last for days and it only ends when the system reaches an accuracy or statistically desired point. Some networks, however, never learn. This can be the case because the input data lacks the particular information needed to produce the intended output. The absence of sufficient data prevents networks from converging completely. Ideally, there should be enough information to allow for the holding back of a portion of the data for a test to evaluate the performance of the model [32].

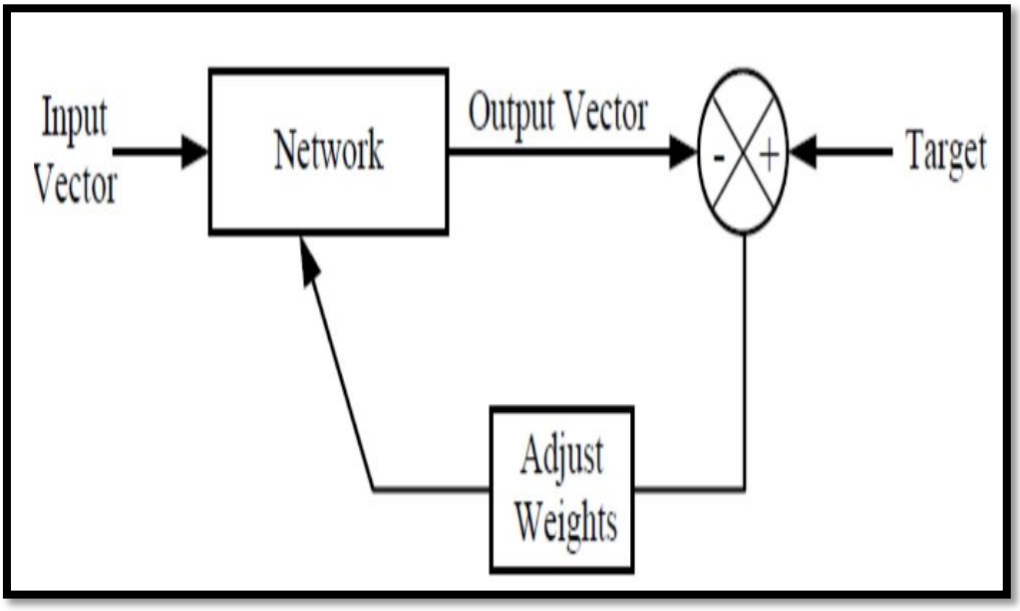


Figure (2.1): The block diagram of supervised training [33].

### 2.2.2.2 Unsupervised Training

Unsupervised training, as illustrated in figure (2.2) does not require an exact target dataset. The system itself must then choose which features to employ in order to gather the input data. This is frequently referred to as adaptability or self-regulation. This is especially useful when there are no known answers [32].

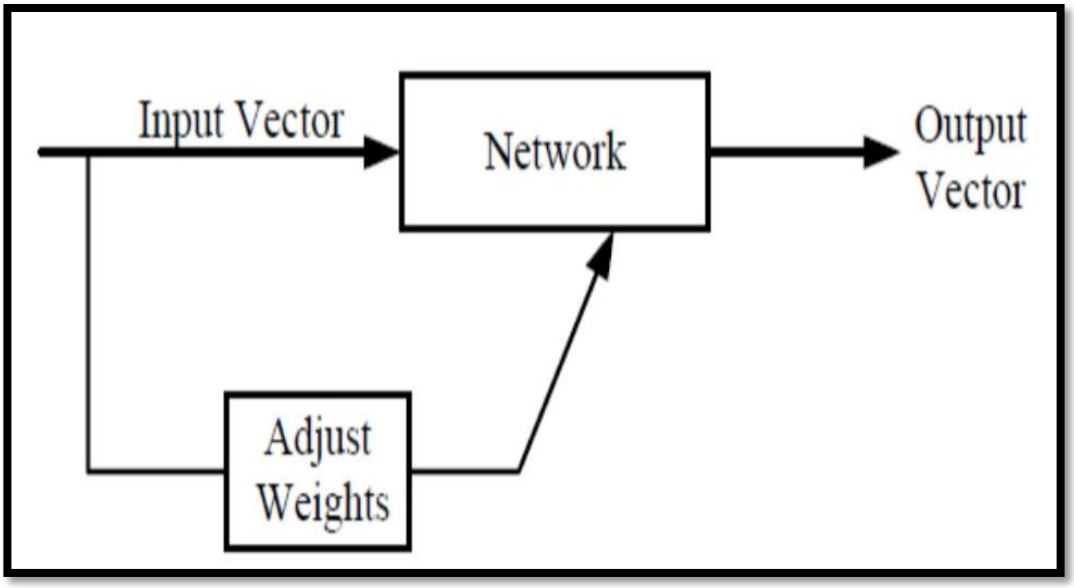


Figure (2.2): The block diagram of unsupervised training [33].

### 2.2.3 Back-propagation Training Algorithm

Let me point out that $E_p$ the error function for pattern $P_i$, the target output for pattern $PT_{pj}$ on node $j$, although $o_{pj}$ reflects the actual output at the node $w_{ij}$ which is the weight from node $i$ to node $j$.

Step 1: Set up the weights and offsets.

Set low random values for all weights and node offsets.

Step 2: Present input and desired output

Presents a continuous valued input vector $x_{0,}x_{1,}, \ldots \ , x_{N-1}$ and describe the preferred outputs $t_{0,}\, t_{1,} \ldots , t_{N-1}$. In the case of using the network as a classifier, all preferred outputs are typically set to zero except for the one that corresponds to the class the input belongs to. This desired result is 1.

Step 3: Calculate the actual outputs

Use the sigmoid non-linearity.

$$f(\alpha) = \frac{1}{1 - e^{(\alpha - \theta)}} \tag{2.2}$$

To calculate outputs $o_0, o_1, \ldots, o_{N-1}$.

Step 4: Adapt weights

Use an iterative algorithm to advance from the output nodes to the initial hidden layer. To modify weights, he uses:

$$w_i(t+1) = w_{ij}(t) + \eta \delta_j x_i' \tag{2.3}$$

Where $\eta$ is a gain term, $\delta_j$ is an error term for node $j$, $x_i'$ is either the output of node $i$ or an input, and $w_{ij}$(t) is the weight from hidden node $i$ or from an input to node $j$ at time $t$.

If output node is node $i$, then

$$\delta_i = o_i \,.\, (1 - o_i) \,.\, (t_i - o_i) \tag{2.4}$$

Where:

$t_i$: Desired output of node $i$.

$o_i$: Actual output.

If internal internal node is node $j$, then

$$\delta_j = x'_j.\left(1 - x'_j\right).\sum_{k>j} \delta_k \, w_{jk} \tag{2.5}$$

Where $k$ is distributed over all nodes in the layers above node $j$. Similar modifications can be made to internal node thresholds by assuming that they are connection weights on links from auxiliary constant-valued inputs. Convergence occurs faster. If a momentum term is given and weight changes are smoothed with [54]:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_i x'_i + \alpha \left(w_{ij}(t) - w_{ij}(t-1)\right) \tag{2.6}$$

Where $0 < \alpha < 1$.

Step 5: Repeat by going to step 2 [34].

## 2.3 Deep Learning

Deep learning is a subset of machine learning that is itself a branch of artificial intelligence [35]. It consists of hierarchical architecture, where each higher layer builds on its previous lower layer, which makes it popular for the first time as hierarchical learning [36]. The beginning of deep learning was in 2007. Deep learning works well with a massive amount of data to solve a complicated problem, which allows giant technology companies including Facebook, Microsoft, Amazon, Baidu, and Google to use today. Many applications improved with deep learning such as Object Recognition, Object detection, Automatic Machine Translation, Investment Modeling, and drug discovery.

Because the hierarchical representation of the neural network, deep learning manages data in a non-linear approach, so deep learning can extract

high-level features from the input data. For example, in image processing, deep learning can be used in several applications such as edge extraction, image segmentation, shape recognition, and face verification.

Deep learning algorithms such as CNN boost their efficiency on conventional algorithms due to the automated function learned without the need for human computational effort as it does in conventional machine learning techniques. The conventional machine learning technique requires a feature vector corresponding to the raw data (intensity pixel values in the case of an image) as an input to detect and process it so that the handcrafted features must be extracted and later fed into a classifier. Deep learning algorithms can quickly learn these features on their own. The goal of designing deep learning algorithms is to overcome the problems that traditional machine learning algorithms have been unable to solve [37]. The most important problems that deep learning overcame can be described as follows [37] [38].

1. Curse of Dimensionality

Many fields such as numerical analysis, data processing, and machine learning are faced with the problem of curse dimensional. The popular theme of curse dimensional problems is that the amount of space increases so rapidly that the data available are sparse as the dimensionality increases. Deep learning has overcome this problem due to the enormous ability of its algorithms to handle multiple dimensions [36].

2. Local Constancy and Smoothness Regularization

Machine learning algorithms require to be driven by previous beliefs about the type of function they can learn. The smoothness or the local

consistency is among the most commonly used beliefs. An algorithm is said to have smoothness if within a small region the learned function should not be changed very much. Many simpler Machine learning algorithms depend on local consistency beliefs to generalize well, and so these algorithms fall to scale statistical challenges in AI tasks [36].

Deep learning may be used to overcome numerous problems with applications for computer vision, some of which have been solved through deep learning including object recognition, object detection, segmentation, image classification, image caption, speech recognition, generative models, manufacturing, biometrics recognition system , similarity learning, gaming, and many others [36] [38].

## 2.4 Dataset

For a generalization performance, picking the appropriate dataset for a particular classification task is critical. In the end, the quality of the underlying dataset itself has a significant impact on the model's capacity to correctly identify invisible data. The network's input X for an image classification task is made up of a variety of color or gray scale images and the ground truth labels that go with them (Y). It is possible to think of image X as ($c \times h \times w$) tensor of integers with color channels and associated height and breadth. Red, green, and blue (RGB) intensities at each pixel are represented by a color image's three channels. On the other hand, the gray scale intensity is represented by the single channel of a gray scale image. It is common practice to divide the entire data set into three subsets called the training set, validation set, and test set [39].

### 2.4.1 Training Set

The training set is the raw data used to train machine learning models. By specifying learnable parameters in convolutional layers and weights in fully connected layers, training set is a technique for lowering the tolerances between output predictions and base truth labels supplied in a training data set. Back-propagation algorithm is used to train neural networks which largely depends on the gradient descent optimization technique and the loss function. The loss function computes the model's performance under learnable parameters and specific weights using forward-propagation in a training data set. The learnable parameters such as weights are changed by the loss value using back propagation and the gradient descent optimization procedure [39].

### 2.4.2 Validation Set

A validation set is used to optimize and evaluate model parameters [39].

### 2.4.3 Testing Set

A testing set is a set of data that is not used during training. It is usually used only once, the moment model parameters and hyper parameters are completely defined to evaluate the model's generalization performance [39].

## 2.5 Types of Dataset

The datasets used in this method are most common for image classification tasks. Arranged in order of increasing difficulty to generalization, those are MNIST, KMNIST, and EMNIST.

### 2.5.1 Modified National Institute of Standards and Technology Dataset

MNIST dataset is a big dataset that corresponds to the handwritten numbers from 0 to 9 that are flattened and centered into a fixed-size image (28 × 28) pixel. It is contain 70000 image, divided into 60000 training cases and 10000 testing cases. It is used to train image processing systems. In the area of deep learning, it is also utilized for testing and training [40]. Figure (2.3) shows picture of the MNIST dataset.

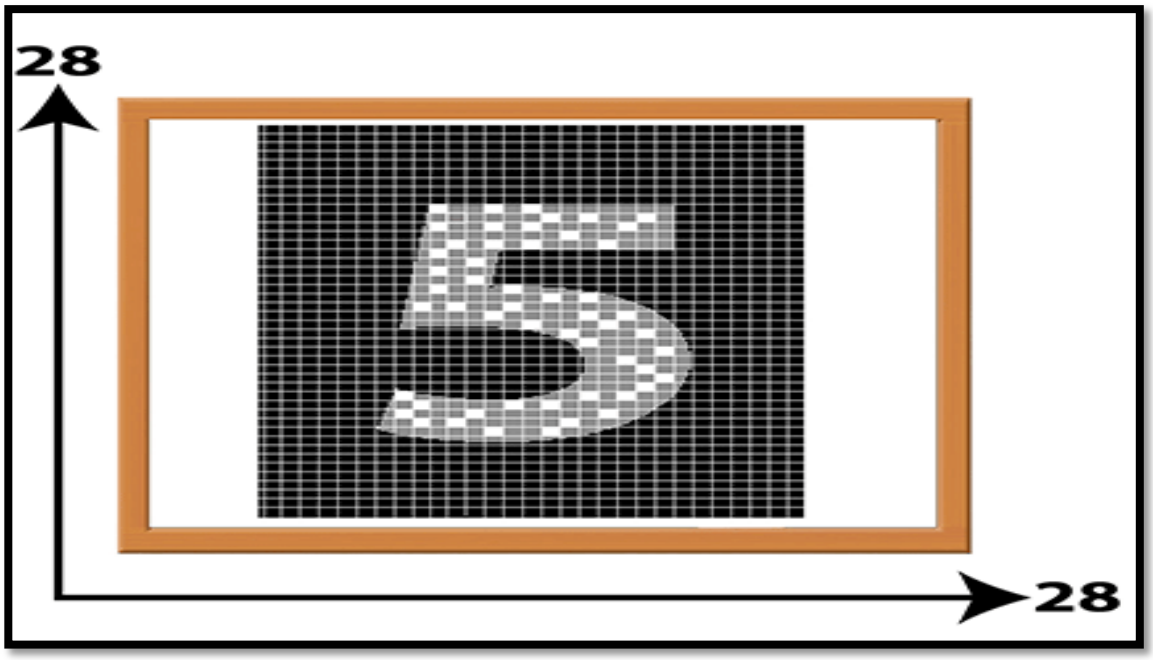


Figure (2.3): Example of the MNIST dataset [41].

### 2.5.2 Kuzushiji-MNIST Dataset

KMNIST dataset is the most famous dataset in the field of deep learning, it contains 10 categories of Japanese cursive handwritten characters, and centered into a fixed-size image (28 × 28) pixel. It is contain 70000 image, divided into 60000 training cases and 10000 testing cases.. It is also

used to train image processing systems [42]. Figure (2.4) shows some other examples of "impossible" characters.

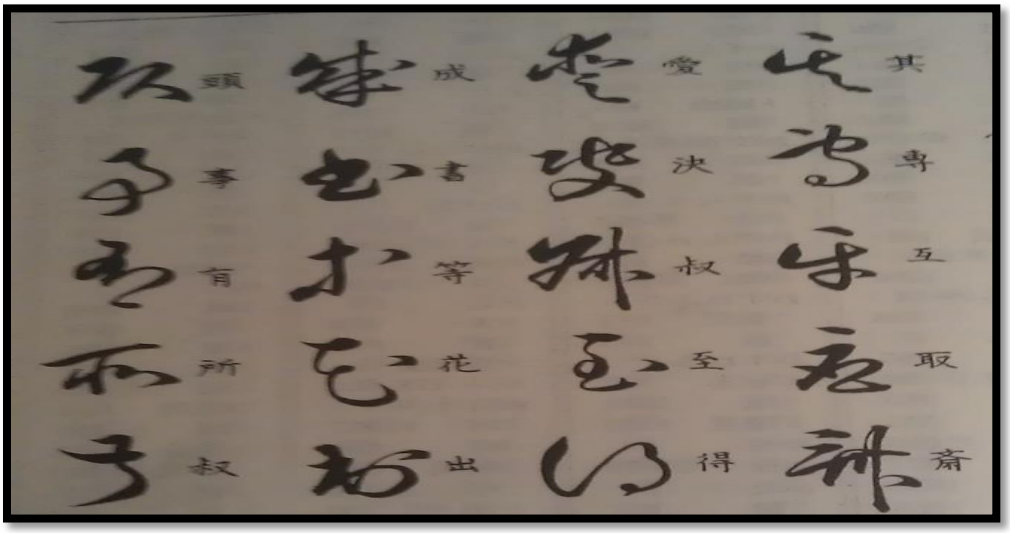

Figure (2.4): Example of the KMNIST dataset [43].

### 2.5.3 Extended-MNIST Dataset

EMNIST dataset is a balanced dataset that contains 47 categories of handwritten letters and numbers, and centered into a fixed-size image (28 × 28) pixel. It is contain 131600 image, divided into 112800 training cases and 18800 testing cases. It is used to train image processing systems in the field of deep learning [44]. Figure (2.5) shows example of the EMNIST dataset.

Figure (2.5): Example of the EMNIST dataset [45].

## 2.6 Mini-Batch

Batch means using all the data to calculate the gradient during one iteration. In the case of a large data set, this process will be very slow. So, using only a small set of all the total data to calculate the gradient during one iteration and make the training process faster, this is called mini-batch. Batch size means the number of samples passed to the network during a single training. The higher the batch size, the more memory space you'll need [39].

## 2.7 Convolution Neural Network

CNN is a kind of deep learning architecture depends on a specific feed forward neural network. The idea for CNN was influenced by the visual cortex exploration of the brain. CNNs are a promising method that can be used

image and video processing and can be applied in image classification and recognition [46] [47]. The CNN performs two functions, the extraction function then the prediction function. Each function used certain layers to achieve a particular purpose. These layers are convolution, non-linear (activation), pooling, flattening and fully connected layers.

### 2.7.1 Basic Parts of Convolution Neural Network Architecture

Generally, the CNN architecture can be divided into two essential parts: Feature extractor part, and prediction part.

1. Feature extractor part

It is the first part of CNN architecture, It does extract image features and transforms into feature maps. CNN's feature extractor architecture consists of one or more convolution layers, representing the activation function used in that layer to distinguish the special signal from the useful features of each hidden layer. the first part of CNN passes through the convolution layer to convolve the filter with the image and produce feature maps which are reduced later with the pooling layer, and use the previously generated feature maps to be as input feature maps and apply the same process on them, it stills go to the layer by layer and still extract powerful feature and get smaller size feature maps. The first layer feature may be corners, lines, etc., while the upper layer includes a harder feature, finally the final powerful feature, reduced dimension feature maps are flattened to generate low dimensional feature vector to be fed into the prediction part [48] .

2. prediction part

After extracting feature maps and reducing dimensions by selecting the best features between them, the low dimensional feature vector fed into a prediction part. The prediction part works as a traditional fully connected artificial neural network. The prediction part performs either the classification or the regression task according to the type of task that the network is training for. When works as a classifier, it returns the likelihood of the class that the input image may belong to it. while when working on regression task it returns a vector of prediction values. The prediction part involves number of fully connected layers to do that purpose [48].

### 2.7.2 Layer of Convolution Neural Network

The CNN has five layers.

#### 2.7.2.1 Convolution Layer

The convolution layer in the CNN is a feature extractor that extracts various features from the input image [35] [49]. Convolutional is a mathematical process to combine two groups of information. It is used convolution filters to extract features from the input image and learn these features using input data arrays whereas allowing the cultivation of the spatial relationship of each feature in the image by using these filters to generate feature maps. The convolution process was accomplished by sliding the filter / kernel across an input image. On each spatial location, the element-wise array is multiplicity and compute the result [50].

To illustrate the convolution layer process in figure (2.6), suppose the input image with size 5 × 5 pixels, and kernel matrix with size 3 × 3 as shown in figure (2.6).

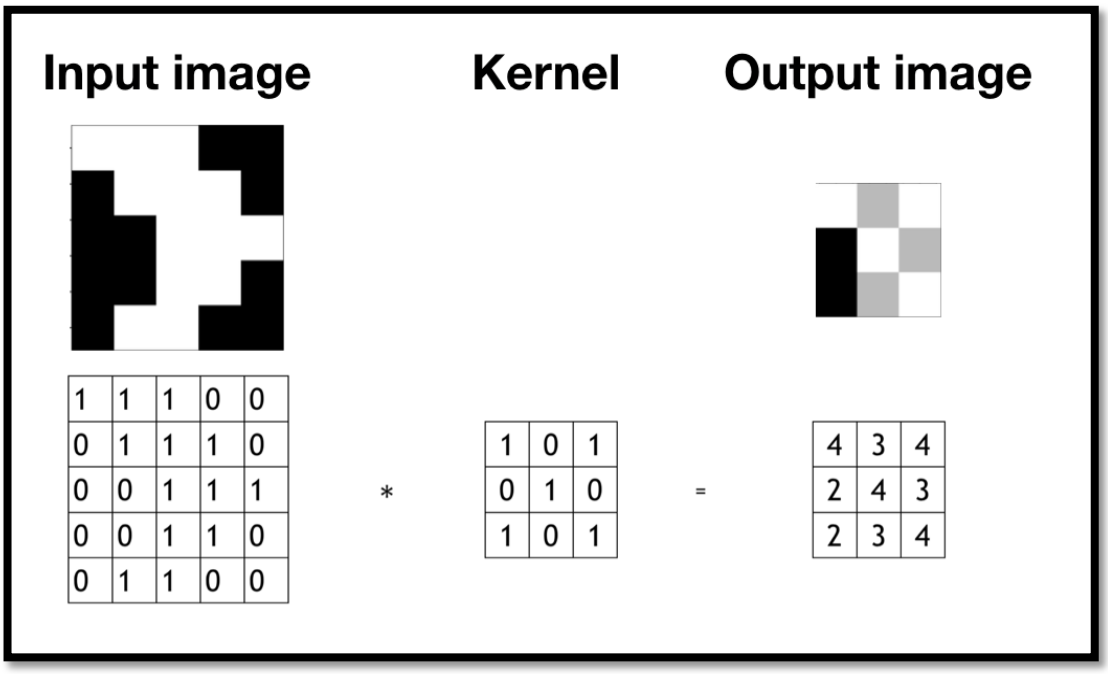


Figure (2.6):The input image and kernel-matrix [51].

Then the filter matrix is applied to the input image in a convolution method, and the output for this process is a "Feature Map". The convolution process is shown in the figure (2.7).

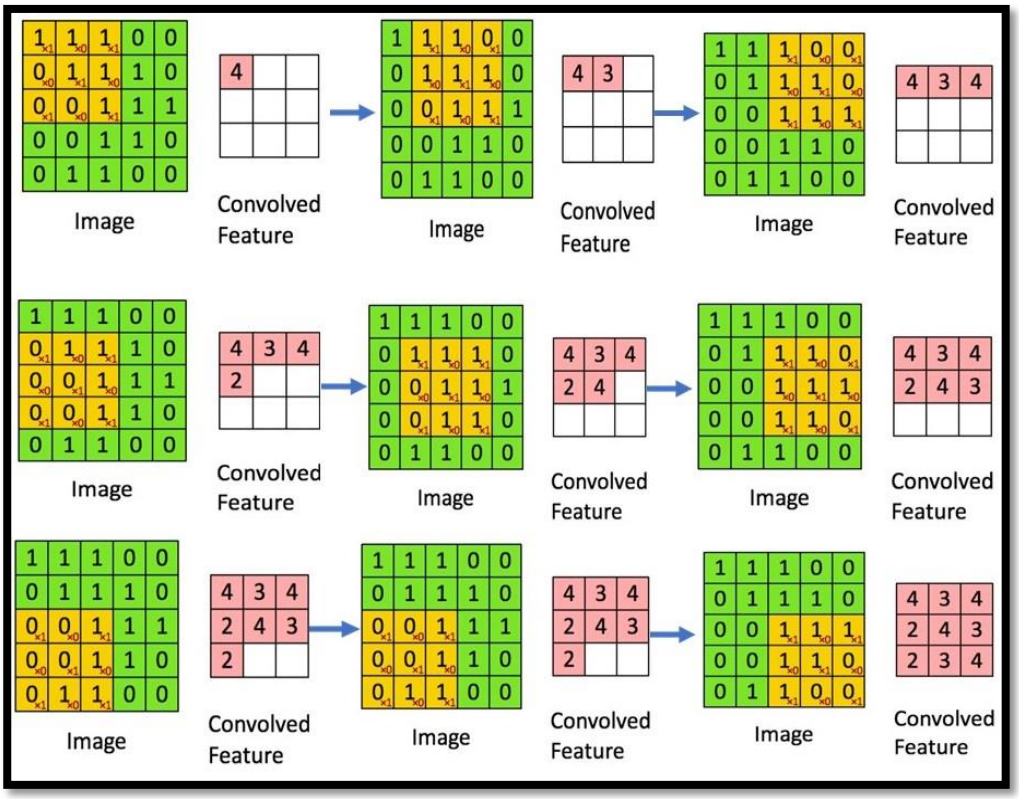


Figure (2.7): The convolution process [51].

The CNN architecture has many parameters that use to control the output size of the convolution layer, these parameters are called hyper parameters [36]. Some of the important CNNs' hyper parameters are as follows:

• Number Filters: Different reasonable numbers of filters can be used with different sizes.

• Filter size: Filter or kernel size must be squared L * L, smaller than the input image and greater than 2 × 2.

• Zero-padding: this is an efficient technique for controlling the dimensionality of output volumes by performing the process of filling at the input boundary.

• Stride: number of cells (pixels) to move the kernel through or down at the same time. If the stride is too small, then the receptive field will. Equation (2.7) calculated the output size of the matrix.

$$W_{out} = \frac{W - F + 2P}{S} + 1 \quad (2.7)$$

Where:

$W_{out}$: Output size.

$W$: Input size.

$F$: Filter size.

$S$: Number of stride.

$P$: Number of padding

### 2.7.2.2 Activation Layer ( Non-Linear Layer )

Non-Linear layers apply many functions such as ReLUs function, sigmoid function, and etc.. These functions are employed to determine the distinct features of each hidden layer. The feature maps output from the convolution layers becomes an input to non-linear layers to convert the linear output into non-linear [50].

### 2.7.2.3 Pooling Layer

The objective of the Pooling Layer or subsampling layer is the dimensional reduction of the input feature map to generate a reduced dimension output feature map by keeping only the most important information. There are two ways followed by this layer to reduce the

dimensionality which are: max pooling and average pooling [50] [52]. The Max pooling operation selects the maximum value from clustering neurons values, and the average pooling operation the selects average of the clustering neurons' values then averaging fraction rounds to the nearest integer in each region as shown in figure (2.8).

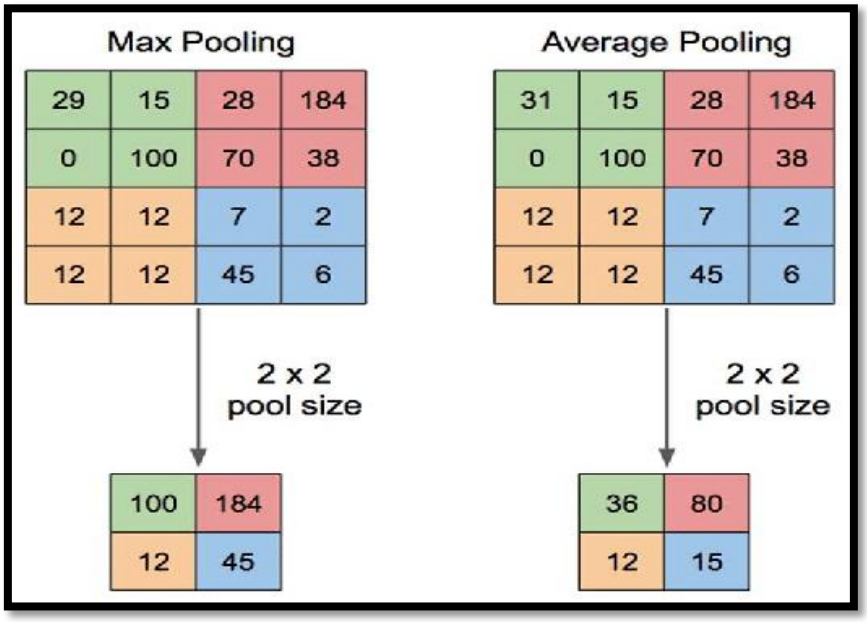


Figure (2.8): The max pooling and average pooling with a 4 × 4 cluster size [53].

### 2.7.2.4 Flattening Layer

In the flattening layer, the transform an array of certain dimensions into a one-dimensional array, until it is fed into the next layer [54]. Figure (2.9) illustrates the flattening process.

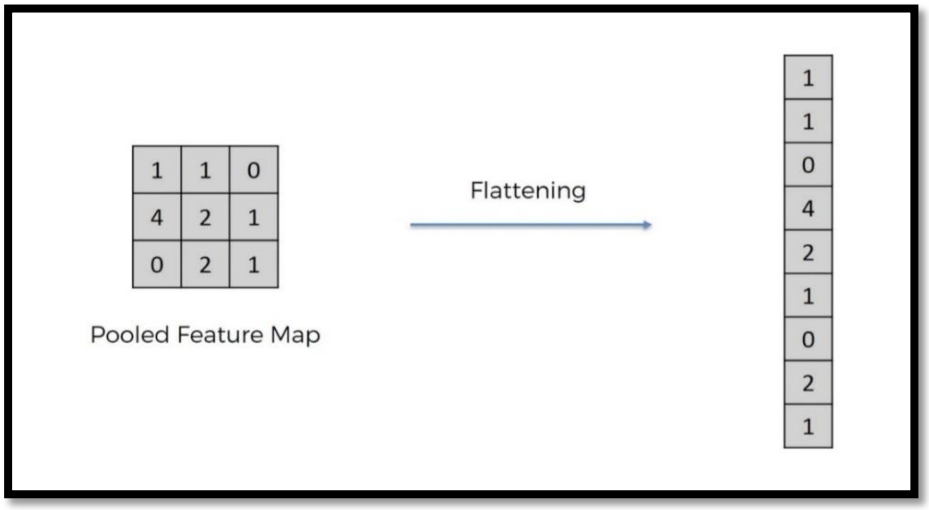


Figure (2.9): Flattening process [55].

### 2.7.2.5 Fully Connected Layers

The fully connected layers are the last layers of CNN, and use to perform the prediction phase of CNN. These layers can be represented as a multi-layer neural network that receives the output feature map of previous layers as input. The resulting feature map from the last previous layer which is 3D in shape (width, height, depth) must be flattened firstly to be 1D vector shape before input to the Fully-Connected Layers. The last layer in fully-connected layers is called the output layer, this layer is responsible for making a decision and predicting the output values [50]. Figure (2.10) shows the general perception of a fully connected layer.

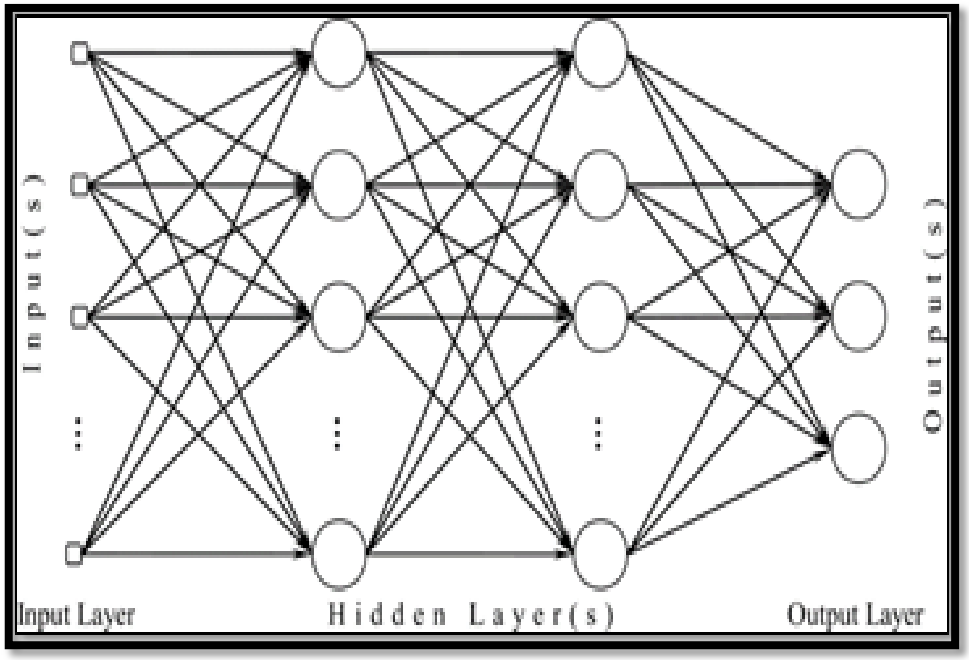


Figure (2.10): General perception for Fully Connected Layer [56].

## 2.8 Optical Neural Network

Recently, it has been demonstrated that linear light-wave diffraction, interference, and ONNs built on nan photonic circuits are effective for machine learning, but deep networks still lack non-linear optical activation functions. Although non-linear optical activation functions have been suggested for use, their experimental implementation has become a barrier to the further extension of ONNs in practical applications [57].

### 2.8.1 Spatial Light Modulator

In the 1980s, optical neural networks became well-known when Hopfield proposed their mathematical model in 1982. Emerging optical technologies at the time such as optical bistability were capable of performing

the threshold operations required for the fulfillment of this idea. Spatial Light Modulator (SLM) is a

device that can modify the beam and the spatial distribution of light waves in real-time. It is a crucial component of optical computing, real-time optical data processing, and other technologies. After that, we'll demonstrate how to design an ONN and create optical communication using SLM [30].

With the help of SLM and Fourier lenses, IT shows how to create a linear process using an AONN with customized linear operations. The intensity of incident light in various SLM regions indicates distinct input nodes during the linear operation phase. All diffracted beams are superimposed by SLM and Fourier lenses in the same direction, and they all converge to that location on the focus plane to perform linear summing. Through the use of an iterative feedback mechanism, AONN can generate linear matrix elements and achieve a reasonable level of precision [30].

Show a fully functional AONN, non-linear optical activation functions implemented in laser-cooled atoms with Electromagnetically Induced Transparency (EIT), linear operations controlled by SLM and Fourier lenses. Due to the independence of all mistakes from distinct optical neurons, an AONN can be scaled up in size. Furthermore, without modifying the physical structure, their hardware solution may be reconfigured for new purposes. A statistical Ising model's order and disorder phases can be successfully identified; they demonstrate its competence and viability in machine learning applications [30].

## 2.8.2 General All-Optical Neural Network Structure

As illustrated in figure (2.11.a), the neurons in a typical ANN are usually organized in layered patterns without links between neurons of the same layer. The neurons in the following layer receive input from the neurons in the previous layer [57].

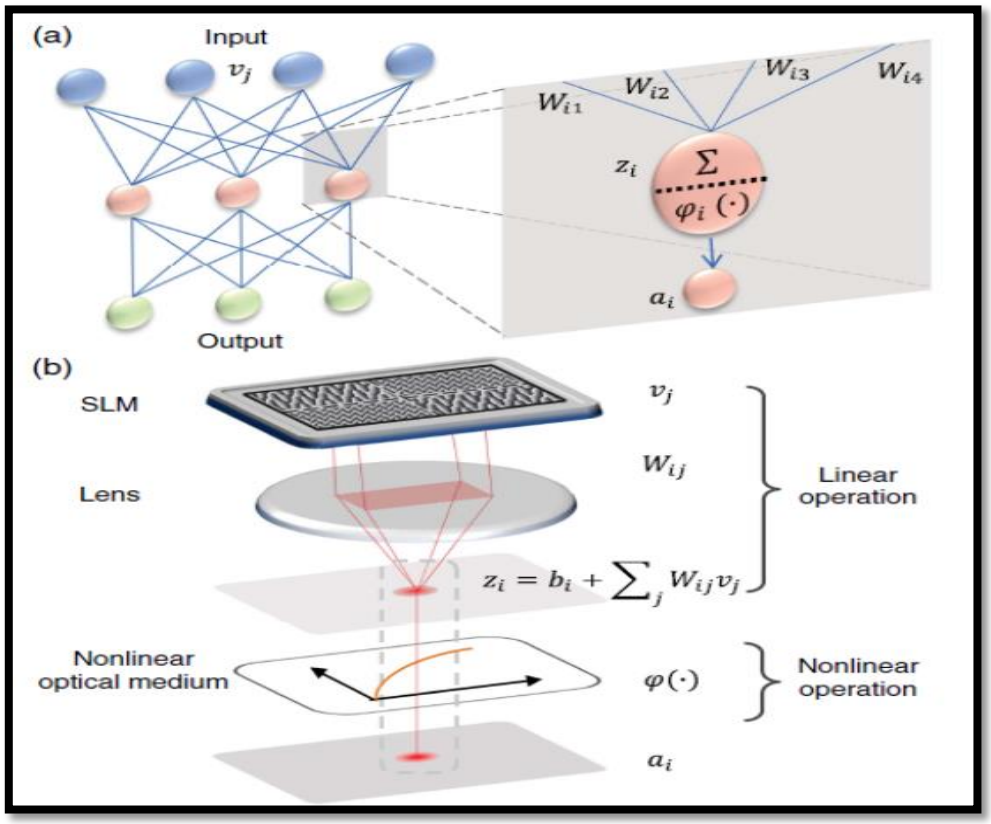


Figure (2.11): Generally, AONN. (a) A usual two layer neural network. (b) Diagram of investigational realization of an optical neuron involving non-linear and linear processes [57].

The following two steps can be used to abstract the artificial neuron working standard:

1. Received numerously weighted $w_{ij}$ input signals $v_j$ from neurons in the previous layers using a linear procedure with certain bias $b_i$.

$$z_i = b_i + \sum_j w_{ij}.v_j \quad (2.8)$$

2. Non-linear activation functions are used to create a new output signal while analyzing all of the input signals $a_i = \Phi(z_i)$.

Their optical setup uses EIT as demonstrated in figure (2.11.b), to realize the non-linear optical activation function while a SLM accompanied by a Fourier lens realizes the linear operations. Since the signals in the AONN are represented in light power as opposed to the transverse electric neurons seen in typical diffractive ONNs, $v_i$, $z_i$ ,$a_i \geq 0$, and the actual matrices components fulfill $1 \geq w_{ij} \geq 0$ [57].

### 2.8.3 Linear Optical Operation

The incident light intensities at various locations in the SLM will represent the input layer nodes $v_j$ in the linear operation process. By superimposing various phase grating, the incident light beams $v_i$ may be separated into a variety of directions $j$ with weight $w_{ij}$. As illustrated in figure (2.11.b), the SLM was positioned at the lens back focal plane, performing a Fourier transforms and summing all diffracted beams into a point at the front image plane as the linear summing $z_i = b_i + w_{ij}.v_j$. The linear biases $b_i$ might be obtained using added inputs. The given matrix elements $w_{ij}$ are acquired using the Gerchberg-Saxton increasing with time technique [57].

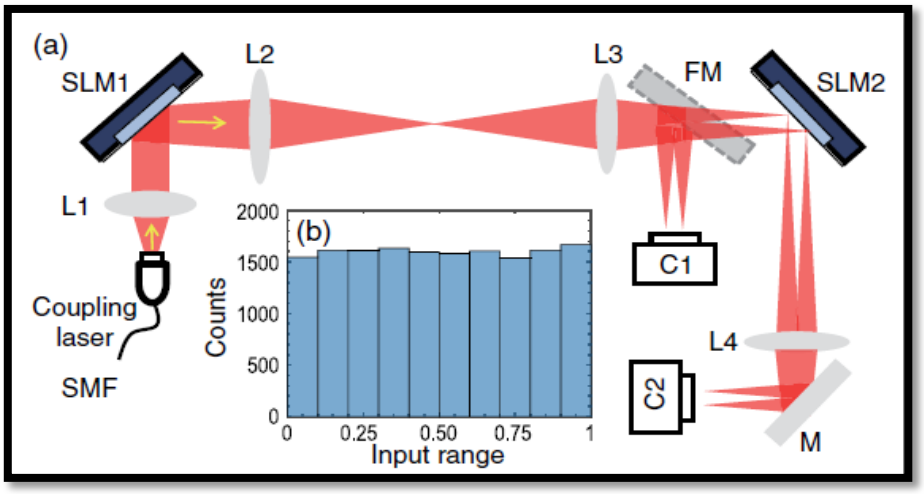


Figure (2.12): Optical setup, linear optical operation, and characterization. Single-mode fiber (SMF), flip mirror (FM), mirror (M), optical lens (L1–L4), and camera (C1–C2) [57].

The optics setup for carrying out the linear operations is illustrated in figure (2.12). Without compromising generalization, the researchers use 8-to-4 linear operations as illustrated. The first SLM (SLM1) is an incident with a collimated single-mode fiber (SMF) coupling laser beam that selectively reflects eight different beam points. These eight points are projected onto the second SLM (SLM2) as the input $v_j$ via a 4-f optical lens arrangement (L2 and L3). $v_j$ is measured and monitored using the first camera (C1) and flip mirror (FM). At the Fourier plane of lens L2, stray coupling light is prevented. Each laser beam is split into four different beams following SLM2. The summing process is carried out via the Fourier lens L4, while the second camera records the four output points (C2) [57].

### 2.8.4 Activation Functions

Activation functions are crucial components of neural networks because they introduce non-linearity. A neural network would be a linear regressed without activation functions. Before the input is transmitted to the output layer of neurons, it undergoes non-linear processing [39]. The output of the neural network is obtained using the activation functions. There are many types of activation functions available that are used depending on the problem to be solved. Generally, activation functions are divided into two types: linear and non-linear activation functions. The linear activation function is used in forward-propagation neural networks only, and to solve simple problems that can be represented linearly. One of the most important disadvantages of linear activation functions cannot be used in back-propagation networks because the derivative of the function is constant [28]. A non-linearity is a significant aspect that aims to get it in a multi-layer neural network to make non-linear. The importance of the non-linear activation function in a neural network can be focused on the available data that cannot be separated in a linear format [58]. The most significant non-linear activation functions used in neural networks are sigmoid, ReLU, tanh, and softmax.

#### 2.8.4.1 Sigmoid

Sigmoid is a common non-linear activation function. It has an S-shape and transforms data in the range of 0 to 1 [39]. The mathematical expression for sigmoid in equation (2.9) is:

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \tag{2.9}$$

### 2.8.4.2 Rectified Linear Unit

Rectified Linear Unit (ReLU) is a non-linear activation function and is often used in deep neural networks. This function returns zero if it received a negative value, but any positive value returns the same received value [39]. The mathematical expression of ReLU is presented in equation (2.10) is:

$$ReLU = \max(0, x) \tag{2.10}$$

### 2.8.4.3 Hyperbolic Tangent

Hyperbolic Tangent (Tanh) is s-shaped, so it is similar to the logistic sigmoid. The Tanh function has a range between -1 to 1 always [39]. The mathematical expression of Tanh in equation (2.11) is:

$$\text{Tanh}(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2.11}$$

### 2.8.4.4 Softmax Activation Function

The softmax activation function is non-linear activation function and often applied in the last layer in the model to convert the network output into a probability distribution. The softmax activation function has great benefit when used in classification problems, in particular when using multi-class classification problems because the output of the function has a probability distribution between 0 and 1, so the softmax returns the probabilities of each category, and the target category has the highest probability value [28]. The mathematical expression of softmax in equation (2.12) is:

$$p(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{2.12}$$

$p(z)_i$: The network output's softmax probability distribution.

z: Input vector.

k: Number of classes.

### 2.8.5 Loss Function

Early neural network models calculated the difference between the actual output and the expected output which was used to determine the error. At the moment, several formulas have appeared for calculating errors in neural networks, these formulas are called Loss Functions [59] [60]. When using various loss functions, this can lead to a different error values for the same prediction, therefore the type of loss function has a major impact on the output of the network. There are three main types of loss functions: Classification Loss Functions, Regression Loss, and Embedding Loss Functions [37]. Classification loss functions are used with classification problems. The Regression Loss functions are used in regression problems when the output variables are continuous. While the Embedding loss functions are used with tasks that need to measure the similarity between two inputs. The following are some types of loss functions.

#### 2.8.5.1 Mean Square Error

Mean square error (MSE) is one of the important regression loss functions which measures the square difference between the real and expected value [61]. The following equation (2.13) illustrates the MSE.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2 \quad (2.13)$$

Where:

$y_i$: Actual target vector.

$p_i$: The output predicted vector.

$n$: vector length.

### 2.8.5.2 Mean Absolute Error

Mean Absolute Error (MAE) is somewhat distinct in terms of definition, from MSE which measures the absolute value of the difference between model predictions and the actual data [62]. The mathematical expression of the MAE in equation (2.14) :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|z_i^{(L)} - t_i| \quad (2.14)$$

Where:

$z_i$: Actual target vector.

$t_i$: The output predicted vector.

$n$: vector length.

### 2.8.5.3 Categorical Cross Entropy

This loss function is used in Multi-class classification problems. In these problems, the target can only belong to one out of many possible categories, and the model must decide which one. This function is designed to calculate the difference between two probability distributions; therefore, this function is often used in networks that use the softmax activation function [63]. It is calculated using the following equation (2.15):

$$CCE = -\sum_{i} t_i \log(p_i) \tag{2.15}$$

Where:

$t_i$: Target value for the $i-th$ output neuron.

$p_i$: The network output's softmax probability distribution, as shown in equation (2.12).

## 2.8.6 Implementing Optical Back-propagation

Forward-propagation translates neuron activations from layer $l-1$ to neuron inputs at layer $l$ as a function of data seeded at the input layer ($a^{(0)}$) as in equation (2.16).

$$z_j^{(l)} = \sum_i w_{ji}^{(l)} a_i^{(l-1)} \tag{2.16}$$

Before each neuron receiving a non-linear activation function, $a_j^{(l)} = g(z_j^{(l)})$ using a weight matrix $w^{(l)}$ (with subscripts labeling individual neurons). Equation (2.17) illustrates how the researchers assess the loss function $\mathcal{L}$ and calculate, its gradient about the weights at the output layer [4].

$$\frac{\partial \mathcal{L}}{\partial \mathcal{L}_{ji}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)} \tag{2.17}$$

Where $\delta_j^{(l)} \equiv \partial \mathcal{L} / \partial z_j^{(l)}$ also known as the error at the $j-th$ neuron in the $l-th$ layer. Using the chain rule :

$$\delta_j^{(l)} = \sum_k \frac{\partial \mathcal{L}}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} = g'(z_j^{(l)}) p_j^{(l+1)} \tag{2.18}$$

Where $p_j^{(l+1)} = \sum_k \delta_k^{(l+1)} w_{kj}^{(l+1)}$. Given the error at the output layer, i.e. $\delta^{(L)}$, which is calculated directly from the loss function, the errors $\delta^{(L-1)}, \ldots, \delta^{(1)}$ for all preceding layers are sequentially found using equation

(2.18). These errors, as well as the activation $a^{(l-1)}$ of all neurons, allow to calculate the gradients equation (2.17) of the error function for all weights, and it is followed by the usage of gradients descent [4].

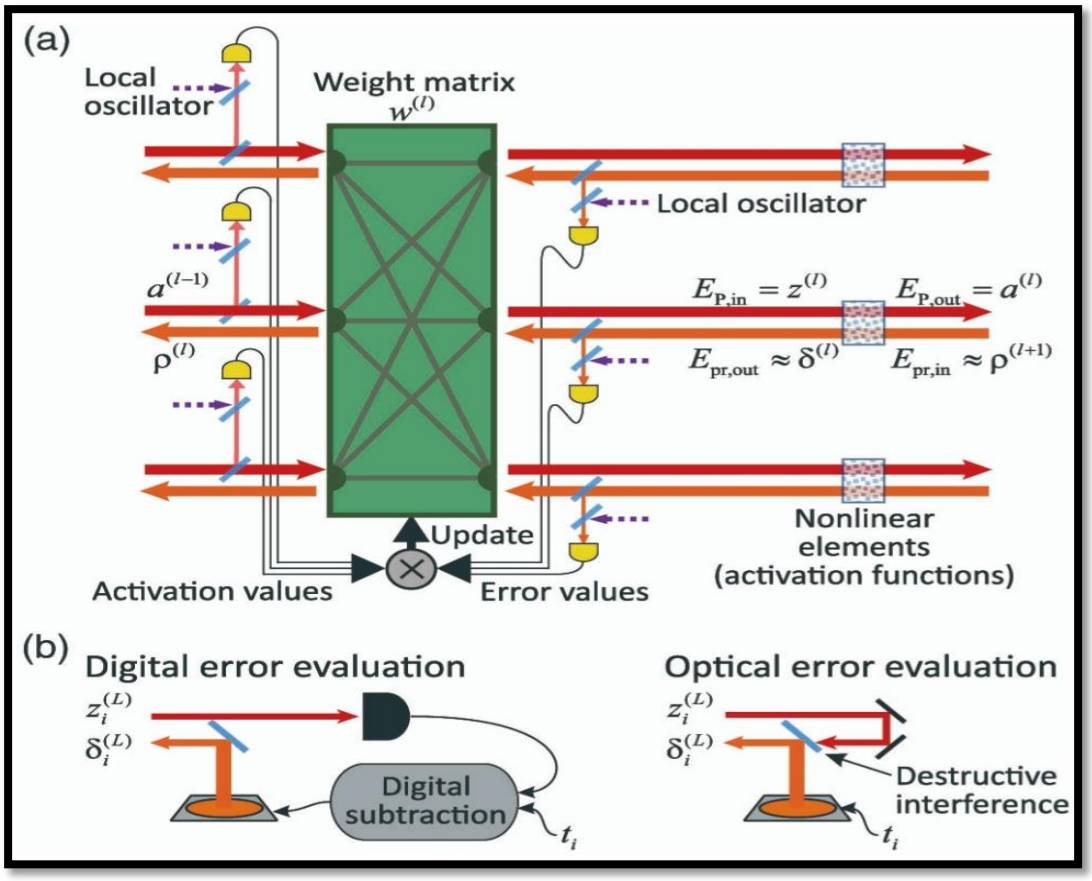


Figure (2.13): ONN with all-optical forward and backward-propagation. (a) A single ONN layer, and the red color refers to forward and the orange dashed refers to backward-propagation. (b) Error computation carried done optically or digitally at the output layer [4].

For this system, the neurons are field amplitudes with real values in various spatial modes, and the transformation equation (2.16) is simply realized as a linear optical (interferometric) process. Calculating $p^{(l+1)}$ in the back-propagation equation (2.18) right side employs at the same weight matrix, enabling it to function by physically optical signal backward

propagation using the linear optical agreement, as illustrated in figure (2.13.a). Yet, without using digital electronics, by multiplying that signal by the activation function's derivative, $g'\left(z^{(l)}\right)$ is difficult. This problem requires an optical activation function implementation with these characteristics: (i) a non-linear response to input coming from the front; (ii) a linear response to input coming from the back; (iii) modulation of backward input using non-linear function derivative while using non-linear optics for this purpose is natural which is a challenging criterion to achieve, the unit must react differentially to light that travels forward and backward. Here, they demonstrate how the well-known pump-probe arrangement can be used to solve this issue via saturable absorption. Take a look at how a two-level medium might react to the passage of a strong pump $E_P$ and a weak probe $E_{Pr}$ (e.g. atomic vapor). According to equation (2.19), a non-linear function of the input is the pump transmission [4].

$$E_{p,out} = g\left(E_{p,in}\right) = \exp\left(-\frac{\alpha_0/2}{1+p_{p,in}^2}\right)E_{p,in} \tag{2.19}$$

Where $\alpha_0$ is the resonant optical depth, the saturation threshold is considered to equalize all fields. The pump transmission g (.) is plotted in figure (2.14.a) at $\alpha_0$ of 1 and 30. In the unsaturated area, high optical depth produces substantial non-linearity, whereas, a strong enough pump causes the medium to become nearly transparent in the saturated area. However, because the pump does not alter the atomic media's transmittance, when the probe is sufficiently weak, the pump's absorption coefficient which is specified in equation (2.18), which is linearly absorbed [4].

$$\frac{E_{pr,out}}{E_{pr,in}} = \exp\left(-\frac{\alpha_0}{1+E_{p,in}^2}\right) \tag{2.20}$$

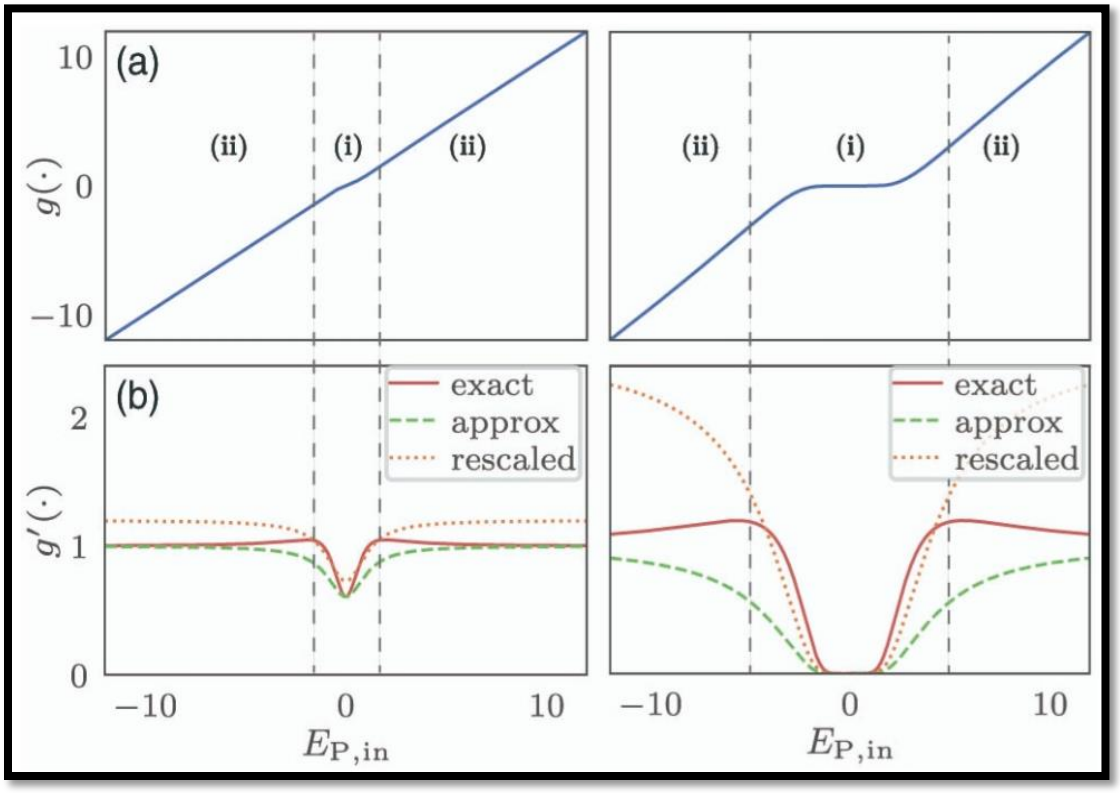


Figure (2.14): Response of a saturable absorber. (a) The transmission derivative of a SA unit with optical depths of 1 on the left and 30 on the right. In panel (b), these also approximate the derivatives with and without rescaling, which are the real probe transmissions from equation (20) [4].

Because both beams are in resonance with the atomic transition, they may represent as real-valued without losing generality because the phase of the electric field remains identical. The pump and probe operate as the forward-propagating signal and back-propagating error in an ONN, respectively, satisfying the necessary features (i) and (ii) of our optical non-linear unit. However, condition (iii) still needs to be satisfied. The pump transmission's derivative, as illustrated in equation (2.21) exact [4]:

$$g'(E_{p,in}) = \left[1 + \frac{\alpha_0 E_{p,in}^2}{(1+E_{p,in}^2)^2}\right] \exp\left(-\frac{\alpha_0/2}{1+E_{p,in}^2}\right) \tag{2.21}$$

Figure (2.14.b) illustrates the derivatives at $\alpha_0$ of 1 and 30. The most significant finding, they made was that the square-bracketed factor in equation (2.21) can frequently be interpreted as constant, in which case the back-propagation transmission of equation (2.20) is a decent approximation of the desired response in equation (2.21) up to a constant factor. The learning rate is absorbed in a steady scaling of the network gradients, satisfying Feature (iii)[4].

The suggested technique can be used for integrated or free space platforms. On-chip SA has been shown in an integrated environment using the atomic vapor and other non-linear media, in additionally to optical interference units on-chip, which give intra-layer weights by integrating phase-shifters and attenuators. A conventional atomic vapor cell's non-linear unit can be used by a SLM to execute the requisite matrix multiplication in free space. The weight gradients in equation (2.18) must be mapped to the proper updates of the control factors (phase-shifters and attenuators) in the integrated case, which requires a further nontrivial step; nevertheless, this problem was recently handled. On the other hand, the update calculation is simplified in the free-space version because discrete blocks of SLM pixels are used to regulate individual weights. No matter the platform, weighted interconnections with requirements for energy conservation could only be implemented by passive optical elements. The weight matrices may be quickly produced with normalized weights by scaling the input layer neuron activations appropriately, so this is not a practical barrier in networks with a single layer of non-linear activations. Absorption via the vapor cell reduces the available field amplitude for succeeding layers in deep networks with

several layers. Interlayer amplification utilizing semiconductor optical amplifiers, for example, can be used to mitigate this [4].

The only components of the proposed ONN that require electronics are (a) real homogeneous or heterogeneous measurements of exploited neuron activations $(a^{(l)})$ and error terms $(\delta^{(l)})$ in each layer, (b) creation of network inputs and reference packets, with along (c) update weights. The speed of this process is not essential to the performance of the ONN because the update (c) is computed as an average over multiple elements rather than for each individual training set element (a "mini-batch" or training period). Since the targets and input generation are different from the computation performed by the ONN, it calls for quick optical modifiers, which are widely accessible on the market. Finally, after taking the measurements (a), it figures out the product $\delta_j^{(l)} a_i^{(l-1)}$ and averages it throughout the mini-batch. Electronic gate arrays can be used to carry out this procedure. For a network with $L$ layers of $N$ neurons, this requires $2LN$ measurements and $LN^2$ offline multiplications. The two signals can also be directly optically interfered with, and the intensity of the interference can then be used to multiply them. The optical multiplication will necessitate setup phase stability and the added cost of $2LN^2$ photodetectors. However, it would do away with the requirement for offline multiplications and reference beams. These detectors' data will also require $LN^2$ procedures in order to calculate the new weight matrices which will have to be conducted once every epoch. Because the modulators refresh rate is limited, these actions may cause a performance bottleneck. Despite the frequency degeneracy of the activation and error signals in their scheme, their counter-propagating configuration allows them to be clearly distinguished at the detection stage. Moreover, the two's relative phase counter-propagating

signals will not affect the non-linear unit's operation. The bandwidths of the intra-layer amplifiers and SAs cause the majority of the latencies associated with signal optical propagation in the ONN. The photo detection and multiplication of $\delta_j^{(l)} a_i^{(l-1)}$, all have additional processing performance constraints, as does the transmission of computed weight matrix gradients to matching actuators inside the ONN. This restriction can be avoided by using large batches, though, as the latter conversion only occurs once every training batch. Once the error $\delta^{(L)}$ in the output layer has been calculated and re-subtracted, the back-propagation phase of the ONN training process is initiated. The mean square error loss function is used to train the ONN, as illustrated in equation (2.14). On a balanced beam splitter, this can be computed by measuring the interference between the target and network outputs. Figure (2.14.b) illustrates this method of error calculation in the right panel, while the left panel displays the conventional method, which involves computing the errors offline (electronically) [4].

### 2.8.7 Optimization Function

The weights of the neural network are updated by a function called Optimization Function. During the training process, the optimization functions change the weights of the network and try to reach the lowest value of the loss function and create an output of the model correct as possible. The optimizer associates the loss function with network parameters by updating the weight's network according to the output of the loss function. The optimization algorithms typically measure the gradient, i.e. the partial loss function derivative concerning weights, and the weights are adjusted in the opposite direction of the measured gradient. This cycle is repeated until the model reaches a minimal loss of function is reached [60].

### 2.8.8 Type of Optimization Algorithms

One of the most crucial aspects of neural network optimization is selecting the optimization algorithm. There are three primary categories of optimization techniques [64]. The first is referred to as batch processing or deterministic gradient algorithms, which process all training samples concurrently in a sizable batch. The second type is referred to as stochastic or online approaches which only use one sample at a time. Nowadays, most algorithms combine the two. During training, they use only a part of the training set at each epoch. These algorithms are called mini-batch methods.

The optimization algorithms can be split into two main groups: Constant Learning Rates Algorithms such as SGD, and Adaptive Learning Algorithms [59] [65]. In the first group, the learning rate $\eta$ is chosen manually. The task of choosing the learning rate in this type of algorithm is somehow difficult. When choosing a relatively small learning rate, the learning process is slowed and the training time becomes too large. While choosing a relatively large learning rate, it can lead to fluctuation in the loss value around the minimum value, this hinders the convergence process. While the algorithms of the second group do not need to set the learning rate manually, they have a heuristic approach that takes care of adjusting the learning rate value. The algorithms of the second category are characterized by having a variable learning rate during the training process, in contrast to the algorithms of the first class. Therefore, several algorithms appeared which belong to these two categories, the most important of which are illustrated as follows:

### 2.8.8.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is one of the Constant Learning Rate Algorithms that tries to update the network parameters more repeatedly. In this algorithm, the network parameters are modified on each training sample after the loss calculation. So, if the dataset contains 900 samples, the network parameters will be modified 900 times in one cycle of the dataset. The SGD equation (2.22) that is used to update parameters in a neural network is illustrated as follows [65]:

$$W^{(K+1)} = W^{(K)} - \eta * \left(\Delta J(W)\right) \tag{2.22}$$

$W$: Network Parameter (weights, biases).

$\eta$: Learning Rate.

$\Delta J(W)$: Gradient.

### 2.8.8.2 Adam

Adam is one of the adaptive learning rate optimization algorithms, it measures individual learning rates for various parameters [65]. Adam sets the learning parameter automatically; this was done by using the first and second moment estimation. The moment is the expectation of a random_variable at the power of $n$ [66]. The moment can be illustrated in equation (2.23):

$$M_n = E[X^n] \tag{2.23}$$

Where:

$M$: The moment.

$X$: A random variable.

The following equations use to estimate the first and second moments of Adam.

$$M_t' = \frac{M_t}{1 - \beta_1} \tag{2.24}$$

$$V_t' = \frac{V_t}{1 - \beta_2} \tag{2.25}$$

Where:

$M_t$: The previous first moment.

$V_t$: The previous second moment and initialized with 0 in the first step.

$\beta_1, \beta_2$: Are new parameters, they have default values of 0.9 and 0.999 respectively.

After calculating the value of the first and second moments, the following equation (2.26) is used to update the network weights.

$$W_t = W_{t-1} - \eta \frac{M}{\sqrt{V_t'} + \epsilon} \tag{2.26}$$

Where:

$W$: Network Weights.

$\eta$: Learning Rate.

$\epsilon = 10^{-8}$.

### 2.8.9 Learning Rate

The learning rate is a tiny positive adjustable hyper parameter that controls how the model responds to the predicted error each time the model weights are updated. It is used in neural network training. It can be challenging to choose the learning rate since a value that is too little could lead to a drawn-out training process that becomes stuck, while a value that is too large could cause the training process to learn a suboptimal set of weights too rapidly or become unstable. The difficulty in training deep learning neural networks is that the learning rate must be properly chosen [67].

### 2.8.10 Evaluation Measures

Evaluating performance is an important aspect when designing a model. For the model to be reliable, the researchers must choose an evaluation tool commensurate with the nature of the model's work. Often, when evaluating models more than one scale is used to ensure the correct evaluation of the model. The evaluation measures are divided into three main types: the measures are used to evaluate classification tasks, the measures are used to evaluate regression tasks, and the measures are used to evaluate clustering tasks.

There are several kinds of evaluation measures are available for Classification tasks; such as Accuracy and loss.

### 2.8.10.1 Accuracy

Accuracy is a metric for the performance of a model. It is typically expressed as a percentage. The number of correct predictions over the total number of predictions is known as accuracy. Although the value is frequently linked to overall or end model accuracy, it is frequently graphed and tracked during the training phase. It can be expressed in equation (2.27) [68]:

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions} \times 100\% \qquad (2.27)$$

### 2.8.10.2 Loss

Loss is a value that symbolizes the total of the model errors. The loss will be considerable if the errors are high, indicating that the model does not perform well. The lower the loss is, the better model performs. Loss is more difficult to interpret than accuracy [69].

# CHAPTER THREE

# Proposed System

# CHAPTER THREE

# The Proposed System

## 3.1 Introduction

This chapter describes the main stages used to construct the proposed system. The proposed method involves several stages to be able to train an optical neural network through nonlinear units by a saturable absorption (SA) to perform image classification tasks for three types of data and compare it with artificial neural networks. This chapter has two parts, in the first part, the study presents the architecture of the proposed system at artificial neural networks. The second part presents the architecture of the proposed system at optical neural networks.

## 3.2 System Requirement

The implementation method requires high computing resources. So the proposed system was implemented using the following software requirements.

• Software

1. Operating System: Windows 10, 64 bit.

2. Programming Language: Python 3.7.3.

3. Development Environment: Pycharm 2022.1.

Neural networks are programmed in the Python program using some of the modern automatic differentiation libraries. These libraries were helpful to reduce processing time. Figure (3.1) shows the process of importing these libraries, pickle file is used to serialize and untie sequencing objects in Python,

matplotlib library to plot the results of the study, numpy library to load data into an array, Pytorch library is used to build an actual neural network from by introducing a library called torchvision, this library has some useful things like data loaders, datasets and data converters for raster images.

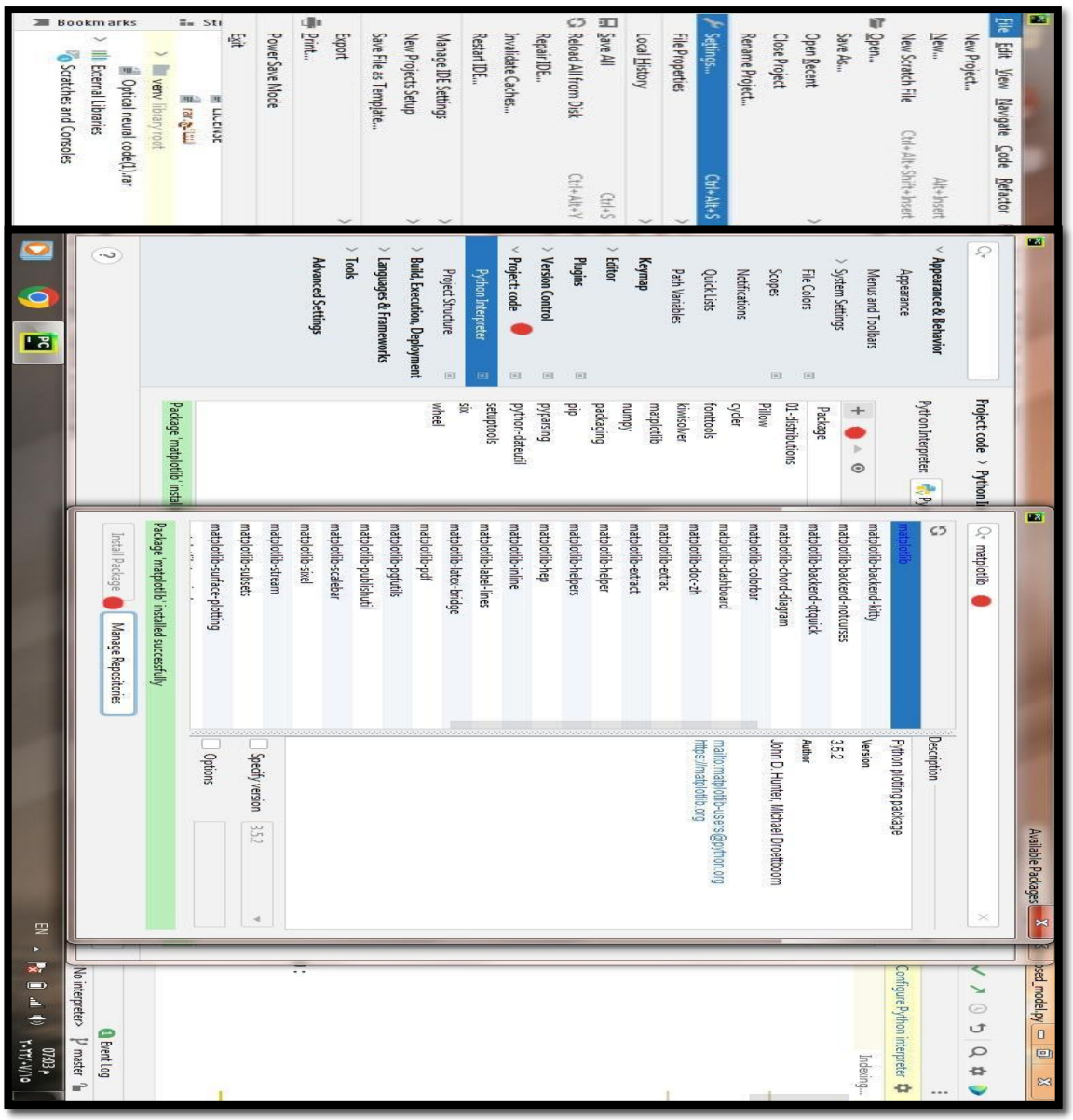


Figure (3.1): Method of import libraries.

## 3.3 Proposed Methodology

The proposed methodology involves four main parts, as follows: image dataset, reading mini-batch, network architecture, and training network. The whole proposed method can be illustrated in the following flow chart (3.2).

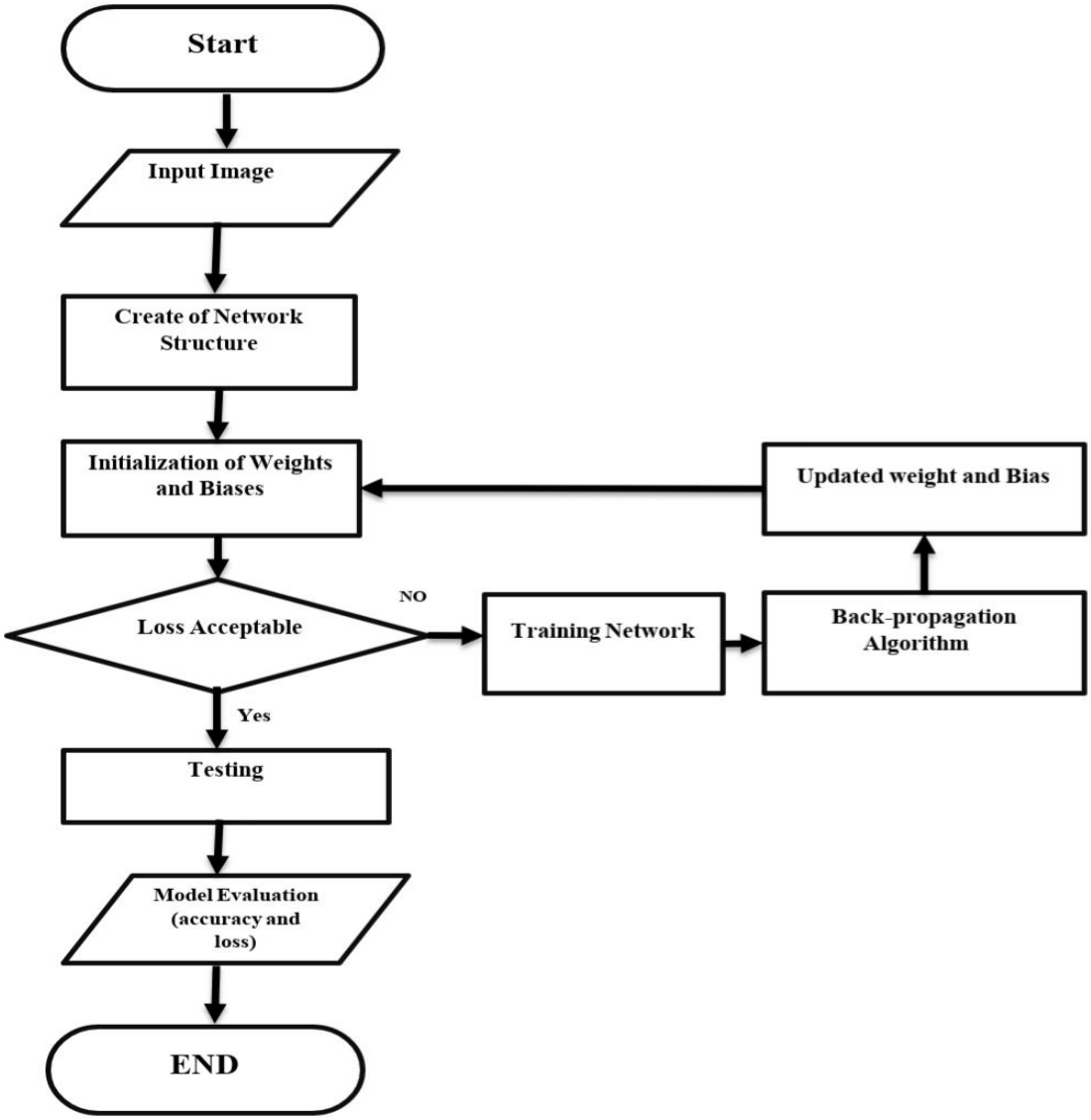


Figure (3.2): Process of the proposed system or methodology.

### 3.3.1 Image Dataset

Three different datasets were used: MNIST, KMNIST, and EMNIST, the all datasets contain images. MNIST and KMNIST contain a total of 70000 images, divided into 10000 testing cases and 60000 training cases. There are 18800 testing cases and 112800 training cases out of 131600 images in EMNIST. All categories are equally represented in both the testing and training sets for all datasets. The image dimensions (28 × 28) pixels (i.e. 784 digits) of gray scale. The image was represented in the computer in the form of a one-dimensional array (height × weight × depth) (i.e. (28 × 28 × 1)).

### 3.3.2 Reading Mini-Batch

To make it easier to read the training and test datasets, a mini-batch has been used. On each iteration, the data repeater reads a small batch of batch-size data at a time. In the case of training, batch_size_train = 64 sample for all data types (MNIST, KMNIST, and EMNIST). In the test case, batch-size-test = 128 sample for both MNIST, and KMNIST. In the case of EMNIST, the data is large, so batch_size_test = 1000 sample was used to get the testing process done faster.

### 3.3.3 Network Architecture

The network architecture has two parts: Convolution Neural Network/ Artificial Neural Network, and Convolution Neural Network/ Optical Neural Network.

### 3.3.3.1 Convolution Neural Network/ Artificial Neural Network

The CNN/ANN process includes several steps that introduced in detail, as illustrated in figure (3.3).

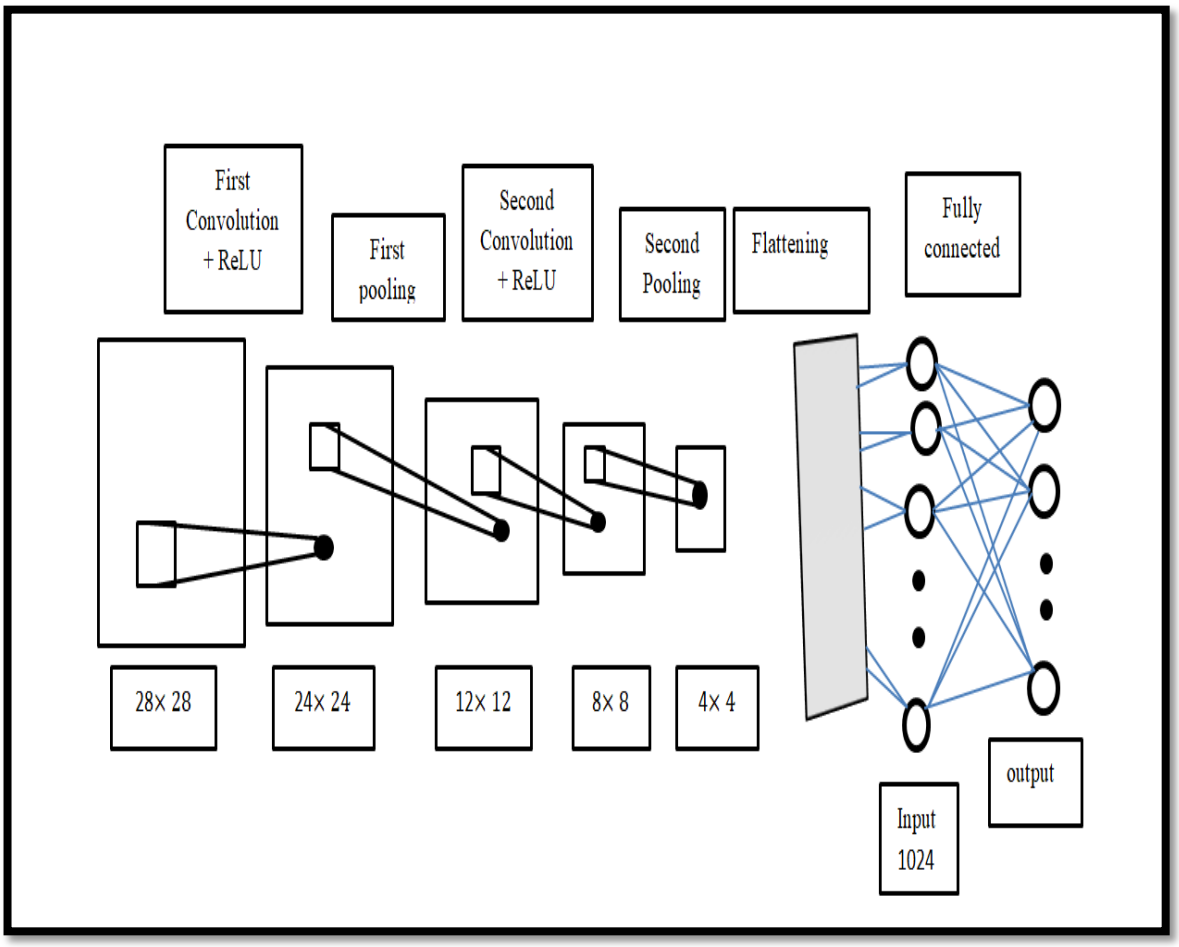


Figure (3.3): Architecture of CNN/ ANN.

## A. First Convolution Layer + ReLU

The first convolution layer contains a channel with size of 32; the layer wraps inputs with a filter. The size of this filter is (5 × 5) (i.e. the filter array size (5 × 5 × 1)), stride = 1, padding = 0, and the filter values are randomly chosen to be 0 or 1. The convolution process is carried out by positioning the filter array in the upper left corner of the input array, multiplying the values of

the filter array by the values of the input array, computing the results, and then moving the filter array to the right by one (i.e. moving by the amount of stride), repeating the multiplication operation and compute the result. The process repeats of moving the filter array over the input array until the filter array reaches the upper right corner of the input array. The filter array moves down by one (i.e. moving by the amount of stride) and repeats the process of applying the filter array from left to right until the entire input array is finished. After this process will get an array smaller in size (24 × 24 × 32) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size). Then the output data feeds to the ReLU activation function.

## B. First Pooling Layer

The channel size in the first pooling layer is 32. This layer includes a filter size (2 × 2) (i.e. a filter array size (2 × 2 × 1)), stride = 2, padding = 0, and used max pooling. The convolution process occurs by putting the filter array in the array's upper left corner (24 × 24 × 32), and taking the largest number, then moving the filter array to the right by two steps (i.e. moving by the amount of stride) and also taking the largest number and repeating this process until the filter array reaches the upper right corner of array (24 × 24 × 32). Then the filter array moves down by two steps (i.e. moving by the amount of stride) and repeats the process of applying the filter array from left to right until the entire array (24 × 24 × 32) is finished. After this process get an array smaller in size (12 ×12 × 32) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size).

## C. Second Convolution Layer + ReLU

The second convolution layer contains a channel with size of 64; the layer wraps inputs with a filter. The size of this filter is (5 × 5) (i.e. a filter array size (5 × 5 × 1)), stride = 1, padding = 0, and the filter values are randomly chosen to be 0 or 1. The convolution process is done by placing the filter array in the upper left corner of the array (12 × 12 × 32), multiplying the values of the filter array by the values of the array (12 × 12× 32), computing the results, and then moving the filter array to the right by one (i.e. moving by the amount of stride), repeating the multiplication operation and compute the result. The process repeats of moving the filter array over the array (12 × 12 × 32) until the filter array reaches the upper right corner of the array (12 × 12 × 32). The filter array moves down by one (i.e. moving by the amount of stride) and repeats the process of applying the filter from left to right until the entire array (12 × 12 × 32) is finished. After this process will get an array smaller in size (8 × 8 × 64) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size). Then the output data feeds to the ReLU activation function.

## D. Second Pooling Layer

The channel size in the second pooling layer is 64. This layer includes a filter size (2 × 2) (i.e. a filter array size (2 × 2 × 1)), stride = 2, padding = 0 , and used max pooling. The convolution process occurs by putting the filter array in the array's upper left corner (8 × 8 × 64) and taking the largest number, then moving the filter array to the right by two steps (i.e. moving by the amount of stride) and also taking the largest number and repeating this process until the filter array reaches the upper right corner of the array (8 × 8 ×

64). The filter array moves down by two steps (i.e. moving by the amount of stride) and repeats the process of applying the filter array from left to right until the entire array (8 × 8 × 64) is finished. After this process will get an array smaller in size (4 × 4 × 64) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size).

## E. Flattening Layer

The flattening layer converts the dimensional array (4 × 4 × 64) into a one-dimensional array (1 × 1024).

## F. Fully Connected Layer

The ANN for the datasets is developed into three layers (input layer, hidden layer, and output layer). The input layer will take the input signal to be processed and work weighted signal, there are three hidden layers (512, 256, and 128) between the input layer and the output layer that does all the arithmetic, the output of each layer is an input to the next layer, and the output layer does the job required for classification, this layer contains 10 neurons in the case of MNIST and KMNIST, these numbers represent the category labels or the desired output. It is a fully connected network to perform the input image classification. The weights values between 0 and 1 and bias = 1 are added to the layers. The forward propagation process takes place; the input layer passes the input 1024 neurons to the first hidden layer. The first hidden layer is where the calculations begin. It contains 512 neurons, each of which is fed by an input array. After calculating the result using equation (2.1). Each neuron generates an output that is fed into each neuron of the next layer (i.e. the second hidden layer is 256 and using equation (2.1) is repeated). The output is propagated to the third hidden layer of 128 neurons, and using equation (2.1)

is repeated. A non-linear activation function is applied to all outputs. The study used five types of non-linear activation functions: 1. SA (approximate) (using equation (2.20)). 2. SA (exact) (using equation (2.21)). 3. ReLU (using equation (2.10)). 4. Tanh (using equation (2.11)). 5. Sigmoid (using equation (2.9)). The output of the activation function is fed into the last layer which is the output layer. The softmax function as in equation (2.12) is used in the output layer to obtain outputs with probability ratios in the range 0 to 1 and the sum total of 1. The highest probability is selected for training example, and the rest other fewer probabilities are converted to 0. The error is calculated using the CCE loss function by equation (2.15). It is noticed that the output of loss function at the beginning is large, so the accuracy is low and the loss is high. The model has improved by model training and changing the parameters in a way that reduces the loss function. For neural networks, a common way to do this is by using back-propagation algorithm. The study will use Adam Optimizer which performs update using equation (2.26). Form training happens by repeating the process of scrolling forward and backward in the direction that reduces the loss function to the minimum value. The networks were trained on KMNIST for 40 epochs and MNIST for 30 epochs. With each epoch, the accuracy of the neural network may be increase and the loss decrease until the neural network reaches the stage of stability. After completing the training process, the test set was used to evaluate the performance of the model.

### 3.3.3.2 Convolution Neural Network/ Optical Neural Network

The CNN/ONN process includes several steps that introduced in detail, as illustrated in figure (3.4).

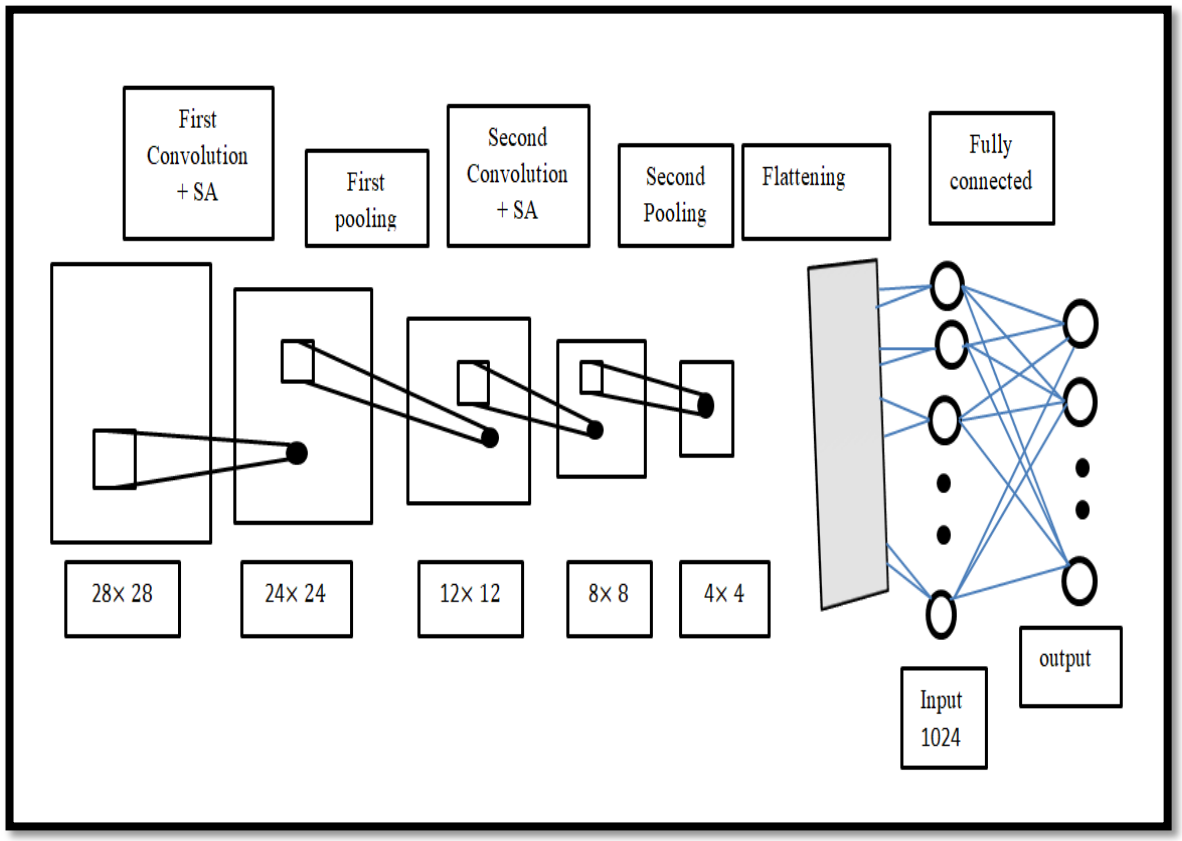


Figure (3.4): Architecture of CNN/ ONN.

## A. First Convolution Layer + SA

The first convolution layer contains a channel with size of 32; the layer wraps inputs with a filter. The size of this filter is (5 × 5) (i.e. a filter array size (5 × 5 × 1)), stride = 1, padding = 0, and the filter values are randomly chosen to be 0 or 1. The convolution process is carried out by positioning the filter array in the upper left corner of the input array, multiplying the values of the filter array by the values of the input array, computing the results, and then moving the filter array to the right by one (i.e. moving by the amount of

stride), repeating the multiplication operation and compute the result. The process repeats of moving the filter array over the input array until the filter array reaches the upper right corner of the input array. The filter array moves down by one (i.e. moving by the amount of stride) and repeats the process of applying the filter array from left to right until the entire input array is finished. After this process will get an array smaller in size (24 × 24 × 32) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size). Then the output data feeds to the SA activation function.

## B. First Pooling Layer

The channel size in the first pooling layer is 32. This layer includes a filter size (2 × 2) (i.e. a filter array size (2 × 2 × 1)), stride = 2, padding = 0, and used max pooling. The convolution process occurs by putting the filter array in the upper left corner of the array (24 × 24 × 32), and taking the largest number, then moving the filter array to the right by two steps (i.e. moving by the amount of stride) and also taking the largest number and repeating this process until the filter array reaches the upper right corner of the array (24 × 24 × 32). The filter array moves down by two steps (i.e. moving by the amount of stride) and repeat the process of applying the filter array from left to right until the entire array (24 × 24 × 32) is finished. After this process will get an array smaller in size (12 × 12 × 32) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size).

## C. Second Convolution Layer + SA

The second convolution layer contains a channel with size of 64; the layer wraps inputs with a filter. The size of this filter is (5 × 5) (i.e. a filter array size (5 × 5 × 1)), stride = 1, padding = 0, and the filter values are randomly chosen to be 0 or 1. The convolution is done by placing the filter array in the upper left corner of the array (12 × 12 × 32), multiplying the values of the filter array by the values of the input array, computing the results, and then moving the filter array to the right by one (i.e. moving by the amount of stride), repeating the multiplication operation and compute the result. The process repeats of moving the filter array over the array (12 × 12 × 32) until the filter array reaches the upper right corner of the array (12 × 12 × 32). The filter array moves down by one (i.e. moving by the amount of stride) and repeats the process of applying the filter array from left to right until the entire array (12 × 12 × 32) is finished. After this process will get an array smaller in size (8 × 8 × 64) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size). Then the output data feeds to the SA activation function.

## D. Second Pooling Layer

The channel size in the second pooling layer is 64. This layer includes a filter size (2 × 2) (i.e. a filter array size (2 × 2 × 1)), stride = 2, padding = 0, and used max pooling. The convolution process occurs by putting the filter array in the array's upper left corner (8 × 8 × 64), and taking the largest number, then moving the filter array to the right by two steps (i.e. moving by the amount of stride) and taking the largest number and repeating this process until the filter array reaches the upper right corner of the array (8 × 8 × 64).

The filter array moves down by two steps (i.e. moving by the amount of stride) and repeats the process of applying the filter array from left to right until the entire array (8 × 8 × 64) is finished. After this process will get an array smaller in size (4 × 4 × 64) by using equation (2.7) (i.e. the result will produce an image with better features and reduced size).

## E. Flattening Layer

The flattening layer converts the dimensional array (4 × 4 × 64) into a one-dimensional array (1 × 1024).

## F. Fully Connected Layer

The ONN for the datasets is developed into three layers (input layer, hidden layer, and output layer). The input layer will take the input signal to be processed and work weighted signal, there are three hidden layers (512, 256, and 128) neurons between the input layer and the output layer that does all the arithmetic, an output of each layer is being an input to the next layer, and the output layer does the job required for classification, this layer contains 10 neurons in the case of MNIST and KMNIST, and 47 neurons in the case of EMNIST, these numbers represent the category labels or the desired output. It is a fully connected network to perform the input image classification. The weights values between 0 and 1 and bias = 1 are added to the layers. The forward propagation process takes place; the input layer passes the input 1024 neurons to the first hidden layer. The first hidden layer is where the calculations begin. It contains 512 neurons, each of which is fed by an input array. After calculating the result using equation (2.1). Each neuron generates an output that is fed into each neuron of the next layer (i.e. the second hidden layer is 256 and using equation (2.1) is repeated). The output is propagated to

the third hidden layer of 128 neurons, and using equation (2.1) is repeated. A non-linear activation function is applied to all outputs. The study used five types of non-linear activation functions: 1. SA (approximate) (using equation (2.20)). 2. SA (exact) (using equation (2.20)). 3. ReLU (using equation (2.10)). 4. Tanh (using equation (2.11)). 5. Sigmoid (using equation (2.9)). The output of the activation function is fed into the last layer which is the output layer. The error is calculated using the mean absolute error loss function by equation (2.14). It is noticed that the output of loss function at the beginning is large, so the accuracy is low and the loss is high. The model has improved by model training and changing the parameters in a way that reduces the loss function. For neural networks, a common way to do this is by using back-propagation algorithm. The study will use Adam Optimize which performs update using equation (2.26). Form training happens by repeating the process of scrolling forward and backward in the direction that reduces the loss function to the minmum value. The networks were trained on (KMNIST and EMNIST) for 40 epochs and MNIST for 30 epochs. With each epoch, the accuracy of the neural network may increase and the loss decrease until the neural network reaches the stage of stability. After completing the training process, the test set was used to evaluate the performance of the model.

### 3.3.4 Training Network

The following steps are the structures of the training network.

1. The images would be supplied 64 (batch_size_train) times during one iteration from 60000, training records.
2. Utilizing forward propagating would next feed the batch_size_train into the network.

3. This would deliver the network output that represents the predicted class names. In addition to the actual values, the loss would be produced as a variable object by the loss function.
4. After that, use the back-propagation algorithm to calculate the update of the variables.
5. By using the optimizer's update function to change the model parameters since the optimizer already contains a reference to the network and access to the variables and calculated gradients.

In addition to the aforementioned actions, the test is carried out with the same training steps except that the usage batch_size_test = 128 for both MNIST, and KMNIST. Batch_size_test = 1000 in the case of EMNIST, and the back-propagation is not used.

# CHAPTER FOUR

# Results and Discussion

# CHAPTER FOUR

# Results and Discussion

## 4.1 Introduction

The results from the different stages of the training process for ANNs and ONNs are presented. In this chapter, the results are illustrated with graphics made in the python program. For a supplemental description of what was done in each step in chapter three.

## 4.2 Results and Discussion of Artificial Neural Networks

In this part, ANNS were implemented for the purpose of comparison only. The study used the accuracy and loss curves to determine network performance, the y-coordinate represents an accuracy value and a loss value, while the x-coordinate represents the number of epochs, and the system parameters are defined as follows:

1. Batch_Size_Train = 64. (In the case of KMNIST, and MNIST datasets).

2. Batch_Size_Test = 128. (In the case of KMNIST, and MNIST datasets).

3. Number of epochs = 40 (for KMNIST dataset), and Number of epochs = 30 (for MNIST dataset).

4. Seed 5 (i.e. five activation functions).

Seed 0 (indicates to SA (approx.) activation function).

Seed 1 (indicates to SA (exact) activation function).

Seed 2 (indicates to ReLU activation function).

Seed 3 (indicates to Tanh activation function).

Seed 4 (indicates to sigmoid activation function).

5. Learning Rate =$1 \times 10^{-4}$.

Table (4.1) illustrates the accuracy and loss values of the KMNIST and MNIST datasets.

Table (4.1): Results of ANN.

| Activation Function | KMNIST Dataset | MNIST Dataset |
|---|---|---|
| SA (approx.) | $(95.90 \pm 1)\,\%$ | $(99.14 \pm 1)\,\%$ |
| SA (exact) | $(95.64 \pm 1)\,\%$ | $(99.28 \pm 1)\,\%$ |
| ReLU | $(95.68 \pm 1)\,\%$ | $(99.22 \pm 1)\,\%$ |
| Tanh | $(96.26 \pm 1)\,\%$ | $(99.16 \pm 1)\,\%$ |
| Sigmoid | $(95.62 \pm 1)\,\%$ | $(99.30 \pm 1)\,\%$ |

Figure (4.1) illustrates the accuracy and loss curves of the KMNIST dataset.

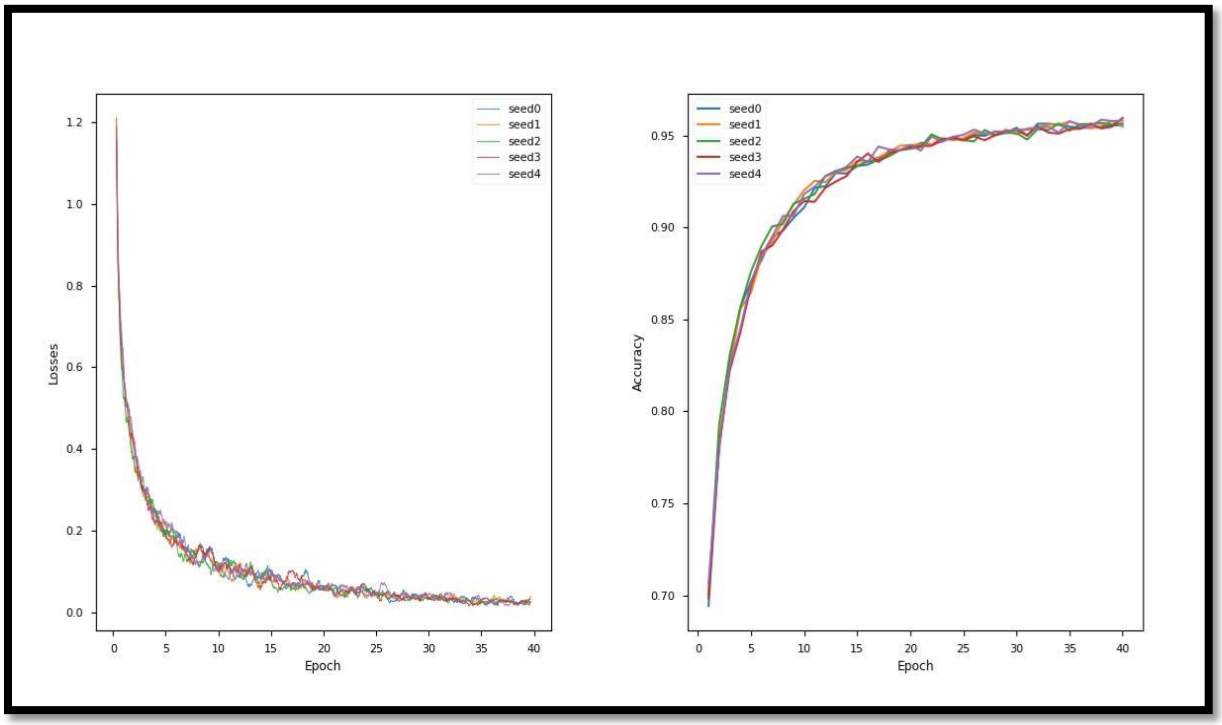


Figure (4.1): The accuracy and loss of the KMNIST dataset in ANN.

Figure (4.2) illustrates the accuracy and loss curves of the MNIST dataset.

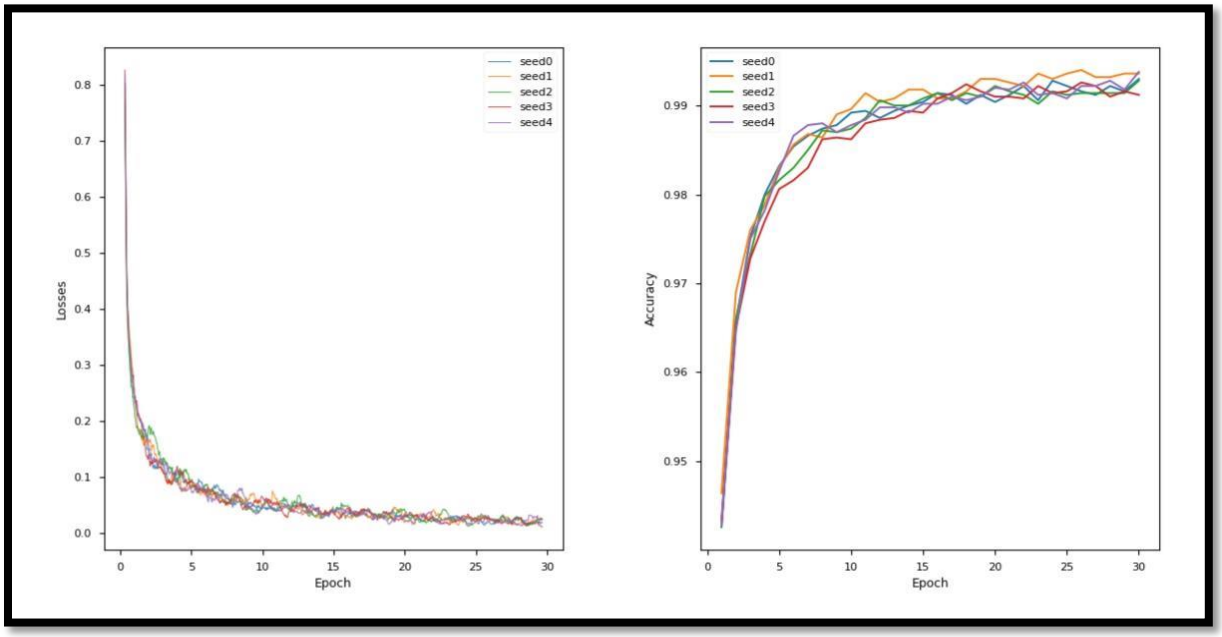


Figure (4.2): The accuracy and loss of the MNIST dataset in ANN.

Figure (4.1), figure (4.2), and the results in table (4.1) show that the accuracy of the system increases to almost optimum levels but the loss is high.

## 4.3 Results and Discussion of Optical Neural Networks

In this part, the study used the accuracy and loss curves to determine the network performance, the y-coordinate represents an accuracy value and a loss value, while the x-coordinate represents the number of epochs, and the system parameters are defined as follows:

1. Batch_Size_Train = 64. (In the case of MNIST, KMNIST, and EMNIST datasets).

2. Batch_Size_Test = 128. (In the case of MNIST, and KMNIST datasets), and Batch_Size_Test = 1000. (In the case of EMNIST dataset).

3. Number of epochs = 40 (for KMNIST, and EMNIST), and Number of epochs = 30 (for MNIST).

4. Seed 5 (i.e. five activation functions).

Seed 0 (indicates to SA (approx.) activation function).

Seed 1 (indicates to SA (exact) activation function).

Seed 2 (indicates to ReLU activation function).

Seed 3 (indicates to Tanh activation function).

Seed 4 (indicates to sigmoid activation function).

5. Choosing the learning rate is very difficult. So, the KMNIST dataset trained at different learning rates ($1 \times 10^{-3}, 1 \times 10^{-4}, \text{and } 1 \times 10^{-5}$). To find out which one fits the network best.

5.1 Learning rate $1 \times 10^{-3}$.

Table (4.2) illustrates the accuracy and loss values of the KMNIST dataset at learning rate $1 \times 10^{-3}$.

Table (4.2): Results of the KMNIST dataset in ONN (learning rate $1 \times 10^{-3}$).

| Activation Function | KMINST Dataset |
|---|---|
| SA (approx.) | $(97.04 \pm 0.1)$ % |
| SA (exact) | $(96.78 \pm 0.1)$ % |
| ReLU | $(97.12 \pm 0.1)$ % |
| Tanh | $(9.32 \pm 10)$ % |
| Sigmoid | $(96.60 \pm 0.1)$ % |

Figure (4.3) illustrates the accuracy and loss curves of the KMNIST dataset at learning rate $1 \times 10^{-3}$.

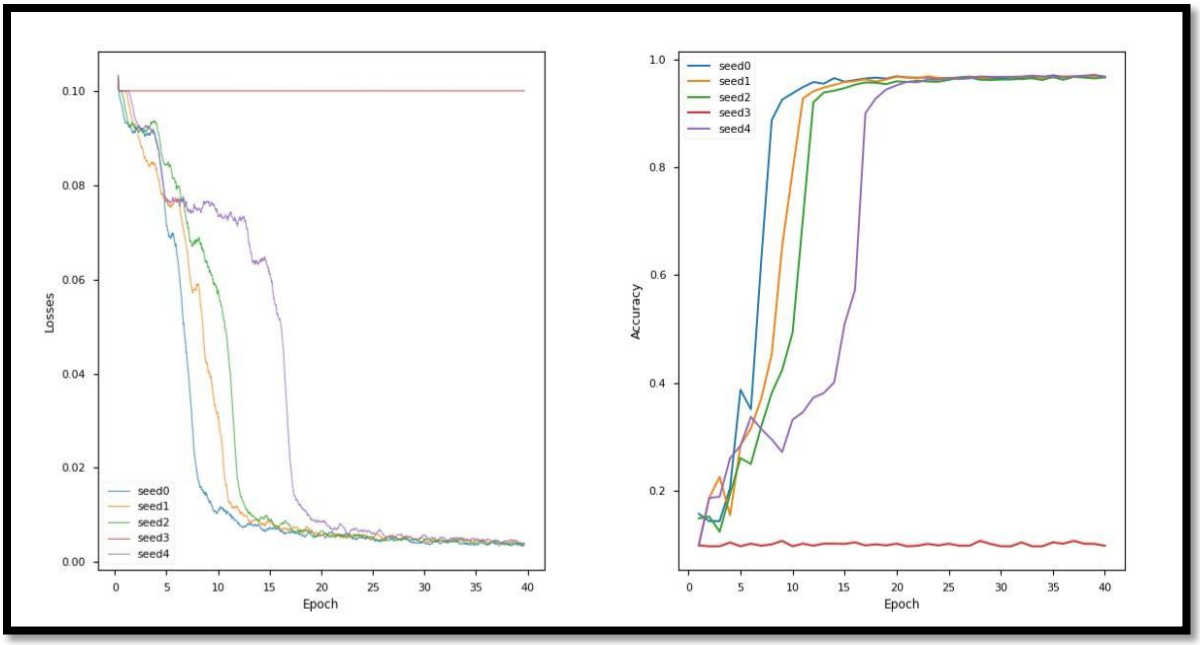


Figure (4.3): The accuracy and loss of the KMNIST dataset in ONN ( learning rate $1 \times 10^{-3}$).

Figure (4.3) and the results in table (4.2) show that the high learning rate value gave better results to a group of activation functions ( SA(approx.), SA (exact), ReLU, and sigmoid), but in the Tanh activation function it gave suboptimal results because of high speed or unstable training process.

5.2 Learning rate $1\times 10^{-4}$.

Table (4.3) illustrates the accuracy and loss values of the KMNIST dataset at learning rate $1\times 10^{-4}$.

Table (4.3): Results of the KMNIST dataset in ONN (learning rate $1\times 10^{-4}$).

| Activation Function | KMINST Dataset |
|---|---|
| SA (approx.) | $(97.04 \pm 0.1)$ % |
| SA (exact) | $(97.06 \pm 0.1)$ % |
| ReLU | $(97.08 \pm 0.1)$ % |
| Tanh | $(96.64 \pm 0.1)$ % |
| Sigmoid | $(96.90 \pm 0.1)$ % |

Figure (4.4) illustrates the accuracy and loss curves of the KMNIST dataset at learning rate $1\times 10^{-4}$.

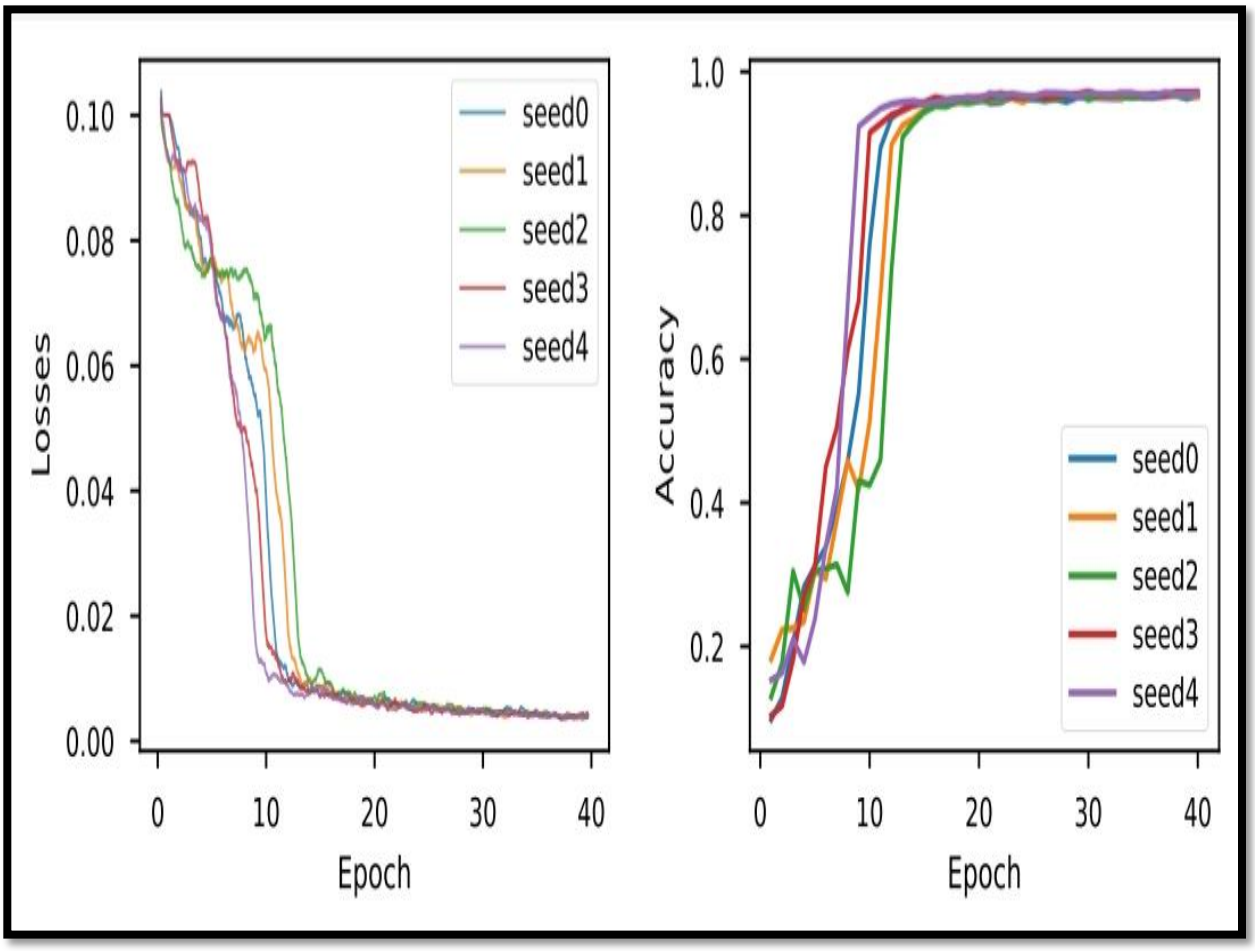


Figure (4.4): The accuracy and loss of the KMNIST dataset in ONN (learning rate $1\times 10^{-4}$).

Figure (4.4) and the results in table (4.3) show that decreasing the value of learning rate gave the system high accuracy and low loss in all activation functions.

5.3 Learning rate $1\times 10^{-5}$.

Table (4.4) illustrates the accuracy and loss values of the KMNIST dataset at learning rate $1\times 10^{-5}$.

Table (4.4): Results of the KMNIST dataset in ONN (learning rate $1\times 10^{-5}$).

| Activation Function | KMINST Dataset |
|---|---|
| SA (approx.) | $(63.70 \pm 6.6)$ % |
| SA (exact) | $(67.12 \pm 6.2)$ % |
| ReLU | $(63.90 \pm 6.4)$ % |
| Tanh | $(66.66 \pm 6.5)$ % |
| Sigmoid | $(63.86 \pm 6.4)$ % |

Figure (4.5) illustrates the accuracy and loss curves of the KMNIST dataset at learning rate $1\times 10^{-5}$.

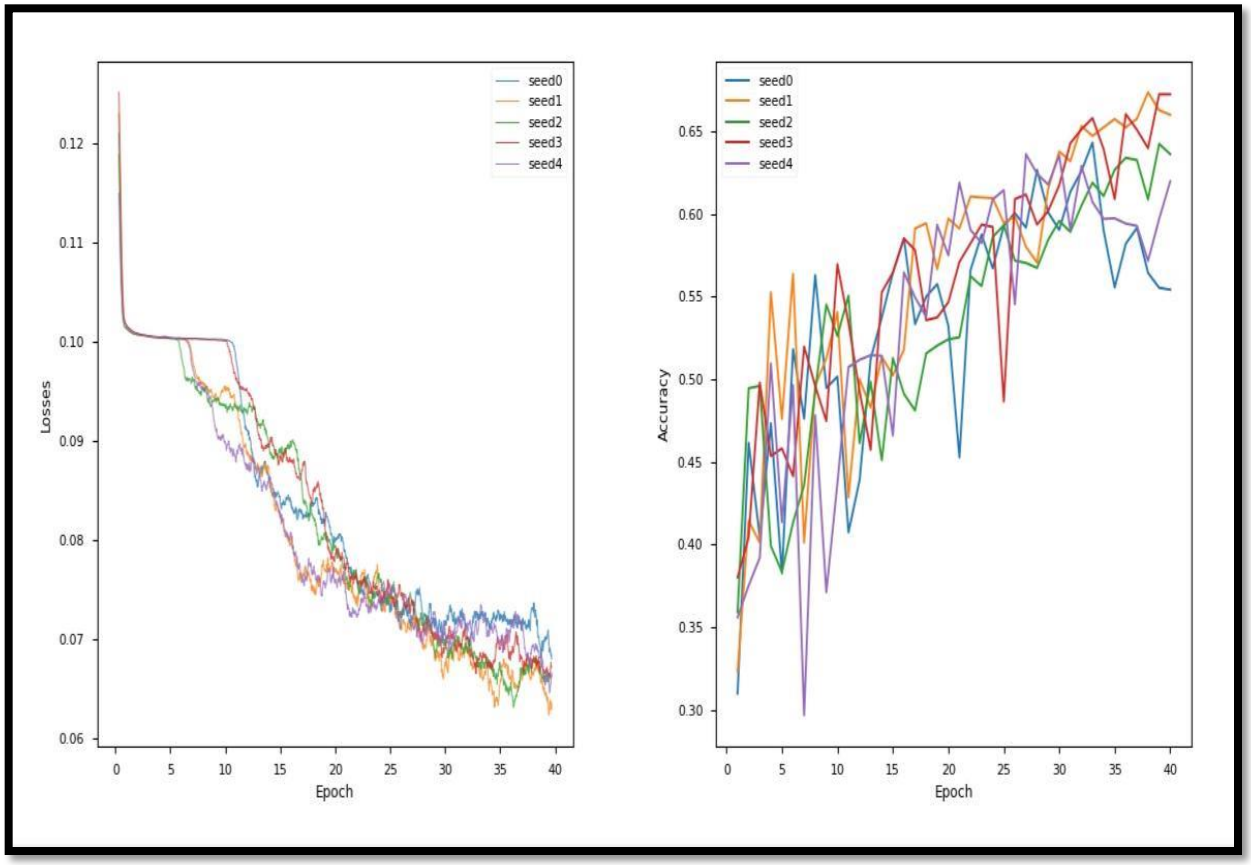


Figure (4.5): The accuracy and loss of the KMNIST dataset in ONN (learning rate $1\times 10^{-5}$).

Figure (4.5) and the results in table (4.4) show that the accuracy of the system increases with decrease in loss, but the lengthy training process because the value of learning rate is few.

The best learning rate is $1 \times 10^{-4}$ because it has given the study the best results. So, all datasets are trained with a learning rate of $1 \times 10^{-4}$.

Learning Rate $=1 \times 10^{-4}$. (In the case of MNIST, KMNIST, and EMNIST dataset).

6. The optical depth is very important in the SA activation function. So, the KMNIST dataset is trained at different optical depths ($\alpha_0$ = 2, $\alpha_0$ = 10, and $\alpha_0$ = 31) to see which fits the network best.

6.1 Optical depth ($\alpha_0$) = 2.

Table (4.5) illustrates the accuracy and loss values of the KMNIST dataset at optical depth ($\alpha_0$ = 2).

Table (4.5): Results of the KMNIST dataset in ONN ($\alpha_0$ = 2).

| Activation Function | KMINST Dataset |
|---|---|
| SA (approx.) | $(91.34 \pm 1.1)$ % |
| SA (exact) | $(90.90 \pm 2.4)$ % |
| ReLU | $(91.82 \pm 1)$ % |
| Tanh | $(83.24 \pm 3.4)$ % |
| Sigmoid | $(91.40 \pm 1.1)$ % |

Figure (4.6) illustrates the accuracy and loss curves of the KMNIST dataset at optical depth ($\alpha_0$ = 2).

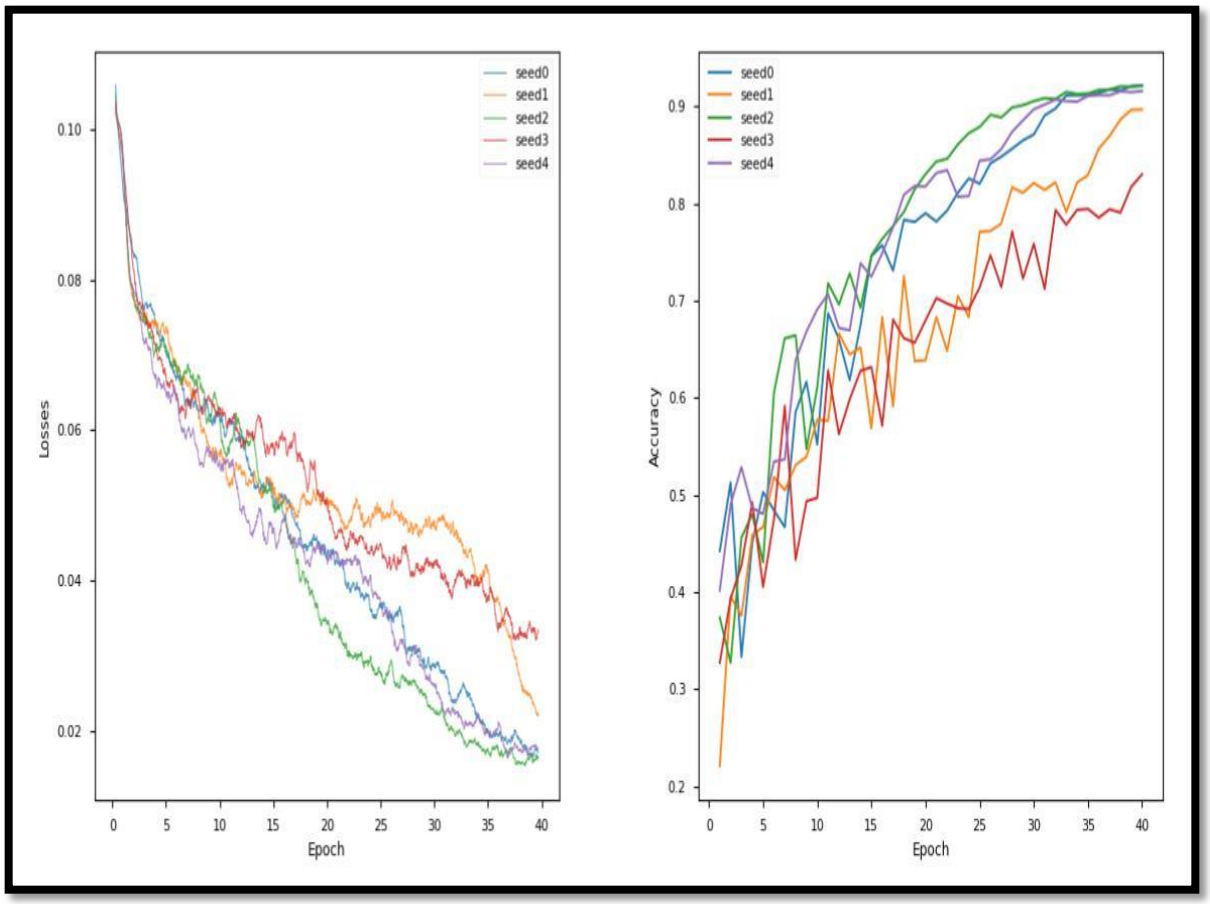


Figure (4.6): The accuracy and loss of the KMNIST dataset in ONN ($\alpha_0$ = 2).

Figure (4.6) and the results in table (4.5) show that the low optical depth value made the performance of the approximate derivatives well because the low optical depth leads to weak non-linearity in the unsaturated region as illustrated in figure (2.14.a) on the left.

6.2 Optical depth ($\alpha_0$) = 10.

Figure (4.4) illustrates the accuracy and loss curves of the KMNIST dataset at optical depth ($\alpha_0$) = 10, and table (4.3) illustrates the results of the KMNIST dataset at optical depth ($\alpha_0$) = 10. The results show that the network performance improves with increasing optical depth and it is increased to nearly optimal levels at optical depth ($\alpha_0$) = 10.

6.3 Optical depth ($\alpha_0$) = 31.

Table (4.6) illustrates the accuracy and loss values of the KMNIST dataset at optical depth ($\alpha_0$ = 31).

Table (4.6): Results of the KMNIST dataset in ONN ($\alpha_0$ = 31).

| Activation Function | KMINST Dataset |
|---|---|
| SA (approx.) | $(18.82 \pm 9.2)$ % |
| SA (exact) | $(9.64 \pm 10)$ % |
| ReLU | $(9.62 \pm 10)$ % |
| Tanh | $(9.64 \pm 10)$ % |
| Sigmoid | $(9.56 \pm 10)$ % |

Figure (4.7) illustrates the accuracy and loss curves of the KMNIST dataset at optical depth ($\alpha_0$ = 31).

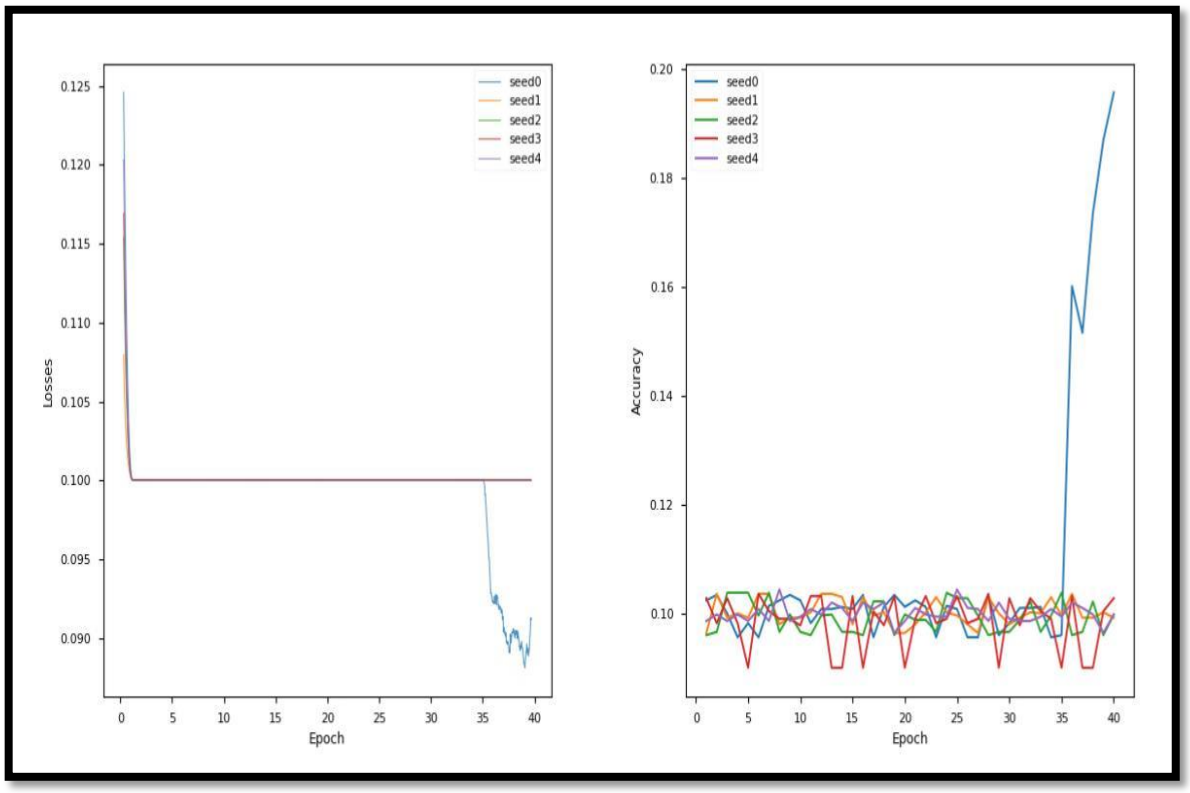


Figure (4.7): The accuracy and loss of the KMNIST dataset in ONN ($\alpha_0$ = 31).

Figure (4.7) and the results in table (4.6) show that the high optical depth value made the approximate derivatives decrease, which makes the non-linear part large in the unsaturated region as illustrated in figure (2.14.a) on the right.

The best optical depth is ($\alpha_0$ = 10) because it has given the study the best results. So, all datasets are trained with an optical depth of ($\alpha_0$ = 10).

Optical depth ($\alpha_0$) = 10. (In the case of MNIST, KMNIST, and EMNIST dataset).

Table (4.7) illustrates the accuracy and loss values of the EMNIST, KMNIST, and MNIST datasets in ONN using learning rate ($1 \times 10^{-4}$) and optical depth ($\alpha_0$ = 10).

Table (4.7): Results of ONN.

| Activation Function | EMNIST Dataset | KMINST Dataset | MNIST Dataset |
|---|---|---|---|
| SA (approx.) | (89.73 ± 0.3) % | (97.04 ± 0.1)% | (99.50 ± 0.1) % |
| SA (exact) | (89.35 ± 0.2) % | (97.06 ± 0.1)% | (99.28 ± 0.1) % |
| ReLU | (89.28 ± 0.2) % | (97.08 ± 0.1)% | (99.42 ± 0.1) % |
| Tanh | (89.24 ± 0.2) % | (96.64 ± 0.1)% | (99.50 ± 0.1) % |
| Sigmoid | (89.19 ± 0.2) % | (96.90 ± 0.1)% | (99.36 ± 0.1) % |

Figure (4.8) illustrates the accuracy and loss curves of the EMNIST dataset in ONN.

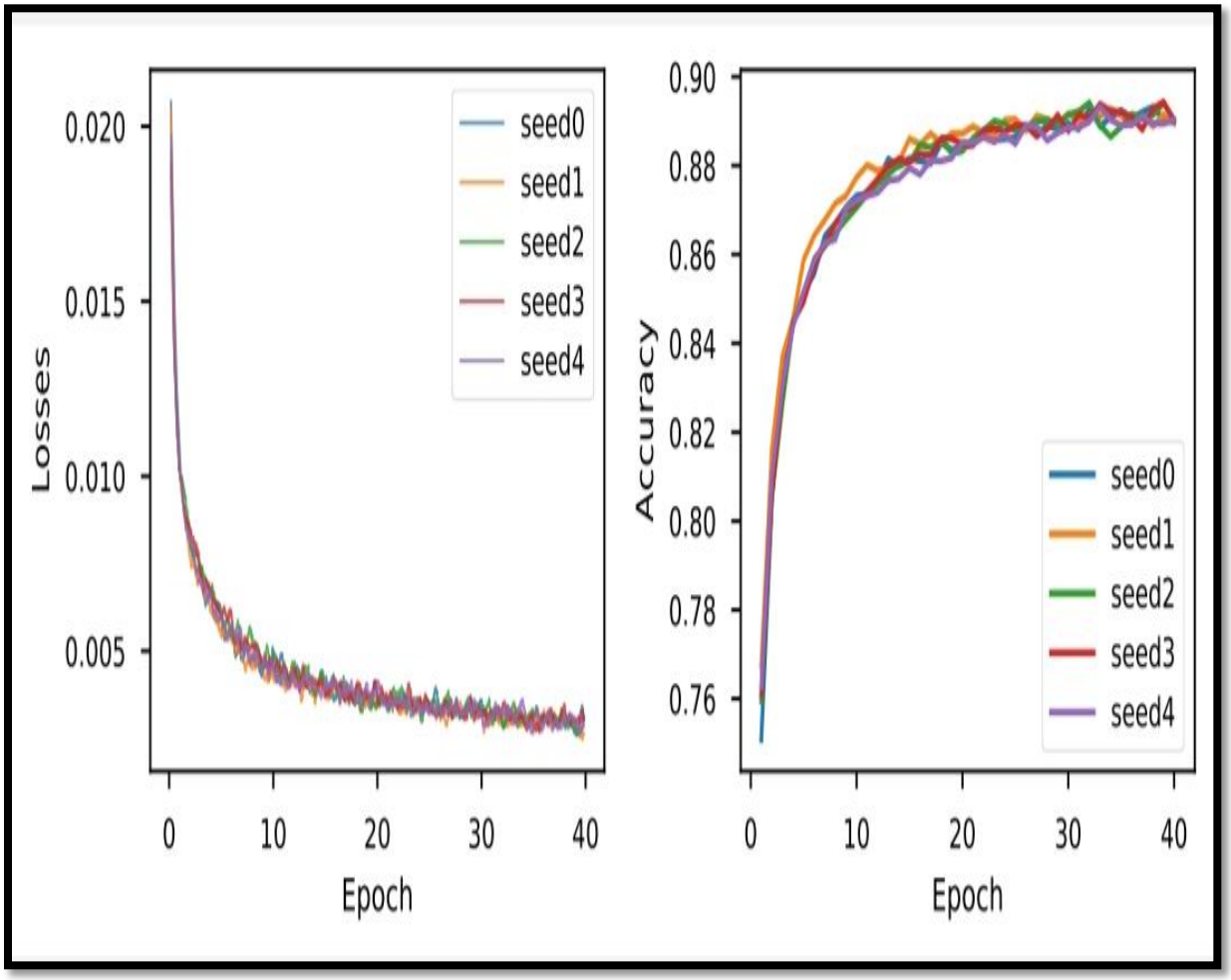


Figure (4.8): The accuracy and loss of the EMNIST dataset in ONN.

Figure (4.4) illustrates the accuracy and loss curves of the KMNIST dataset in ONN.

Figure (4.9) illustrates the accuracy and loss curves of the MNIST dataset in ONN.

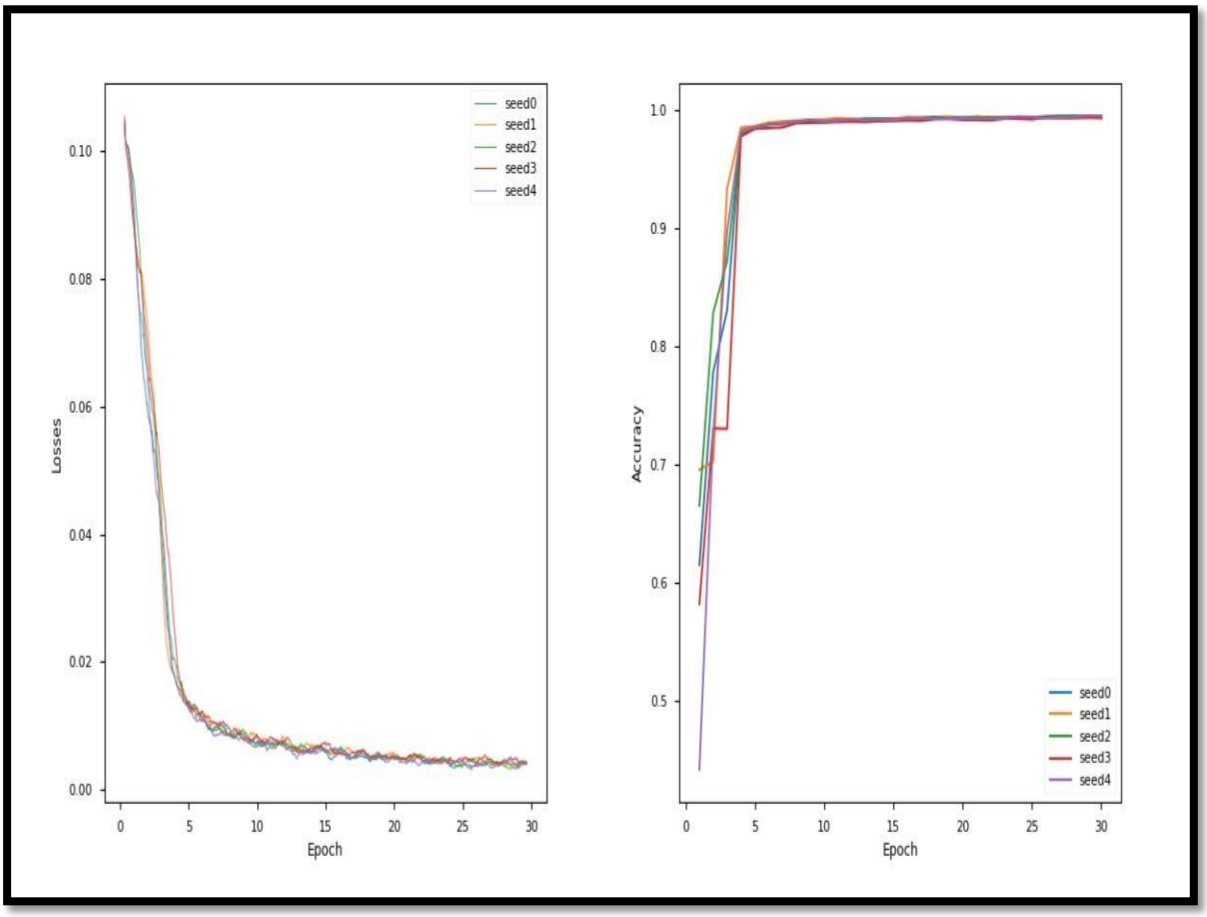


Figure (4.9): The accuracy and loss of the MNIST dataset in ONN.

The results in table (4.7) show that there is no difference in performance of the SA activation functions regardless of whether the true derivatives in equation (2.21) or the derivatives approximations in equation (2.20).

## 4.4 Comparison between Optical Neural Networks and Artificial Neural Networks.

The MAE loss function is used to train ONNs, whereas CCE is used as the baseline for ANNs. The reason for this decision is that an optical setup makes it simple to calculate gradients for MAE loss, while softmax in CCE requires offline computation. However, as CCE is the standard choice for

classification issues in the deep learning world, the ANNs employ it. Table (4.8) illustrates the results of the ANNs and the results of the ONNs.

Table (4.8): A comparison between ANNs and ONNs.

| Activation Function | Results of ANNs | | Results of ONNs | |
|---|---|---|---|---|
| | KMNIST Dataset | MNIST Dataset | KMNIST Dataset | MNIST Dataset |
| SA (approx.) | $(95.90 \pm 1)\%$ | $(99.14 \pm 1)\%$ | $(97.04 \pm 0.1)\%$ | $(99.50 \pm 0.1)\%$ |
| SA (exact) | $(95.64 \pm 1)\%$ | $(99.28 \pm 1)\%$ | $(97.06 \pm 0.1)\%$ | $(99.28 \pm 0.1)\%$ |
| ReLU | $(95.68 \pm 1)\%$ | $(99.22 \pm 1)\%$ | $(97.08 \pm 0.1)\%$ | $(99.42 \pm 0.1)\%$ |
| Tanh | $(96.26 \pm 1)\%$ | $(99.16 \pm 1)\%$ | $(96.64 \pm 0.1)\%$ | $(99.50 \pm 0.1)\%$ |
| Sigmoid | $(95.62 \pm 1)\%$ | $(99.30 \pm 1)\%$ | $(96.90 \pm 0.1)\%$ | $(99.36 \pm 0.1)\%$ |

The results in table (4.8) show that the accuracy of ANNs is almost close to the accuracy of ONNs. As for the value of the loss ONNs is much better than the value of the loss of ANNs, as well as for the processing speed in the implementation of ONNs, it was faster than the speed of the implementation of ANNs and the reason is due to the distinctive characteristics of ONNs compared to ANNs.

## 4.5 Comparing the Results with Related work

In this comparison, the results of the study were compared with a suitable resource for this study. The paper written by (X. Guo, et al., (2021)) [4] "Back-propagation through non-linear units for the all-optical training of neural networks", table (4.9) shows the results of the suggested method and the paper [4].

Table (4.9): Results of the suggested method and the paper [4].

| Activation Function | Results of suggested Method | | | Results of Paper [4] | | |
|---|---|---|---|---|---|---|
| | EMNIST Dataset | KMNIST Dataset | MNIST Dataset | EMNIST Dataset | KMNIST Dataset | MNIST Dataset |
| SA (approx.) | (89.73 ± 0.3)% | (97.04 ± 0.1)% | (99.50 ± 0.1)% | (87.9 ± 0.4)% | (95.4 ± 0.3)% | (99.3 ± 0.1)% |
| SA (exact) | (89.35 ± 0.2)% | (97.06 ± 0.1)% | (99.28 ± 0.1)% | (88.1 ± 0.3) | (96.3 ± 0.1)% | (99.4 ± 0.1)% |
| ReLU | (89.28 ± 0.2)% | (97.08 ± 0.1)% | (99.42 ± 0.1)% | (88.6 ± 0.2)% | (96.1 ± 0.3)% | (99.3 ± 0.1)% |
| Tanh | (89.24 ± 0.2)% | (96.64 ± 0.1)% | (99.50 ± 0.1) | (87.5 ± 0.2)% | (95.6 ± 0.2)% | (99.2 ± 0.1)% |
| Sigmoid | (89.19 ± 0.2)% | (96.90 ± 0.1)% | (99.36 ± 0.1)% | (87.7 ± 0.3)% | (95.8 ± 0.2)% | (99.0 ± 0.1)% |

By comparing the results of this study with the results of the previous research [4] as shown in Table (4.9), the results of this study exceeded the results of the previous research [4], and thus benefited from updating the proposed optical neural network training steps benefited from reducing the loss factor and increasing the accuracy factor, and the reason for this is due to use of:

1. This study used max pooling in the pooling layer in the CNN, which takes the largest element in the image segment at a predetermined size instead of average pooling, which averages the elements in the image segment with a predetermined size in paper [4].

2. This study used three hidden layers, the first hidden layer has 512 neurons, the second hidden layer has 256 neurons and the third hidden layer has 128 neurons, but the paper [4] used a single hidden layer consisting of 128 neurons.

3. This study used the loss function is MAE, but the paper [4] used the loss function is MSE.

4. This study used learning rate of $1 \times 10^{-4}$, but the paper [4] used learning rate is $5 \times 10^{-4}$.

5. This study used batch_size_test = 128 for (MNIST and KMNIST), batch_size_test = 1000 for EMNIST, but the paper [4] used batch_size_test = 1000 for all dataset (MNIST, KMNIST and EMNIST).

# CHAPTER FIVE

# Conclusions and Future Works

# CHAPTER FIVE

# Conclusions and Future Works

## 5.1 Conclusions

During the implementation of the proposed methodology, several notes can be concluded as an outcome of this study:

1. The use of ONN in processing data systems is a good method for its distinctive properties, such as high speed, bandwidth, high correlation, internally balanced processing, and low power consumption which makes it a very promising method to be used in the future.

2. SA non-linear activation function depends on the optical depth, so choosing the optical depth is critical. SA non-linear activation function achieves the best results at medium optical depths, but at very low and very high optical depths, the results are not at the desired level.

3. Using batch_size_train in a model is very important to speed up the training process by taking a small set of the total dataset in each iteration. If it is not used, it takes the total dataset in each iteration, and this causes a slowdown in the training process. Also, in the test case using batch_size_test.

4. The task of choosing the learning rate in the optimization algorithm is somehow difficult. When choosing a relatively small learning rate, the learning process is slowed, and the training time becomes too large. While choosing a relatively large learning rate can lead to fluctuation in the loss value around the minimum value, this hinders the convergence process.

## 5.2 Future Works

Many directions can be suggested in future work that may enhance the system results and reduce the training time required if they are applied. The following are some of these directions.

1. Use a new dataset to check accuracy and loss that contains the large number of images. The goal of using this dataset is to increase generalization during system training.

2. Working on developing a new loss function and optimization algorithm to reduce the system training time.

3. Using the amplifier's task in the fully connected layer instead of the classification task circumventing the need for SA modules entirely.

4. Designing them practically using Field Programmable Photonic Gate Array (FPPGA), which aims to play a similar role in photonics as Field Programmable Gate Array (FPGA) which is used in electronics.

5. Using other kinds of evaluation measures are available for classification tasks: such as Recall, Precision, and F1 Score.

# Reference


[1] A. RYOU, J. Whitehead, M. Zhelyeznyakov, P. Anderson, C. Keskin, M. Bajcsy, and A. Majumdar, "Free space optical neural network based on thermal atomic nonlinearity." vol. 9, issue 4, 2021, pp. B128-B134.

[2] R. Xu, P. Lv, F. Xu, and Y. Shi, "A survey of approaches for implementing optical neural networks," *Opt. Laser Technol.*, vol. 136, no. July 2020, p. 106787, 2021, doi: 10.1016/j.optlastec.2020.106787.

[3] D. Zhang, P. Wang, G. Luo, Y. Bi, Y. Zhang, J. Yr, Y. Su, Y. Zhang, and J. Pan., "Design of a Silicon-based Optical Neural Network," vol. 93, no. Mmsta, pp. 184–186, 2019, doi: 10.2991/mmsta-19.2019.39.

[4] X. Guo, T. D. Barrett, Z. M. Wang, and A. I. Lvovsky, "Backpropagation through nonlinear units for the all-optical training of neural networks," *Photonics Res.*, vol. 9, no. 3, p. B71, 2021, doi: 10.1364/prj.411104.

[5] Q. Zhang, H. Yu, M. Barbiero, B. Wang, and M. Gu, "Artificial neural networks enabled by nanophotonics," *Light Sci. Appl.*, vol. 8, no. 1, 2019, doi: 10.1038/s41377-019-0151-0.

[6] J. Xiong, Z. Zhang, and J. Xu, "Advances and marine applications of optical neural network," no. March 2021, p. 75, 2021, doi: 10.1117/12.2586296.

[7] Y. guo Wang, and H. Peng Li, "Remote sensing image classification based on artificial neural network," 10.1109/ CMCE.5610049, 25 October 2010, doi: 978-1-4244-7958-0.

[8] N. Sharma and T. Gedeon, “Classification of physiological sensor signals using artificial neural networks,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8227 LNCS, no. PART 2, pp. 504–511, 2013, doi: 10.1007/978-3-642-42042-9_63.

[9] M. Jabardi and H. Kuar, “Artificial Neural Network Classification for Handwritten Digits Recognition,” *Int. J. Adv. Res. Comput. Sci.*, vol. 5, no. 3, April, pp. 107–111, 2014, [Online]. Available: http://www.ijarcs.info/index.php/Ihttp://www.ijarcs.info/index.php/Ijarcs/article/view/2061jarcs/issue/view/38.

[10] A. Graves and N. Jaitly, “Towards End-to -End Speech Recognition with Recurrent Neural Networks,” *ICML’14*, vol. 32, June, pp. ll-1764-ll-1772.

[11] K. Malathi and R. Nedunchelian, “Detecting and classifying diabetic retinopathy in fundus retina images using artificial neural networks-based firefly clustering algorithm,” *ARPN J. Eng. Appl. Sci.*, vol. 11, no. 5, pp. 3419–3426, 2016.

[12] E.G. Silva Júnior, D.B.O. Cardoso, M.C. Reis, A.F.O. Nascimento , D.I. Bortolin, M.R. Martins and L.B. Sousa., “Cotton genotypes selection through artificial neural networks,” *Genet. Mol. Res.*, vol. 16, no. 3, 2017, doi: 10.4238/gmr16039798.

[13] M. B. Mehmandari, A. S. Bakhshayesh, and C. T. Sindi, “Detection and Classification of Wear in Rotary Systems based on ANN and GA,” pp. 9995–10004, 2019, doi: 10.15680/IJIRSET.2019.0810017.

[14] M. Der Yang, K. H. Huang, and H. P. Tsai, “Integrating MNF and HHT transformations into artificial neural networks for hyperspectral image classification,” *Remote Sens.*, vol. 12, no. 14, 2020, doi: 10.3390/rs12142327.

[15] S. M. K. Chaitanya and P. Rajesh Kumar, “Kidney images classification using cuckoo search algorithm and artificial neural network,” *Int. J. Sci. Technol. Res.*, vol. 9, no. 3, pp. 6127–6132, 2020.

[16] A. N. Ismael, H. Al. Abed, and M. A. Abed, “Classification of Groundwater Quality using Artificial Neural Networks in Safwan and Al-Zubayr in Basra,” *CSP*, vol.4, lssue 1, 2020, doi; 10.23977//acss.2020.040105.

[17] D. Vujičić, R. Pavlović, D. Milošević, B. Dorđević, S. Randié and D. Stojic, “CLASSIFICATION OF ASTEROID FAMILIES WITH ARTIFICIAL NEURAL NETWORKS,” Serb. Astron. J. N 201, 2020, pp. 39-48, doi: https://doi.org/10.2298/SAJ2001039V.

[18] A. Jha, C. Huang, H. T. Peng, W. Zhang, B. Shastri, and P. R. Prucnal, “High-speed time series prediction and classification on an all-optical neural network,” Opt. InfoBase Conf. Pap., vol. 1, no. c, pp. 2–4, 2016, doi: 10.1364/ofc.2022.tu3g.-57169

[19] G. Fennessy, “Optical Neural Networks,” 16 May, 2018, arXiv: 1805.06082v1 [cs.CV].

[20] J. Chang, V. Sitzmann, X. Dun, W. Heidrich, and G. Wetzstein, “Hybrid optical-electronic convolutional neural networks with optimized

diffractive optics for image classification," *Sci. Rep.*, vol. 8, no. 1, pp. 1–10, 2018, doi: 10.1038/s41598-018-30619-y.

[21] S. Geoffroy-Gagnon, F. Shokraneh, and O. Liboiron-Ladouceur, "Analysis of an analog optical neural network," *Front. Opt. - Proc. Front. Opt. + Laser Sci. APS/DLS*, no. January 2020, pp. 10–12, 2019, doi: 10.1364/FIO.2019.JW3A.96.

[22] R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić, and D. Englund, "Large-Scale Optical Neural Networks Based on Photoelectric Multiplication," *Phys. Rev. X*, vol. 9, no. 2, pp. 1–12, 2019, doi: 10.1103/PhysRevX.9.021032.

[23] M. Y.-S. Fang, S. Manipatruni, C. Wierzynski, A. Khosrowshahi, and M. R. DeWeese, "Design of optical neural networks with component imprecisions," *Opt. Express*, vol. 27, no. 10, p. 14009, 2019, doi: 10.1364/oe.27.014009.

[24] M. Paraschiv, R. Padrino, P. Casaris, and A. F. Anta, "Very Small Neural Networks for Optical Classification of Fish Images and Videos," *JM SE 10(6): 736*, May 2020, doi: 10.3390/jmse10060736.

[25] H. Zhang, M. Gu, X. D. Jiang, J. Thompson, H. Cai, S. Paesani, R. Santagati, A. Laing, Y. Zhang, M. H. Yung, Y. Z. Shio, F. K. Muhammad, G. Q. Lo, X. S. Luo, B. Dong, D. L. Kwong, L. C. Kwek & A. Q. Liu., "An optical neural chip for implementing complex-valued neural network," *Nat. Commun.*, vol. 12, no. 1, pp. 1–11, 2021, doi: 10.1038/s41467-020-20719-7.

[26] Y. Sun, Mingli Dong, Mingxin Yu, Jiabin Xia, Xu Zhang, Yuchen Bai, Lidan Lu, and Lianqing Zhu., "Nonlinear All-Optical Diffractive Deep Neural Network with 10.6 μ m Wavelength for Image Classification," *Int. J. Opt.*, vol. 2021, 2021, doi: 10.1155/2021/6667495.

[27] T. Wang, S. Y. Ma, L. G. Wright, T. Onodera, B. C. Richard, and P. L. McMahon, "An optical neural network using less than 1 photon per multiplication," *Nat.* Commun., vol. 13, no. 1, pp. 1–8, 2022, doi: 10.1364/ofc.2022.tu3g.3.

[28] S. Sharma and A. Athaiya, "ACTIVATION FUNCTIONS IN NEURAL NETWORKS," *IJEAST,* vol. 4, Issue 12 , ISSN No. 2455-2143 , pp. 310-316, April 2020.

[29] P. A. Birdi, Yashika, Tanjyot Aurora, "Study of Artificial Neural Networks and neural implants," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 1(4), pp. 258–62, 2013.

[30] X. Sui, Q. Wu, J. Liu, Q. Chen, and G. Gu, "A review of optical neural networks," *IEEE Access*, vol. 8, pp. 70773–70783, 2020, doi: 10.1109/ACCESS.2020.2987333.

[31] K. I. Ahamed and S. Akthar, "A Study on Neural Network Architectures," *Comput. Eng. Intell. Syst.*, vol. 7, no. 9, pp. 1–7, 2016, [Online]. Available: https://iiste.org/Journals/index.php/CEIS/article/view/32857.

[32] Ms. Sonali. B. Maind and Ms. Priyanka Wankar, “Research Paper on Basic of Artificial Neural Network,” *IJRITCC*, vol.2, Issue: 1, pp. 96-100, January 2014.

[33] M. I. A. ALFarra, “Improving Solar Power System’s Efficiency Using Artificial Neural Network,” 2018, [Online]. Available: https://library.iugaza.edu.ps/thesis/124591.pdf.

[34] C. Denz, *Optical Neural Networks*. 1998.

[35] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, “Deep Learning for Computer Vision: A Brief Review,” *CIN*, vol. 2018, doi: 10.1155/2018/7068349.

[36] M. Guarascio, G. Manco, and E. Ritacco, “Deep learning in Encyclopedia of Bioinformatics and Computational Biology,” ABC of Bioinformatics, 2018.

[37] C. C. Aggarwal, “Neural Networks and Deep Learning,” 2018.

[38] K. Yun, A. Huyen, and T. Lu, “Deep neural networks for pattern recognition,” in Advances in Pattern Recognition Research, 2018.

[39] F. Schilling, “The Effect of Batch Normalization on Deep Convolutional Neural Networks,” p. 113, 2016, [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A955562&dswid=-5716.

[40] Y. LeCun, C. Cortes, C. Burges, and ATT Labs, “MNIST handwritten digit database,” http://yann.lecun.com/exdb/mnist (2010).

[41] https://www.javatpoint.com/pytorch-mnist-dataset-of-image-recognition.

[42] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, “Deep learning for classical Japanese literature,”, 2018, arXiv:1812.01718.

[43] http://naruhodo.weebly.com/blog/introduction-to-kuzushiji.

[44] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “EMNIST: an extension of MNIST to handwritten letters,”, 2017, arXiv: 1702.05373.

[45] https://www.simonwenkel.com/notes/ai/datasets/vision/EMNIST.htm emnist-balanced.

[46] S. Ji, W. Xu, M. Yang, and K. Yu, “3D Convolutional neural networks for human action recognition,”, *ICMR*, 2010, doi: 10.1109/TPAMI.2012.59.

[47] A. Krizhevsky. I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,”, ACM, vol. 60, lssue 6, pp. 84-90, June 2017, doi: 10.1145/3065386.

[48] C. Zhang. P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. “Optimizing FPGA based accelerator design for deep convolutional neural networks,”, *ISFPGA*, PP. 161-170, February *2015*, doi: 10.1145/2684746.2689060.

[49] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, “Computer vision for human-machine interaction,”, *EECMS*, 2018, doi: 10.1016/B978-0-12-813445-0.00005-8.

[50] N. Milosevic, “Introduction to Convolutional Neural Networks,” 2020.

[51] “Convolutional Neural Network | Introduction to Data Science,” [Online]. Available: https://scientistcafe.com/ids/convolutional-neural-network.html.

[52] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, “Flexible, high performance convolutional neural networks for image classification,” *IJAMEC,* 2011, doi: 10.5591/978-1-57735-516-8 /IJCAI11-210.

[53] “Max-pooling and Average-pooling,” https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451/download.

[54] R. Yamashita, M. Nishio, R. Kinh Gian Do, K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *NCBI,* 2018, https://doi.org/10.1007/s13244-018-0639-9.

[55] “Convolutional Neural Networks (CNN) ,” https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening.

[56] “Layers of a Convolutional Neural Network,” https://wiki.tum.de/plugins/servlet/mobile?contentId=23570015#content/view/23570015.

[57] Y. Zuo, B. Li, Y. Zhao, Y. Jiang, Y. Chen, P. Chen, G. Jo, J. Liu, and S. Du., “All-optical neural network with nonlinear activation functions,” *Optica*, vol. 6, no. 9, p. 1132, 2019, doi: 10.1364/optica.6.001132.

[58] M. A. Nielsen, “Neural Networks and Deep Learning.,” Determination Press, 2018.

[59] R. E. Neapolitan and R. E. Neapolitan, “Neural Networks and Deep Learning,” in Artificial Intelligence, 2018.

[60] S. Theodoridis, “Neural Networks and Deep Learning,” in Machine Learning, 2015.

[61] P. M. Buscema, G. Massini, M. Breda, W. A. Lodwick, F. Newman, and M. Asadi-Zeydabadi, “Review on Artificial neural network,” *IJAREEIE*, vol. 6, lssue 6, June 2017, doi: 10.15662/IJAREEIE.2017.0606142.

[62] C. J. Willmott and K. Matsuura, “Advantages of the Mean Absolute Error (MAE) Over the Root Mean Square Error (RMSE) in Assessing Aven Model Performance,” *Climate Research*, vol. 30, no. 1, pp. 79-82, 2005, doi: 10.3354/cr030079.

[63] K. Hu, Z. Zhang, X. Niu, Y. Zhang, C. Cao, F. Xiao, and X. Gao, “Retinal Vessel Segmentation of Color Fundus Images Using Multiscale Convolutional Neural Network with an Improved Cross-Entropy Loss Function,” *Neurocomputing*, vol. 309, pp. 179 191, 2018, https://doi.org/10.1016/j.neucom.2018.05.011.

[64] Q. V. Le, J. Ngiam. A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, “On optimization methods for deep learning,” *2nd.ICML*, 2011.

[65] S. Ruder, “An overview of gradient descent optimization algorithms,” *Computer Science,* 15 June 2017.

[66] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[67] R. Rajakumari and L. Kalaivani, “Breast Cancer Detection and Classification Using Deep CNN Techniques,” *IASC*, vol.32,2022.

[68] W. Jiang, “MNIST-MIX: a multi-language handwritten digit recognition dataset,” *IOP*, vol. 1, no. 2, 2020, doi: 10.1088/2633-1357/abad0e.

[69] Larry D. Jones and A. Foster Chin, “Electronic Instruments and Measurements,” John Wiley & Sons, 1987.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابـل
كلية الهندسة / قسم الهندسة الكهربائية

# تصميم ومحاكاة نظام شبكة عصبية هجينة

## رســـالـــــة

مقدمة الى مجلس كلية الهندسة جامعة بابل في استيفاء
جزئي لمتطلبات درجة الماجستير
في الهندسة الكهربائية / الكترونيك الصناعي

من قبل
زينب فارس طالب هاشم

أشــراف
الأستاذ الدكتور / أبراهيم عبد الله مرداس

١٤٤٤هـ ٢٠٢٣م

**الملخص**

على الرغم من أن الشبكات العصبية الاصطناعية تحقق تقدما كبير في مجالات التكنولوجيا الواسعة والمتنوعة، لكن تتطلب عملية تدريب معلمات الشبكة العصبية الاصطناعية قدرًا كبيرًا من الوقت والطاقة. من الضروري استخدام شبكات عصبية جديدة تختلف عن الشبكات العصبية الاصطناعية المعروفة باسم الشبكات العصبية الضوئية. يمكن لشبكات العصبية الضوئية تحسن اداء المهام المطلوبة بشكل كبير من خلال الاستفادة من خصائصها العالية مثل السرعة العالية وعرض النطاق الترددي العالي والمعالجة المتوازنة العالية والاستهلاك المنخفض للطاقة. تقترح الدراسة تدريب شبكة عصبية بصرية من خلال وحدات غير خطية بواسطة امتصاص قابل للتشبع (SA) لأداء مهام تصنيف الصور. يتم تدريب واختبار الشبكات العصبية بمساعدة مجموعات البيانات (KMNIST، EMNIST، MNIST). تحتوي جميع البيانات على صور بأبعاد (28 × 28) بكسل بتدرج الرمادي. تقوم الشبكة العصبية التلافيف بمعالجة الصور عبر سلسلة من طبقات لاستخراج ميزات الصور. بعدها يدخل الاخراج الى الشبكة العصبية الضوئية التي تقوم بمعالجة الميزات بشكل أكبر وإجراء التصنيف النهائي. يستخدم متوسط الخطأ المطلق لتحديد الخطأ بين المخرجات الفعلية والمتوقعة. ثم يُعاد نشر الخطأ عبر الشبكة لتحديث الأوزان باستخدام خوارزمية الانتشار العكسي. تحدث عملية التدريب عن طريق تكرار عملية التمرير للأمام والخلف في الاتجاه الذي يقلل من وظيفة الخسارة إلى أدنى قيمة. بعد الانتهاء من عملية التدريب واستخراج النتائج، تستخدم مجموعة الاختبار لتقييم أداء النموذج. تضمنت نتائج النظام زيادة في عامل الدقة وانخفاض عامل الخسارة لذلك ان النظام المقترح مثالي لتدريب الشبكة العصبية البصرية وأداء مهام تصنيف الصور لمجموعات البيانات. أخيرًا، تمت عمل مقارنة اداء الشبكات العصبية الضوئية مع أداء الشبكات العصبية الاصطناعية الذي يستخدم وظائف تنشيط ReLU. يلاحظ من خلال النتائج أن دقة الشبكات العصبية الاصطناعية قريبة تقريبًا من دقة شبكات العصبية الضوئية. أما بالنسبة لقيمة خسارة الشبكات العصبية الضوئية فهي أفضل بكثير من قيمة خسارة ANNs، وكذلك بالنسبة لسرعة المعالجة في تنفيذ الشبكات العصبية الضوئية فقد كانت أسرع من سرعة تنفيذ الشبكات العصبية الاصطناعية والسبب يرجع إلى الخصائص المميزة للشبكات العصبية الضوئية مقارنة بشبكات العصبية الاصطناعية .