

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/354378596>

# A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition

Thesis · September 2021

---

CITATIONS

0

READS

151

2 authors, including:



[Hind Ibrahim Mohammed](#)

university of Diyala::Al-Muqdad College of Education

2 PUBLICATIONS 1 CITATION

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition [View project](#)



Ministry of Higher Education and  
Scientific Research  
University of Diyala  
College of Science  
Department of Computer Science



# **A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition**

**A Dissertation**

**Submitted to the Department of Computer Science\ College  
of Sciences\ University of Diyala in a Partial Fulfillment of the  
Requirements for the Degree of Master in Computer Science**

**By**

**Hind Ibrahim Mohammed Saba**

**Supervised By**

**Assist.Prof. Dr. Jumana W. Salih**

**2021 A.D.**

**1443 A.H**

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

﴿ قَالُوا سُبْحَانَكَ لَا عِلْمَ

لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ

أَنْتَ الْعَلِيمُ الْحَكِيمُ ﴾

صدق الله العظيم

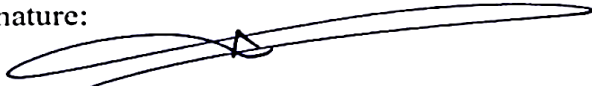
سورة البقرة (الآية: 32)

## **(Supervisor's Certification)**

We certify that this research entitled “A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition” was prepared by (Hind Ibrahim Mohammed) Under My supervision at the University of Diyala Faculty of Science Department of Computer Science, as a partial fulfillment of the requirement needed to award the degree of Master of Science in Computer Science.

(Supervisor)

Signature:



Name: Assist.Prof. Dr. Jumana W. Salih

Date: 5 / 9 / 2021

Approved by University of Diyala Faculty of Science Department of Computer Science.

Signature:



Name: Assist.Prof. Dr. Basher Talib Hamid

Date: 5 / 9 / 2021

**(Head of Computer Science Department)**

## Linguistic Certification

This is to certify that this thesis entitled “A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition” was prepared by (Hind Ibrahim Mohammed) at the University of Diyala/Department of Computer Science, is reviewed linguistically. Its language was amended to meet the style of the English language.

Signature: 

Name: Dr. Ghazwan Mohammed Jaafar

Date: 1 / 9 / 2021

## Scientific Amendment

I certify that the thesis entitled “**A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition**” was prepared by **(Hind Ibrahim Mohammed)** has been evaluated scientifically; therefore, it is suitable for debate by the examining committee.

**Signature:**



**Name: Assist.Prof. Dr. Sawsen Abdulhadi Mahmood**

**Date: 2 / 9 / 2021**

## Scientific Amendment

I certify that the thesis entitled “**A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition**” was prepared by (**Hind Ibrahim Mohammed**) has been evaluated scientifically; therefore, it is suitable for debate by the examining committee.

Signature:



Name: **Assist.Prof. Dr. Ghadah Kadhim Al-Khafaji**

Date: 1/19/2021

# Examination Committee Certification

We certify that we have read the thesis entitled "A Comparison between MSVM and CNN Algorithms in Offline Hand Gesture Recognition" and an examination committee, examined the student (**Hind Ibrahim Mohammed**) in the thesis content and that in our opinion, it is adequate as fulfill the requirement for the Degree of Master in Computer Science at the Computer Science Department, University of Diyala.

(Chairman)

Signature: 


Name: Prof. Dr. Ziyad Tariq Mustafa

Date: 5 / 9 / 2021

Signature: 


Name: Assist. Prof. Dr. Bashar M. Nema (Member)

Date: 1 / 9 / 2021

Signature: 

Name: Dr. Khalid Mohammed Saffer (Member)

Date: 1 / 9 / 2021


Signature: 

Name: Asst. Prof. Dr. Jumana W. Salih (Supervisor)

Date: 5 / 9 / 2021

Approved by the Dean of College of Science, University of Diyala

(The Dean)

Signature: 

Name: Prof. Dr. Tahseen H. Mubarak

Date: / / 2021



# Dedication

I would like to dedicate this work to:

To my father may God have mercy on him and mother May God protect her and prolong her life.

To My husband Abbas for his unlimited love, support, endurance, and encouragement.

To my candle, my children Ruqia, and Mohammed.

To my sisters, brothers, and to everyone who helped me from a friend or fellow...



**Hind Ibrahim Mohammed**

# Acknowledgements

My thanks are above all to God Almighty, who guided my steps towards the path of knowledge and without his help and blessing.

This thesis would not be advanced or you will see the light.

I express my sincere appreciation to my supervisor's Assist. Prof.

Dr. Jumana W. Salih for providing his ideas, inspiration, and continuous support for me during my studies.

I am very grateful to all members and professors of the Computer Science Department at Diyala University.

Finally, I would have never been able to finish my message without especially mention my sisters (Alyaa, Dhamaya, Shaimaa, Esraa, Baidaa, and Nidaa) and support my family and my husband.



**Hind Ibrahim Mohammed**

# ABSTRACT

The process of identifying each letter separately is very important to understand. With this, sign language recognition has become an important technology in artificial intelligence (AI) and machine learning (ML).

This thesis presents two proposed systems for static hand gesture recognition (HGR) based on ML and Deep Learning (DL) algorithms in which several steps are used in the form of phases; image acquisition, image preprocessing, feature extraction, and classification. In the first proposed system, a histogram of oriented gradients (HOG) is utilized for extracting features from each image and then a multi-class support vector machine (MSVM) is applied using the result of the HOG of images to perform the classification process. In the second proposed system, the convolution neural network (CNN) is used through which recognition of static hand gestures is accomplished according to a special structure of this algorithm that consisting of several layers.

The Previous works and researches in that field had a lot of complexity with different accuracy. The obtained results, the second proposed system which adopted DL by using the CNN model outperforms the first system in terms of performance and accuracy, the accuracy rate obtained from the second proposed system was (99.71%) for American Sign Language (ASL) and (99.03%) for Arabic sign language (ArSL), While the accuracy rate obtained from the first proposed system was (95.58%) and (96.16%) for ArSL.

# List of Contents

Subject	Page No.
List of Contents.	I-IV
List of Abbreviations.	V-VI
List of Tables.	VII-VIII
List of Figures.	IX-XI
List of Algorithm.	XII
<b>Chapter One: General Introduction</b>	
1.1 Introduction.	1
1.2 Overview of Hand gesture Recognition.	2-4
1.3 Related Works.	4-8
1.4 Problem Statement.	9
1.5 Aim of the Thesis.	9
1.6 The Organization of the Study.	10
<b>Chapter Two: Theoretical Background</b>	
2.1 Introduction.	11
2.2 Data Acquisition.	11
2.3 Image Preprocessing .	12
2.3.1 Scale Modification	12-13
2.3.2 Conversion RGB images to Grayscale	13
2.4 Feature Extraction.	13-14
2.4.1 Histogram of Oriented Gradients (HOG).	14- 17
2.4.2 Features Normalization.	17
2.4.3 Principal Component Analysis (PCA).	18

2.5 Gesture classification .	19
2.6 Machine Learning Algorithms (ML).	19-20
2.6.1 Support Vector Machine (SVM) .	20-22
2.6.1.1 Linear SVM	22-23
2.6.1.2 Non-Linear SVM	23-25
2.7 Deep Learning Algorithms (DL).	25-26
2.7.1 Convolutional Neural Network (CNN) .	26-27
2.7.2 Basic Structure of CNN.	27-34
2.7.3 The Network Training .	34-35
2.7.4 Back Propagation Algorithm (BP)	36
2.7.4.1 Adaptive Momentum (Adam)	36
2.7.4.2 Nesterov-accelerated Adaptive Moment (Nadam)	36-37
2.7.4.3 Root Mean Square Propagation (RMSProp)	37-38
2.8 Evaluation Measures .	39-41
<b>Chapter Three: Proposed System Design</b>	
3.1 Introduction.	42
3.2 The Proposed System Design.	43- 44
3.3 Image Acquisition Stage.	44-45
3.4 Image Preprocessing Stage.	45
3.5 The First Proposed system: (Multi-Class Support Vector Machine Algorithm (MSVM)).	46
3.5.1 Feature Extraction.	47-49
3.5.2 Z- Score Normalization.	49
3.5.3 Principal Component Analysis (PCA).	50-51
3.6 Recognition Stage Using MSVM Algorithm.	51-52

3.6.1 MSVM Training.	52-53
3.6.2 MSVM Testing.	53
3.6.3 Recognition for ASL and ArSL hand gesture image in the First Proposed system (Performance Measurement).	53
3.7 The Second Proposed System using CNN Algorithm.	53- 56
3.7.1 Feature Extraction Stage .	56
3.7.2 Design Convolution Neural network (CNN) Structure.	56-60
3.7.3 CNN Training .	60- 63
3.7.4 CNN Testing .	63-64
<b>Chapter Four: Experimental Results and Discussion</b>	
4.1 Introduction.	65
4.2 Implementation Environment.	65
4.3 Dataset Acquisition .	66-68
4.4 Evaluation of First Proposed System .	68
4.4.1 Result of Image Pre-Processing.	68
4.4.2 Results of Implementation the Feature Extraction using Histogram of Oriented Gradients (HOG) Algorithm.	68-72
4.4.3 Results of Implementation of Z- Score Normalization.	72-75
4.4.4 Results of Principal Component Analysis (PCA).	75-78
4.4.5 Result of Training and Testing Using MSVM Algorithm.	78
4.4.5.1 Result of MSVM Training	78-79
4.4.5.2 Result of MSVM Testing	80-83
4.4.6 Hand Gesture Recognition in the First Proposed System.	83

4.5 Evaluation of the Second Proposed System (Using CNN).	84-86
4.5.1 Result of the Second Proposed System (Using CNN).	86-88
4.5.2 Result of the CNN Training.	89-91
4.5.3 Result of the CNN Testing.	91-93
4.5.4 Hand Gesture Recognition in the Second Proposed System	93-94
4.6 Comparison, Between the First Proposed System with the Second Proposed System.	94-95
4.7 Proposed system vs. Related Works.	95-97
<b>Chapter Five: Conclusions and Suggestions for Future Work</b>	
5.1 Conclusions.	98-99
5.2 Suggestions for Future Work.	99-100
<b>References</b>	
	101-112
الخلاصة باللغة العربية	113

## *List of Abbreviations*

<b>Abbreviations</b>	<b>Description</b>
<b>1V1</b>	One-Versus-One.
<b>1VR</b>	One-Versus-Rest.
<b>AC</b>	Accuracy.
<b>Adam</b>	Adaptive Momentum.
<b>AF</b>	Activation Function.
<b>AI</b>	Artificial Intelligence.
<b>ANN</b>	Artificial Neural Networks.
<b>ArSL</b>	Arabic Sign Language.
<b>ASL</b>	American Sign Language.
<b>BP</b>	Back Propagation.
<b>CM</b>	Confusion Matrix.
<b>CNN</b>	Convolution Neural Network.
<b>CV</b>	Computer Vision.
<b>DA</b>	Discriminant Analysis.
<b>DCC</b>	Deep Convolution Network.
<b>DCT</b>	Discrete Cosine Transform.
<b>DL</b>	Deep Learning.
<b>DNN</b>	Deep Neural Network.
<b>EEG</b>	Electroencephalography.
<b>EOH</b>	Edge Orientation Histogram
<b>FC</b>	Fully Connected.
<b>FN</b>	False Negative.
<b>FP</b>	False Positive.
<b>HCI</b>	Human-Computer Interaction.
<b>HGR</b>	Hand Gestures Recognition.



<b>HMM</b>	Hidden Markov Models.
<b>HOG</b>	Histogram Of Oriented Gradients.
<b>KNN</b>	K-Nearest Neighbor.
<b>LBP</b>	local binary patterns.
<b>LR</b>	Logistic Regression.
<b>ML</b>	Machine Learning.
<b>MLP</b>	Multilayer Perceptron.
<b>MSE</b>	Mean Squared Error .
<b>MSVM</b>	Multi-Class Support Vector Machine.
<b>Nadam</b>	Nesterov-Accelerated Adaptive Moment.
<b>NB</b>	Naïve Bayes.
<b>NN</b>	Neural Network.
<b>PCA</b>	Principal Component Analysis.
<b>PReLU</b>	Parametric Relu.
<b>ReLU</b>	Rectified Linear Unit.
<b>RF</b>	Random Forests.
<b>RGB</b>	Red Green Blue.
<b>Rmsprop</b>	Root Mean Square Propagation.
<b>ROI</b>	Region Of Interest.
<b>SVM</b>	Support Vector Machines.
<b>Tanh</b>	Hyperbolic Tangent Function.
<b>TN</b>	True Negative.
<b>TP</b>	True Positive.

## *List of Tables*

<b>Table No.</b>	<b>Description</b>	<b>Page</b>
<b>Table (1.1)</b>	Comparison of Related Works	8
<b>Table (4.1)</b>	The Distribution of Hand Gesture Images.	68
<b>Table (4.2)</b>	Example of HOG Features for ASL Images.	70
<b>Table (4.3)</b>	Example of HOG Features for ArSL Images.	71
<b>Table (4.4)</b>	Example of Histogram for Original Features to ASL..	72-73
<b>Table (4.5)</b>	Example of Histogram for Original Features to ArSL.	74
<b>Table (4.6)</b>	Example of PCA Features for ASL Images.	75-76
<b>Table (4.7)</b>	Example of PCA Features for ArSL.	76-77
<b>Table (4.8)</b>	Comparing accuracy with data ratios for ASL.	79
<b>Table (4.9)</b>	Comparing accuracy with data ratios for ArSL.	79
<b>Table (4.10)</b>	The AC of test data for all classes for ASL.	80-81
<b>Table (4.11)</b>	The AC of test data for all classes for ArSL.	81-82
<b>Table (4.12)</b>	Proposed System Design CNN Layers for ASL.	84
<b>Table (4.13)</b>	Proposed System Design CNN Layers for ArSL.	86
<b>Table (4.14)</b>	The AC and loss for each training in 10-Epoch for ASL.	87
<b>Table (4.15)</b>	The AC and loss for each training in 10-Epoch for ArSL.	87
<b>Table (4.16)</b>	Comparison of the layers of the CNN and the AC rate for ASL.	88

<b>Table (4.17)</b>	Comparison of the layers of the CNN and the AC rate for ArSL.	88
<b>Table (4.18)</b>	Comparing the AC with number of epoch for ASL.	90
<b>Table (4.19)</b>	Comparing accuracy with data ratios for ASL.	93
<b>Table (4.20)</b>	Comparing accuracy with data ratios for ArSL.	93
<b>Table (4.21)</b>	The AC comparison between MSVM and CNN algorithms.	95
<b>Table (4.22)</b>	Comparing the first proposed system with previous studies for ASL.	96
<b>Table (4.23)</b>	Comparing the first proposed system with previous studies for ArSL.	96
<b>Table (4.24)</b>	Comparing the second proposed system with previous studies for ASL.	97
<b>Table (4.25)</b>	Comparing the second proposed system with previous studies for ArSL.	97

## *List of Figures*

<b>Figure No.</b>	<b>Description</b>	<b>Page</b>
<b>Figure (1.1)</b>	Sensor-Based Data Glove.	3
<b>Figure (1.2)</b>	Using Computer Vision Techniques.	3
<b>Figure (2.1)</b>	Method of bicubic interpolation.	13
<b>Figure (2.2)</b>	Cells are grouped into larger spatial regions (Blocks).	16
<b>Figure (2.3)</b>	Classification of the most common ML algorithms.	20
<b>Figure (2.4)</b>	The SVM hyperplane between two classes.	21
<b>Figure (2.5)</b>	Multi-Class Support Vector Machine.	22
<b>Figure (2.6)</b>	The relationship between AI, DL and ML.	26
<b>Figure (2.7)</b>	The General Structure of the CNN System.	27
<b>Figure (2.8)</b>	Convolutional Layer.	28
<b>Figure (2.9)</b>	Sigmoid Function.	30
<b>Figure (2.10)</b>	Hyperbolic Tangent Function.	31
<b>Figure (2.11)</b>	types of ReLU Transformation.	32
<b>Figure (2.12)</b>	Two Classic Pooling Methods.	34
<b>Figure (2.13)</b>	Dropout Neural Network.	35
<b>Figure (2.14)</b>	A confusion matrix.	39
<b>Figure (3.1)</b>	Stages of Hand gesture recognition (HGR).	42
<b>Figure (3.2)</b>	Block Diagram of General Proposed Systems.	43
<b>Figure (3.3)</b>	The block diagram for the first proposed system.	46
<b>Figure (3.4)</b>	Block Diagram of Extraction Features using HOG Algorithm.	49

<b>Figure (3.5)</b>	Block Diagram of Second Proposed System.	54
<b>Figure (3.6)</b>	Structure of the CNN algorithm.	57
<b>Figure (4.1)</b>	The Alphabets of the American Sign Language.	60
<b>Figure (4.2)</b>	The Alphabets of the Arabic Sign Language.	66
<b>Figure (4.3)</b>	Randomly Samples After Preprocessing.	69
<b>Figure (4.4)</b>	HOG Features for ASL Images.	70
<b>Figure (4.5)</b>	HOG Features for ArSL Images.	72
<b>Figure (4.6)</b>	Histogram for Original Features to ASL.	73
<b>Figure (4.7)</b>	Histogram for Original Features for ArSL.	75
<b>Figure (4.8)</b>	PCA Features for ASL.	76
<b>Figure (4.9)</b>	PCA Features for ArSL.	78
<b>Figure (4.10)</b>	CM for ASL Using MSVM.	82
<b>Figure (4.11)</b>	CM for ArSL Using MSVM.	83
<b>Figure (4.12)</b>	AC and Loss Validation Change Against Training Epochs (75) for ASL.	89
<b>Figure (4.13)</b>	AC and Loss Validation Change Against Training Epochs for ArSL.	90
<b>Figure (4.14)</b>	AC and Loss Validation Change Against Training Epochs (150) for ASL.	91
<b>Figure (4.15)</b>	CM for ASL Using CNN.	92
<b>Figure (4.16)</b>	CM for ArSL Using CNN.	92
<b>Figure (4.17)</b>	The chart shows the AC comparison between MSVM and CNN algorithms.	95

## *List of Algorithms*

<b>Algorithm No.</b>	<b>Description</b>	<b>Page</b>
<b>Algorithm (3.1)</b>	Conversion RGB image to gray-scale.	45
<b>Algorithm (3.2)</b>	Feature Extraction using HOG Algorithm.	48
<b>Algorithm (3.3)</b>	PCA to Extract Feature.	50-51
<b>Algorithm (3.4)</b>	Multi-Class Support Vector Machine Classifier.	54
<b>Algorithm (3.5)</b>	CNN Training Algorithm to Classify Hand Gesture Images.	55
<b>Algorithm (3.6)</b>	Softmax layer function.	60

# Chapter One

## GENERAL INTRODUCTION

# CHAPTER ONE

## GENERAL INTRODUCTION

### 1.1 Introduction

Gesture recognition in today's technologies is an emerging topic. The major objective of this is to use mathematical algorithms for human-computer interaction (HCI), Typically Gestures may arise from any movement or condition of the body, however generally come from the face or the hand ([1], [2]).

Hand gestures play an important role in communicating human thoughts and feelings, and Sign language is a formal type of hand gestures that are used as a communication device, including visual movements and signs. For the culture of the deaf and speech-impaired. Sign language makes it possible to use several parts of the body such as the fingers, hand, arm, head, torso, or face to communicate details. In the hearing population, sign language is not common, however, although fewer are capable of understanding it. Which creates a real obstacle of contact between both the deaf and the rest of humanity, a question that still has to be completely addressed ( [3], [4]).

The deaf and dumb people have been disconnected from the community, and it is impossible for average people to learn sign language. Not only for deaf and dumb individuals, sign language learning has been adopted, but also as a medium for common people to communicate with them ([5], [6]).

Many of the general assets of sign languages around the world are held by ArSL. Its documentation, though, is in a comparatively early process. ArSL also has many nation versions and dialects as most other sign languages [7].

The literature includes several recommended solutions for the recognition of the automated sign language. ArSL, however, has garnered little attention from academics, unlike American Sign Language (ASL) [7].

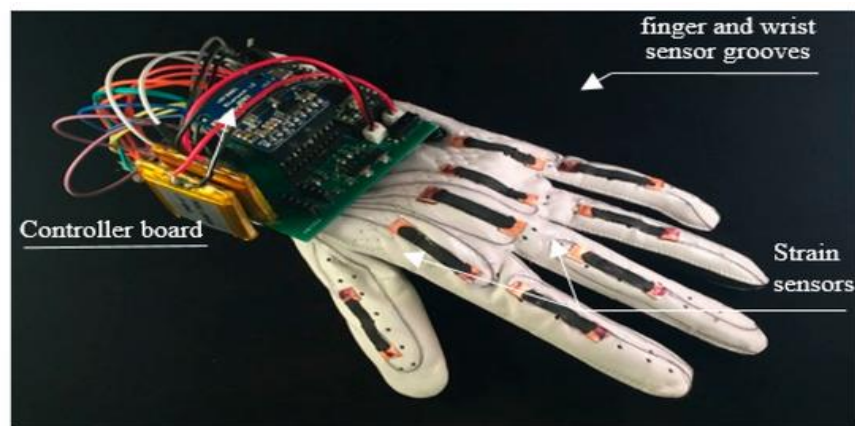


## 1.2 Overview of Hand gesture Recognition

Gesture recognition is a sufficient way to understand and follow human language and to assist in communication and interaction between the user and the computer. Gesture recognition is useful for communicating what they cannot communicate with speech or writing text. The best way to express something meaningful is with gestures [1]. Hand gestures recognition (HGR) is still a large research field that is classified according to the meaning of a gesture and the technologies for implementing these gestures [8]. The type of HGR system that is developed is determined by different taxonomies: environmental factors, a system for capturing gestures could be more or less effective, depending on a variety of factors, including the skills of the individual performing the gesture, the effectiveness of the capture systems, the type of gesture (static or dynamic), and the purpose for which the system was designed [9]. Virtual worlds, smart surveillance, sign language translation, medical systems, and other domains all have HGR applications. However, one of the most important applications have developed is sign language based on machine learning (ML) algorithms using hand gestures [10].

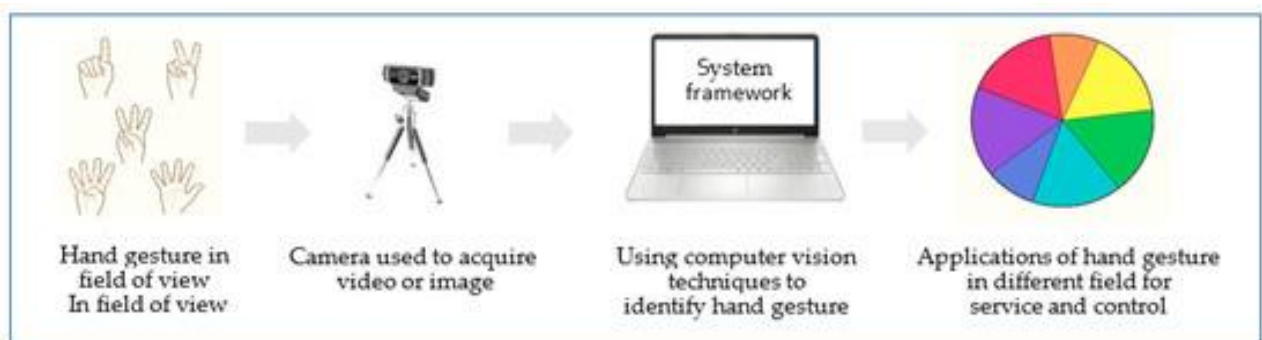
Two types of HGR techniques have been described, recognition based on vision and sensor which linking one or more types of sensors, the gesture data is collected using sensor-based recognition. These sensors are connected to a hand that records the hand's positioning and then analyzes the data gathered for gesture recognition. The data glove is an example of sensor-based gesture recognition. Sensors are shown in Figure (1.1), there are certain limitations to base recognition. First, it is necessary to establish the correct hardware, which is really costly. Second, it impedes normal hand gestures. Thus, the weaknesses of sensor-

based techniques necessitated the creation of vision-based recognition techniques [11].



**Figure (1.1):** Sensor-Based Data Glove [11].

The images of the hands are captured using vision-based methods, which use one or more cameras. Different vision types are shown in Figure (1.2). Stereo cameras, monocular cameras, fish eye cameras, flight time cameras, infrared cameras, and other types of cameras are all available that can be used to capture images. Vision-based techniques use different algorithms for image recognition to achieve hand posture and hand movement. To get the hand position, some vision-based techniques use colored markers. Yet, the limits of vision-based recognition is often influenced by shifts in illumination and cluttered backgrounds, see ([12], [13]).



**Figure (1.2):** Using Computer Vision Techniques [11].

In general, gestures are divided into two categories: static gestures and dynamic gestures. Hand forms are typically used to describe static gestures, while hand motions are used to describe dynamic gestures[14].

HGR systems that rely on geometric features such as fingertips, finger directions, and hand contours are only accurate in situations where they are not occluded or the lighting is not too dark. Shape, color, and texture are present, but they are not sufficiently important for recognition. To define features, images or transformed images are used as the input. The recognizer then implicitly and automatically selects features from the image or transformed image [14].

### 1.3 Related Work

Advanced tools and strategies have greatly enhanced ML algorithms that is a set of learning methods designed to represent structured data that has successfully been applied to the field of image classification to the extent that they can overwhelm human performance. This section reviews the previous studies that used convolutional neural network (CNN) and Support Vector Machine (SVM) to recognize hand gestures for Different Datasets.

**S. Nagarajan and T. S. Subashini (2013) [5]:** proposed a consistent HGR system for ASL based on the features of Edge Orientation Histogram (EOH) and multi-class support vector machine (MSVM). The database of the image contains a total of 720 images in 24 categories of the American Sign Language (ASL) alphabet, each category contains 30 images. The input sign language alphabets' edge graph count is extracted features and applied to the MSVM for categorization. The average accuracy of the system was 93.75%, the system failed to classify some alphabets. as well as the absence of the letters "J" and "Z" in the data set used because these two gestures are dynamic and include movement.

**O. K. Oyedotun and A. Khashman (2016)** [15]: proposed application of deep learning (DL) for the complete (24 ASL) hand gestures acquired from Thomas Moeslund database to the topic of hand gesture recognition. The complicated role of classifies hand signals at reduced error levels has been demonstrated by more biologically based and deep neural networks, including convolution neural networks (CNN) and stacked denoizing autoencoders. The networks considered are based on the data collected and checked. Recognition scores of 91.33% and 92.83%, the data does not contain the two-letter 'j' and, 'z' and the model can obtain higher accuracy if used in another data set

**V. Bheda and N. D. Radpour (2017)** [16]: proposed a model for using deep convolutional networks (DCN) to classify images, the data set ASL was a collection of 25 images for each alphabet and the digits. CNN architecture consisting of multiple convolutional and dense layers. The accuracy was 67% of letters of the alphabet, and 70% for digits. The number of images for each character can be increased to further improve the system's accuracy.

**S. Masood et al. (2018)** [17]: presented the application of CNN for recognizing hand gestures. used 36 different categories, 26 classes for ASL and 10 classes for Numerals (0-9). Accuracy was 96%. The system failed to classify the zero and 'W' alphabet as 'O' and six respectively.

**Reema Alzohairi et al. (2018)** [18]: aimed to recognize ArSL alphabets automatically using Methodology focused on images Especially, An accurate ArSL alphabet recognizer is being designed through the use of numerous visual descriptors. In the extraction process, the extracted visual descriptors are used as input for the One Versus Rest analysis (1VR). As a result, the ArSL gesture models learned 1VR using histogram of oriented gradients (HOG) descriptors are

used. The database contains 30 classes (7 images for each character), image is focused and resize to 200x200 pixels in size. The system accuracy 63.5 %. The system has an issue with a limited dataset for the training algorithm. The number of images for each character can be increased to further improve the system's accuracy.

**R. Ahuja et al. (2019) [19]:** proposed a model that used CNN layers and digital image processing techniques. Open CV was used to track down additional execution methods such as image preprocessing. The archive, which was used to test 24 ASL hand gestures, contains 47,445 photographs, of which 33000 (70%) were used in the training collection and 14445 (30%) were used for testing. The results showed that was accurate at 99.7%. It is attributed that the system detected 24 a letter instead of 26 the absence of the two letters "J" and "Z" in the data set used in the model.

**S. Hayani et.al (2019) [20]:** proposed a model using CNN. This system will detect numbers and letters when fed with a real dataset. Utilized the dataset of images which contained 2,030 images of numbers, and 5,839 images of the 28 different ArSL classes, and the result was an accuracy of 90.02 percent. More accurate results can be obtained by increasing the number of CNN layers and the number of images used for each letter.

**T. Goswami and S. R. Javaji (2020) [21]:** suggested a model that relies on a CNN to recognize and classify hand gestures. The dataset uses 24 classes (27,455 images) to ASL (A-Z), with size (28x28). DL technology based on CNN learns and automatically extracts features to classify each gesture. The proposed model has a test accuracy of 99%. It is attributed that the system detected 24 a letter

instead of 26 the absence of the two letters "J" and "Z" in the data set used in the model.

**M. M. Kamruzzaman (2020)** [22]: proposed a system to detect hand signs with CNN automatically to dataset (ArSL), the system was trained for 100 epochs by optimizer with a cost function. The system is then connected to its signature stage, where a hand sign has been translated with 90% accuracy to Arabic speech. That can be improved by increasing the number of images, as only 100 images were used for each letter.

**A.Sharma et al. (2020)** [23]: proposed a system that used Many various techniques for pretreatments such as HOG, local binary patterns (LBP), and principal component analysis (PCA). This dataset ASL contains 29 classes (3000 image). These methods were successfully implemented to obtain effective results accuracy of Multilayer Perceptron (MLP) 96.96%, K-Nearest Neighbor (KNN) 95.81%, Random Forests (RF) 92.69%, Support Vector Machines (SVM) 85.25%, Logistic Regression (LR) 84.59%, and Naïve Bayes (NB) 72.23%.

Table (1.1): Comparison of Related Works.

No.	Author(s),Year	Ref. No.	Algorithm for Classification	Dataset Size (Images Number)	Accuracy
1.	S. Nagarajan and T. S. Subashini (2013)	[5]	MSVM	720 images in 24 categories ASL	93.75%,
2.	O. K. Oyedotun and A. Khashman (2016)	[15]	CNN	1440 for training , 600 for testing ASL	92.83%
3.	V. Bheda and N. D. Radpour (2017)	[16]	CNN	650 Images , 25 images from 5 people for each alphabet ASL	67%
4.	S. Masood et al. (2018)	[17]	CNN	2524 ASL gestures	96%
5.	Reema Alzohairi et al. (2018)	[18]	MSVM	210 ArSL	63.5 %.
6.	R. Ahuja et al. (2019)	[19]	CNN	47,445 images for 24 classes ASL	99.7%
7.	S. Hayani et.al (2019)	[20]	CNN	5839 images of 28 class ArSL	90.02%
8.	T.Goswami and S. R. Javaji (2020)	[21]	CNN	27,455 images for 24 classes ASL	99%
9.	M. M. Kamruzzaman (2020)	[22]	CNN	100 images for each alphabet (32 classes) ArSL	90%
10.	A.Sharma et al. (2020)	[[23]	MSVM	3000 images ASL	85.25

## **1.4 Problem Statement**

About 70 million people (deaf and dumb) use sign language as their first or mother tongue all over the world and unfortunately they cannot communicate with the general public because they do not understand the meaning of sign language gestures and on the other hand they are unable to understand natural language.

Indeed, it is very important to support this category of society in view of the great development in the world of technology and software, and as a result, it is necessary to prepare systems capable of translating signals into text or speech. If these systems are put in place, it will greatly help them to understand what is going on around them in an easy and simple way.

For the sake of the above, several researchers have proposed the development and implementation of automated systems or human-computer interaction (HCI) to help deaf people and the general public communicate.

## **1.5 Aim of the Thesis**

This thesis aims to build a strong to recognize hand gesture system for ASL and ArSL sign language to help the deaf and dumb people more easily with computer vision applications using multi-class support vector machine (MSVM) classing was designed and applied as the most important algorithm of ML algorithms in the first proposed system. Furthermore, the CNN model is utilized in the second proposed system which is the most powerful algorithm for DL for making a comparison between these techniques to determine the best one in achieving a high degree of accuracy.



**1.6 The Organization of the Thesis**

This thesis is organized into four chapters, in addition to the one already described, and is structured as follows:

Chapter Two describes of the theoretical background of the main systemses that used for the hand gesture recognition based on ML.

Chapter Three presents the details of the proposed recognition and classification algorithms that are used to design the proposed system and the implementation of each one.

Chapter Four gives the experimental results obtained from the implementation of the proposed system.

Chapter Five discusses results, conclusions and lists a number of suggestions for future studies.

# **Chapter Two**

## **THEORETICAL BACKGROUND**

## CHAPTER TWO

### THEORETICAL BACKGROUND

#### 2.1. Introduction

This chapter provides an overview of the theoretical background of the main approaches used in this thesis, Recognition of static hand gestures is a natural medium used for human-computer interaction (HCI), is a very active area of research in computer vision and in Machine Learning (ML) [1].

Various techniques are available for hand gesture recognition in ML and deep learning (DL) as Support vector machine (SVM) and Convolutional Neural Network (CNN) which are the main methods that will be discussed in this thesis to perform feature extraction and classification.

#### 2.2 Data Acquisition

HGR and HCI, in general, have been the subject of extensive research in recent decades. Based on the data collection process, the majority of experiments have taken one of two methods.

In the first approach, the signer is used to interacting with instruments such as data gloves, location trackers, motion sensors, and accelerometers to gather data on hand movements. Furthermore, the second approach was captured using cameras and various imaging instruments (there was no need to touch the signer's body and limit his/her movement). The first solution has the disadvantage of being expensive and inconvenient for the signer. These disadvantages were avoided in the second approach's studies [24].

### **2.3 Image Preprocessing**

The HGR system's initial stage is image preprocessing that removes the unwanted noise.

In the hand gesture preprocessing is the initial step to be performed. Preprocessing requires preparing the images using various techniques such as scale modification, normalization, and noise reduction for feature extraction [25].

#### **2.3.1 Scale Modification**

There are several ways to scale modifications, and one of them is to use interpolation to resize images. It's a method of increasing or decreasing the number of pixels in a digital image. Interpolation is a technique for the size of an image by generating new pixel values and filling in the gaps with other algorithms. The average value of nearly all pixels is used to replace this newly generated pixel. Interpolation is a technique for estimating the significance of unknown data using two or more known data.

There are many types of interpolation, the most important of which are: nearest-neighbor interpolation, Bilinear interpolation, and Bicubic interpolation. Nearest Neighbor Interpolation is the easiest method for interpolation in that every unknown pixel is given an intensity value equivalent to the neighboring pixel. A new scaled image can be created. When an image is expanded, an empty space is created in the original image. The empty spaces will be replaced with the nearest pixels. Bilinear interpolation is another method used for resizing images. Bilinear interpolation is a linear interpolation extension that reduces visual blur when a fractional zoom is measured. The definition of this approach is to select midpoint pixels that are used in the nearest four pixels, the value of which is calculated. The intermediate pixel is created by the closest four-pixel interpolation. The most efficient method is Bicubic Interpolation. Bicubic interpolation operation is chosen when the speed does not important in the

process. The pixel  $B(r', c')$  is formed with interpolation of the nearest  $4 \times 4$  pixels that begin with  $A(r, c)$  and ending to  $A(r+2, c+2)$ . The original image has the two-scale factors denoted as  $S_r$  and  $S_c$  of  $A$ .  $S_r$  and  $S_c$  are the rows and column scale factors correspondingly as shown in Figure (2.1) [32].

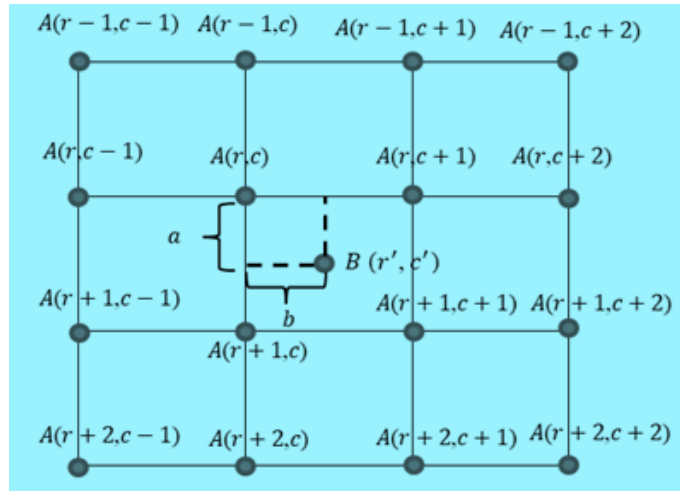


Figure (2.1): Method of bicubic interpolation [26].

### 2.3.2 Conversion RGB images to Grayscale

This operation is done by dividing all RGB values by 255, Equation (2.1) which is used to convert an RGB image to grayscale is as follows[5].

$$\text{Grayscale} = (\mathbf{R} + \mathbf{G} + \mathbf{B}) / 255 \quad \dots\dots (2.1) [5].$$

Where R: Red, G: Green, and B: Blue.

### 2.4 Feature Extraction

Selecting accurate features is important for gesture recognition because the forms, motion, and textures in human gestures are plentiful. Self-occlusion and lighting conditions can make geometric features such as fingertips, finger directions, and hand contours inaccessible and inaccurate. Another important thing to keep in mind is that color, silhouette, and texture cannot perform recognition tasks on their own. To name features explicitly is difficult images or images that have been transformed or changed in some way are used as input.

The classifier then uses features that have been implicitly and automatically derived [27].

Three different feature extraction methods are used, resulting in three different results for each class, which are compared to one another that were a histogram of oriented gradients (HOG), features extracted from the Discrete Cosine Transform (DCT), and features extracted from a pre-trained CNN. The advantages of the feature extraction methods are the reasons why they are used in this thesis. HOG with ML algorithms and CNN algorithms are used in this thesis[28].

### **2.4.1 Histogram of Oriented Gradients (HOG)**

HOG, created by Dalal and Triggs [29], is one of the most widely used methods for identifying human bodies in computer vision and recognition. It takes a photo and breaks it up into small square cells. It then computes a histogram of gradient directions or image edges based on the differences between cells. Normalized local histograms have been used to make the HOG's contrast-detection feature more accurate, and this is why the HOG's performance doesn't vary as a result of different lighting conditions. Due to simple computations, it is a quick descriptor as compared to other descriptors. It has also been demonstrated that HOG is a good descriptor for detection [30]. Moreover, HOG feature extraction has the advantage of lower complexity in terms of computational time and greater accuracy as compared to popular feature extraction [31]. HOG feature extraction mainly includes two stages (Histogram Extraction of Oriented Gradient and Construction of HOG Descriptor ([28], [32])).

#### **1. Histogram Extraction of Oriented Gradient**

At this point, based on each pixel in the image, gradients are extracted and turned into an angular histogram, which is then used as an image texture feature

vector. At pixel, the horizontal and vertical derivatives (i,j) of image I(i,j) are computed as follows:

$$G_i(i, j) = I(i + 1, j) - I(i - 1, j) \dots\dots\dots (2.2)$$

$$G_j(i, j) = I(i, j + 1) - I(i, j - 1) \dots\dots\dots (2.3)$$

The magnitude of gradient is:

$$G(i, j) = \sqrt{G_i(i, j)^2 + G_j(i, j)^2} \dots\dots\dots (2.4)$$

The direction of gradient is:

$$\alpha_0(i, j) = \tan^{-1} \left[ \frac{G_j(i, j)}{G_i(i, j)} \right] \dots\dots\dots (2.5)$$

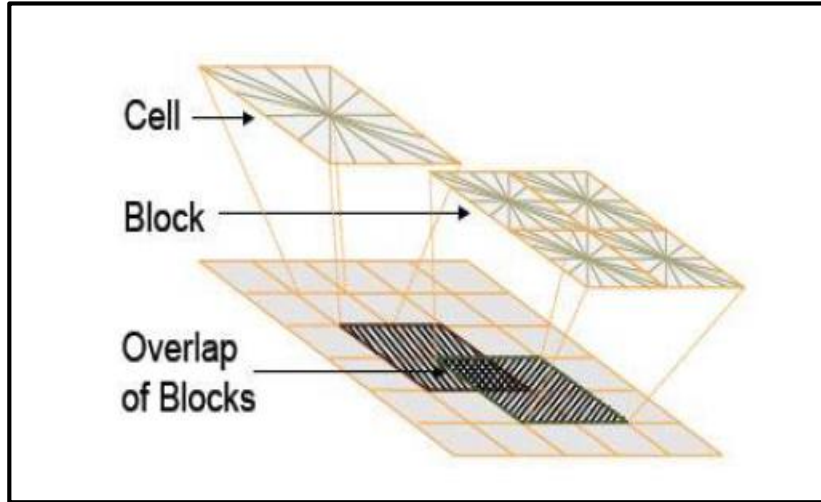
where  $G_i(i, j)$ ,  $G_j(i, j)$  are at pixel (i,j), the horizontal and vertical derivatives are given,  $\alpha_0 \in \left[ \frac{-\pi}{2}, \frac{\pi}{2} \right]$  [33].

## 2. Construction of HOG Descriptor

The input image is divided into small square cells, and the differences are used to compute a histogram of gradient directions or image edges [13]. Based on the orientation of the gradient element centered on it, each pixel calculates a weighted vote for an edge orientation histogram channel, and the votes are accumulated into orientation bins over local spatial regions (cells). The orientation bins are evenly spaced from 0 to 180 degrees (“unsigned”) [15].

Block normalization is used to improve invariance to lighting, shadowing, and other factors by correcting local contrast differences. It means that an intensity measure is computed over a slightly larger spatial area, referred to as a block, and the result is then used in every block to normalize the cell histograms. The four adjacent block feature vectors are then linked to form a superb block. Finally, the HOG feature of the image is built by scanning the image block by block by combining the vectors of all superblocks [15]. As a result of the overlapping of these blocks, some cells are included in more than one block [16].

The process of grouping cells into larger spatial(blocks) regions is depicted in Figure (2.2).



**Figure (2.2):** Cells are grouped into larger spatial regions (Blocks) [34].

For block normalization, there are several different normalization schemes. When we defined  $v$  as a vector non-normalized containing all histograms in a given block,  $\|v\|_1, \|v\|_2$  .be its 1-norm and 2- norm, respectively and  $e$  be some small constant (its value has no influence on the results), then the normalization factor can be obtained from equations (2.6) [34]:

$$L_{2-norm} : f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

$$L_{1-norm} : f = \frac{v}{(\|v\|_1 + e)} \quad \dots\dots\dots (2.6)$$

$$L_{1-sqrt} : f = \sqrt{\frac{v}{(\|v\|_1 + e)}}$$



The HOG of sample images is subjected to principal component analysis (PCA). A linear SVM classifier is trained using selected principal components. During the classification process, a feature vector of a testing image is extracted using HOG-PCA, and the qualified classifier is then used to predict the sign using HOG-PCA features from the testing image [3].

### 2.4.2 Features Normalization

Normalization is a data operation method used to improve a classifier's accuracy. Following the feature subtraction process, various normalization methods may be used to improve the performance [35]. This procedure entails translating attributes into a type that places them within a particular range. The normalization method is particularly useful for classification algorithms such as the K-Nearest Neighbor (KNN) and Neural Network algorithms (NN), where the process of features normalization aids in the acceleration of the training step. There are several methods for feature normalization [36] such as z-score normalization which was used in the proposed system.

- **Z-Score Normalization**

This method begins by calculating the average values of the features, followed by calculating the standard deviation of the features, and then normalization is performed using this equation [35]:

$$Z = \frac{f_i - \mu}{\sigma} \quad \dots\dots\dots (2.7)$$

where  $f_i$  refers for feature values,  $\mu$  for mean of features, and  $\sigma$  for standard deviation, which can be calculated using the equation below:

$$SD \text{ or } \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (f_i - \mu)^2} \quad \dots\dots\dots (2.8)$$

where  $n$  denoted the number of possible values in each feature [37].

### 2.4.3 Principal Component Analysis (PCA)

PCA means reducing large data sets dimensions by translating a wide range of variables into a smaller one first, then reducing the dimensionality of the smaller one by leaving a large majority of the information in the original wide set[38].

Naturally, reducing the number of variables in a data set decreases accuracy; however, the key to dimensionality reduction is to sacrifice some accuracy for simplicity. Because ML algorithms can analyze data more easily and quickly without having to deal with extraneous variables, smaller data sets are easier to explore and imagine[38].

To summarize, PCA's basic concept is to minimize the number of variables in a data set while retaining as much information as possible [38].

(a) Utilize PCA to reduce the dimensions of the original features and to obtain a set of features.

(b) Calculate the "covariance matrix" from the data using the formula below:

$$C = (X - \bar{x}) (X - \bar{x})^T \quad \dots\dots\dots (2.9)$$

where the hand gesture data matrix is  $X$ , and the mean vector of the data matrix is  $\bar{x}$ .

(c) Use the equation below to compute the matrix of eigenvectors  $V$  and the diagonal matrix of eigenvalues  $D$ :

$$V^{-1}CV = D \quad \dots\dots\dots (2.10)$$

(d) By taking the inner product in the data matrix, sorted eigenvectors, and sorting the eigenvectors in descending order of eigenvalues in  $D$ , the data is projected on these eigenvector directions.

$$\text{Projected data} = [V^T (X - \bar{x})^T]^T \quad \dots\dots\dots (2.11)$$

where  $V$  is of  $n \times n$  dimension, and each row of it is an eigenvector. The features can be obtained [39].

## 2.5 Gesture classification

The HGR system's final stage is gesture recognition, which involves feeding an input feature vector extracted from the feature extraction process into a suitable classifier. ML based classifiers have grown in popularity in recent years, owing to their flexibility and proclivity to learn new behaviors. ANN, hidden Markov models (HMM), and help vector machines are all well-studied classification algorithms. Since each classification technique has advantages and disadvantages, a classifier's output cannot be determined solely by the algorithm employed. Some algorithms may work well with one collection of data but not with another ([3], [9]).

Gesture recognition presents the most difficult and challenging tasks in the fields of image processing, computer vision , and image analysis [40].

A DL technique based on CNN is used to recognize each gesture by automatic learning and extracting features [32].

## 2.6 Machine Learning Algorithms (ML)

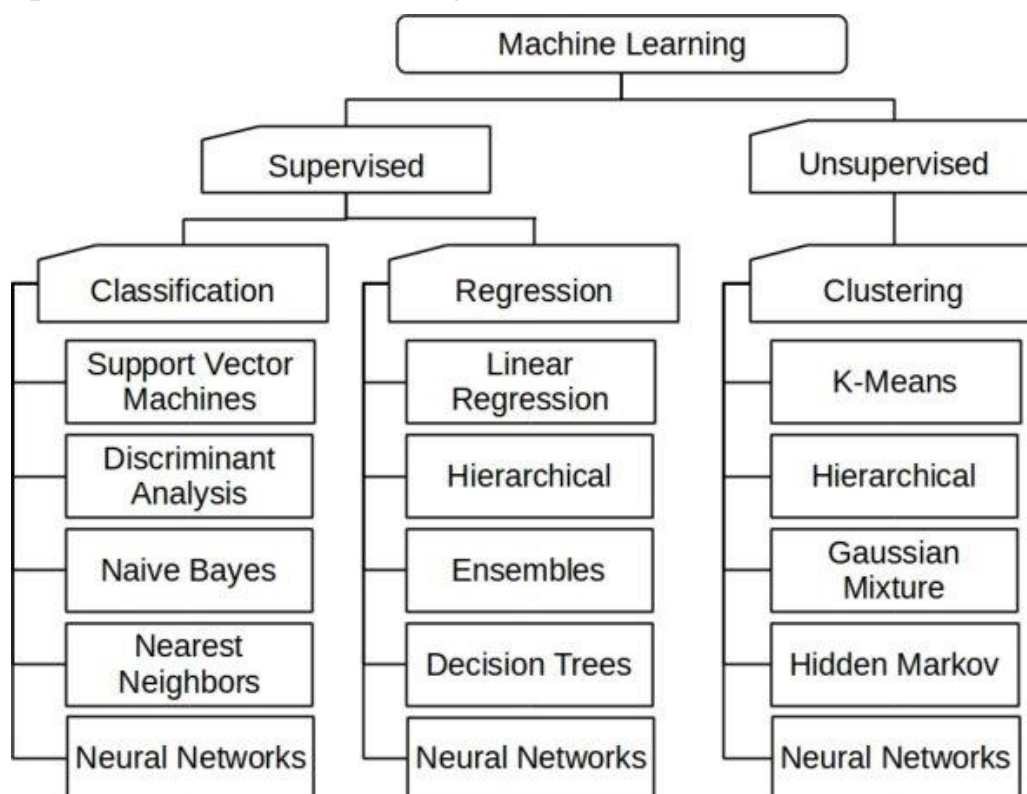
ML is a branch of Artificial Intelligence (AI) that deals with the creation of systems that rely on data or information. Classification is a set of models or functions that distinguishes between classes of data or concepts with the goal of being used as a class prediction model for an unknown class of objects [41].

ML is a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty [42].

Figure (2.3) [43] categorizes classification into supervised and unsupervised ML techniques. Supervised machine learning is a technique for teaching a system to recognize patterns in input data, which can then be used to forecast future data. Classified training data is used to find a feature by applying supervised machine learning on the training data. Unsupervised machine

learning is used to draw inferences from data that is not labeled. Since there is no classified answer, there is no incentive or penalty weightage for the data to be classified into one of the possible classes [3].

Generally, there are many classifier algorithms, such as SVM, Discriminant Analysis (DA), Naïve Bayes (NB), Random Forest (RF), k- NN, and NN algorithms, with each having a different method to predict or choose the set to which a particular observation belongs [44].



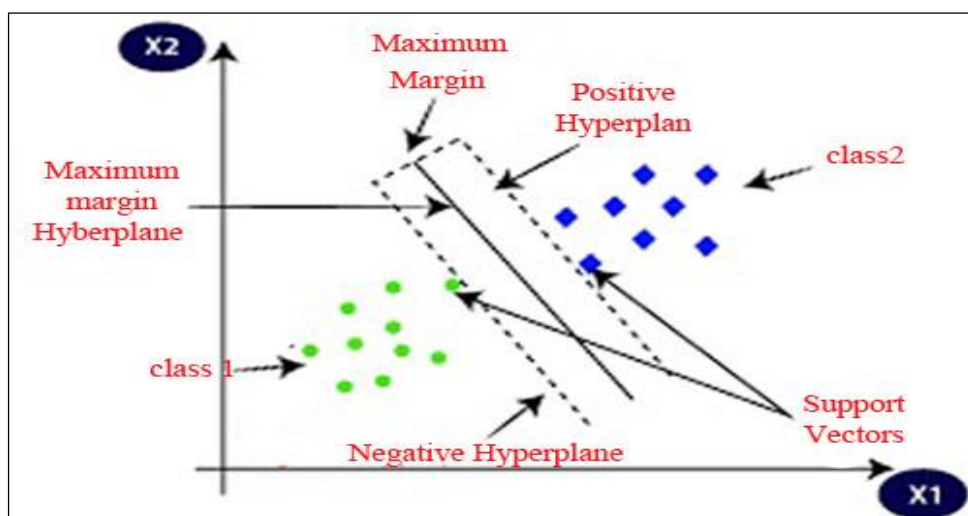
**Figure (2.3):** Classification of the most common ML algorithms [43].

### 2.6.1 Support Vector Machine (SVM)

SVM is a supervised ML approach. It determines the best hyperplane to use to separate the data points. The margin surrounding the separating hyperplane is maximized by SVM. In order to determine the best hyperplane, optimization techniques are used [22]. SVM is a linear classifier that aims to maximize the difference between two sets of data. It classifies by creating an N-dimensional hyperplane that divides the input data into two groups, then probing for an optimal number of dimensions. Supervised learning, for classification and regression tasks, has become more popular in recent years as it avoids overfitting

even for new datasets. Many other classification tasks besides electroencephalography (EEG) signal classification, such as handwritten character recognition, face recognition, and more, have used SVM to be successful. SVM was created by Vapnik [5] in 1998 and is a new type of machine learning that uses the kernel and support vector for learning. By selecting a kernel function, kernel machines provide a flexible framework that may be used for various tasks and domains. Binary classification problems were addressed using a one-versus-all technique, which improved the original SVM design. When it comes to multiclass labels, MSVM considers them to be made up of multiple two-class labels and uses classifiers to overcome issues. To deal with the multiclass problem, a new multiclass classifier is built using the classifier's outputs. The one-versus-all technique entails building one SVM per class that is trained to discriminate samples from one class from samples from all other classes. In general, the maximum output of all SVMs is used to classify an unknown pattern [5].

SVM, a binary typed classifier basis of the supervised learning approach for classifying data into two classes by drawing a hyperplane as Figure (2.4) [45]. When it's about non-linear and multiclass data set, SVM added with an extension and in that case, it's called MSVM as shown in Figure (2.5) [46].



**Figure (2.4):** The SVM hyperplane between two classes[45].

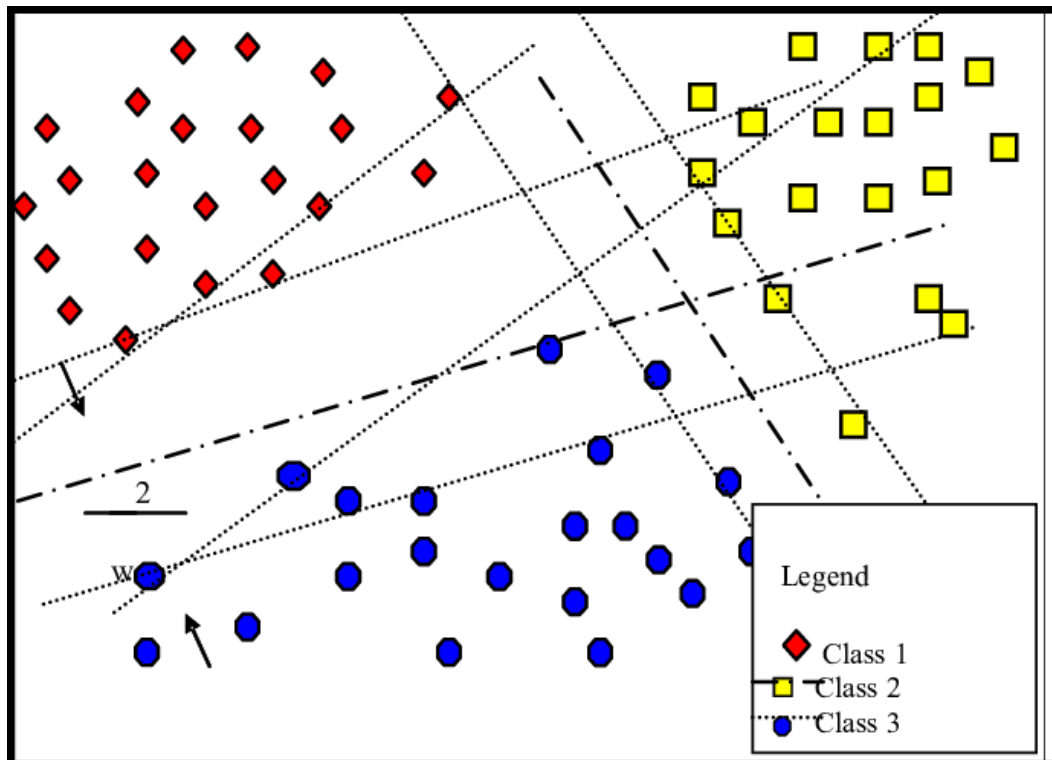


Figure (2.5): MSVM [46].

SVM algorithm is divided into two kinds Linear and non-Linear SVM that illustrate in the following section:

**2.6.1.1 Linear SVM**

If the linear SVM hyperplane, then SVM is defined as linear SVM, e.g. if  $z$  represents pairs training  $(x_i, y_i)$  when  $i=1,2,...,z$ , with category labels  $y$  (1, -1) the direct equation is defining the hyperplane:

$$W \cdot x + b = 0 \quad \dots\dots\dots (2.12)$$

where  $W$  is a vector of weight,  $W = \{w_1, w_2, \dots, w_n\}$ ,  $b$  is a bias, and  $x$  represent attributes. The data classification is considering as equation.

$$f(w \cdot x \cdot y) = sgn(w \cdot x \cdot b) \quad \dots\dots\dots (2.13)$$

where  $sgn$ : the signum function is the derivative of the absolute value function,  $f(x)$  is the function of a hyperplane in  $m$  dimensions thus is got as a series of every point  $x$ .  $x \in \mathbb{R}^m$  that fulfils the equation  $f(x) = 0$  such that the function hyperplane  $f(x)$  functions as a classifier linear predicting class  $y$  for each presented point  $x$ , depended on the subsequent rule decision:

$$W^T \cdot x + b \geq 1 = +1 \quad \dots\dots\dots(2.14)$$

$$W^T \cdot x + b < 0 \text{ for } y = -1 \quad \dots\dots\dots (2.15)$$

Maximizing the margin is constrained optimization trouble that the Lagrange method can solve. A Lagrange multiplier ( $\alpha_i - i$ ) explains every training point  $x_i$ . So, we have

$\alpha_i = 0 \Rightarrow x_i$  has no influence on the hyperplane.

$\alpha_i > 0 \Rightarrow x_i$  these points support vectors that are nearest to the hyperplane.

Also, can calculate weight and bias when obtaining the  $\alpha_i$  value, the weight calculation using the following formula:

$$W = \sum \alpha_i x_i \quad \dots\dots\dots (2.16)$$

Points with ( $\alpha_i = 0$ ) do not consider SVM, therefore the average of the SVM that this is ( $\alpha_i$ ) not equal to zero must take. When support vector with ( $\alpha_i = 0$ ) does not play any role in deciding [47], [48].

### 2.6.1.2 Non-Linear SVM

In most cases, the linear classification does not consider the appropriate classification approach for the non-linear classification used in such situations, where a non-linear kernel function will be used. Linear SVM is fast to train and implement, but with many training examples and not too many features they appear to underperform on complicated datasets. In many applications, non-linear SVM can be more consistent in quality across different problems and the preferred choice, although they lack critical power.

#### A. Kernel Function

A function's kernel is used to transfer testing samples and training to a high-dimensional feature space. This section describes function's kernel to replace functions of mapping. Since the kernel computation is more efficient than the function of mapping, and computation time is normally saved when functions of mapping are replaced with the use of kernels. SVM uses the kernel function  $K(x_n, x_i)$  to transform the raw data space into a higher-dimensional new space. It

employs the dot product transformation function  $\phi(x)$  equation (2.17). The goal is the information that can be collected feasibly and has been translated to a higher dimension. The hyperplane function can be written in formula form (2.18).

$$K(x_n, x_i) = \phi(x_n) \cdot \phi(x_i) \quad \dots\dots\dots (2.17)$$

$$f(x_i) = \sum_{n=1}^N \alpha_n y_n K(x_n, x_i) + b \quad \dots\dots\dots (2.18)$$

where  $\alpha_n$  is a Lagrange multiplier,  $x_n$  is support vector information, and  $y_n$  is a membership class label (+1, -1) with  $n = 1, 2, 3, \dots, N$  where  $N$  is the number of class [47], [49].

**B- Examples of kernels**

The most public example of the SVM higher dimensionality kernel that is commonly used for the SVMs classification are:

1. Linear kernel which is describes as equation:

$$K(x_n, x_i) = (x_n, x_i) \quad \dots\dots\dots (2.19)$$

2. Polynomial kernel which is describe as equation:

$$K(x_i, x_j) = ((x_n, x_i) + C)^d \quad \dots\dots\dots (2.20)$$

3. Radial Basis Kernel Function (RBF) which is describes as equation:

$$K(x_i, x_j) = e^{-\frac{\|x_i, x_j\|^2}{2\sigma^2}} \quad \dots\dots\dots (2.21)$$

4. Sigmoid kernel function which is describe as equation:

$$K(x_i, x_j) = \tanh(K(x_i, x_j) + \theta) \quad \dots\dots\dots (2.22)$$

where  $d, \sigma$  and  $\theta$  are parameters of the specific kernels [47], [49].

**C- Multi-Class SVM**

SVM algorithm was designed to differentiate between two classes, but there are times when more than two classes must be categorized. Multiclass classification problems ( $k > 2$ ) are usually broken down into a series of binary problems that can be solved directly using a generic support vector machine. The one-versus-rest (1VR) and one-versus-one (1V1) systems are



two representative multi-class SVM organization schemes. Both (1VR) and (1V1) are examples of Error-Correcting Output Codes (ECOC), which break down a multi-class problem into a predefined collection of binary issues [50].

### 1. One-Versus-Rest (1VR)

Binary classifiers are separated using the (1VR) technique for k-classification constructs. The binary classifier m-th is trained to use the m<sup>th</sup> type information as examples, with the remaining forms being k - 1 and negative examples. During the evaluation, the binary classifier with the highest output value determines the class label. The (1VR) strategy's unbalanced learning collection is a major flaw. Assuming that total forms have an equal number of examples to train, the rate of plus to minus examples in each individual classifier is 1:k-1. The initial problem's equilibrium is missed in this case [48].

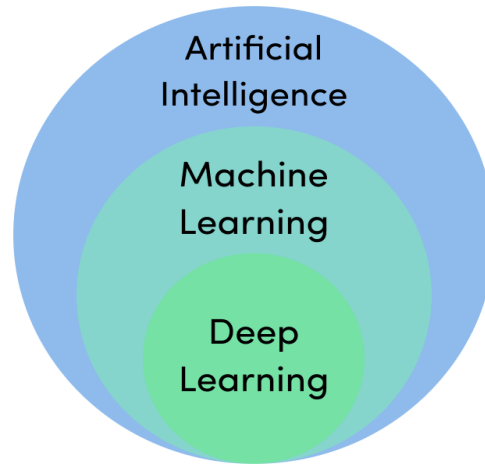
### 2. One-Versus-one (1V1)

Decomposing 1V1 or pairwise classification is another classic technique for multi-type classification. It evaluates all possible pairwise classifiers, resulting in individual binary classifiers  $k \times (k - 1)/2$ . When each classifier is introduced in an example of a test, the one who has won class receives one vote, and the best test is classified with the most votes for class. The classifier sizes created by the 1V1 strategy are almost all larger than the rest strategy's against-one. However, the QP measure is smaller in each classifier, allowing for faster training. However, when compared to the one-versus-rest approach, the 1V1 strategy is more symmetrical [48].

## 2.7 Deep Learning Algorithms (DL)

DL is a research field ML [32] in AI as Figure (2.6) [51] that has Deep neural networks (DNN) are networks capable of unsupervised learning from unstructured or unlabeled input, and it is a form of ML that enables computers

to learn from experience and understand the world in terms of a hierarchy of concepts [22] [52].



**Figure (2.6):** The Relationship Between AI, DL and ML[51].

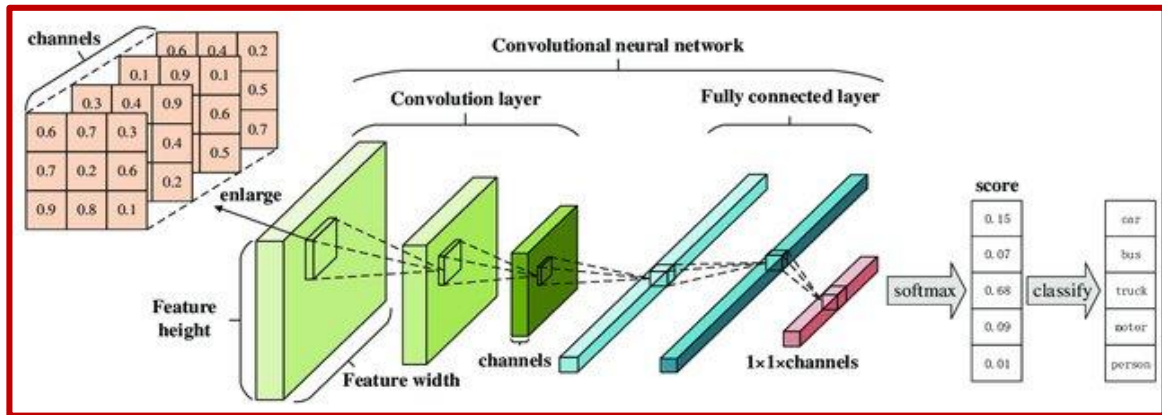
DL was a class of learning methods developed to represent data with complex structures by combining numerous non-linear changes. The NN that is linked to form DNN is the fundamental building blocks of DL [5].

DL algorithms are one promising avenue of research into the automated extraction of complex data representations (features) at high levels of abstraction [5].

DL is often carried out using NN architecture. The term "deep" refers to the total number of layers in the network, the deeper the network. In terms of precision, DL is unrivaled. Advanced tools and methods have dramatically improved DL algorithms to the point where they can outperform human performance, in the near future, because there is very little engineering work required, it will have many more successes [53].

### **2.7.1 Convolutional Neural Network (CNN)**

CNN is an excellent tool for mapping image data into higher-level representations. CNN uses the pixel data it has been given as input to extract features and computes the inference about the pixels [54]. It take this name from the mathematical linear operation between matrixes which is called convolution [55]. The general structure of the CNN system is viewed in Figure (2.7) [56].



**Figure (2.7):** The General Structure of the CNN System [56].

In recent years, CNN had already been used successfully for automatic feature learning. It performed admirably in image categorization, object recognition, and even recognition of human behavior. This outstanding performance can be due to the availability of huge datasets containing millions of samples [3], [11].

### 2.7.2 Basic Structure of CNN

CNN has shown success in picture categorization tasks. Convolutional layer, subsampling or pooling layer, and fully connected layer are the three types of layers in a CNN. Normally, multiple of the above-mentioned layers are stacked to create the whole CNN architecture. The hierarchical characteristics are extracted from a CNN employing the three types of layers. In a CNN, the bottom layers collect low-level features, while the top layers collect and learn higher-level features; this is important for classification tasks, the following explain these layers: [22].

**1. Input Layer:** the input layer contains the pixel values of the image that enter CNN.

**2. Convolution Layers:** The convolution layer as shown in Figure (2.8)[57] is the key element of a convolution network; whose parameters consist of a set of learnable kernels [32]. To get the feature map of this layer, the previous layer's feature map is convoluted with a learned convolution kernel, and the result is

output through an activation function (AF). Each output's graph may be linked to the result of convolution of numerous input feature graphs, allowing the weights to be shared[58].

The number of input images equals the number of output photos, and the resulting image's dimension is reduced. Convolution is applied to each image, and the image's width and length are compressed to acquire more detailed image information[58].

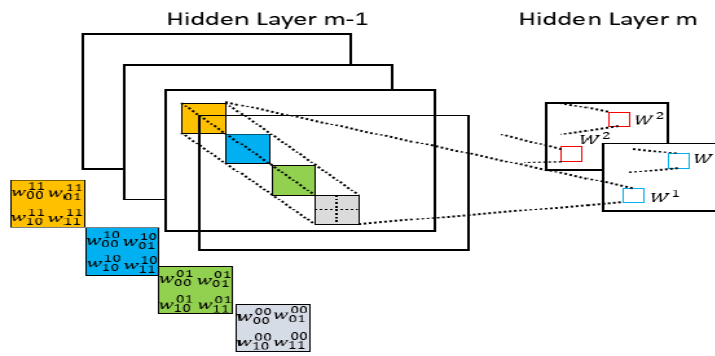


Figure (2.8): Convolutional Layer [57].

All convolution the layers use filters with different size to extract features and pass the feature map to the next layer. The final layer uses filter to extract the prior feature and pass it to the Softmax AF to output [59].

To change the behavior of a convolutional layer, three main parameters must be modified in a CNN. Filter size, stride, and padding are the three parameters. The output size for each convolution layer can be calculated as equations expressed follows:

$$No. \text{ parameters} = \text{output channels} * (\text{input channels} * \text{window size} + 1) \quad \dots\dots (2.23)$$

Where output channels denoted to features maps that result from convolution layer, input channels denoted to previous layer, window size it means filter size, and 1 refer to stride size

$$Output_{size} = \frac{\text{input}_{size} - \text{filter}_{size} + 2 * \text{padding}_{size}}{\text{Stride}_{size}} + 1 \quad \dots\dots (2.24)$$

where  $\text{output}_{\text{size}}$  = the size of the output Convolution layer,  $\text{input}_{\text{size}}$  = the size of input image, padding refers to the extra rows and columns of pixels on an image matrix, Stride denotes how many steps we are moving in each step in convolution. By default, it is one, and  $\text{filter}_{\text{size}}$  = the size of filter [22], [60].

The convolution layer's output is created by combining the multiple feature maps. After that, the output is sent through the AF, which generates nonlinear output [22].

### 3. Non-linear Layer (Activation Function)

Weighted sums of input and biases are used to calculate the total input for a neuro and whether or not it is capable of being fired. It manipulates the presented data using gradient processing, most commonly gradient descent, and then outputs the parameters of the presented data to the neural network. In some journals, these AFs are referred to as a transfer function ([40], [44]).

AFs are used to control the outputs of out NNs in a variety of domains, including object recognition and classification [43]. They are either linear or nonlinear in nature, depending on the function they represent. To convert linear equations into nonlinear equations, the process must join the AF. Below is a list of some of the most popular AF ([40], [44]).

#### A- Sigmoid Function

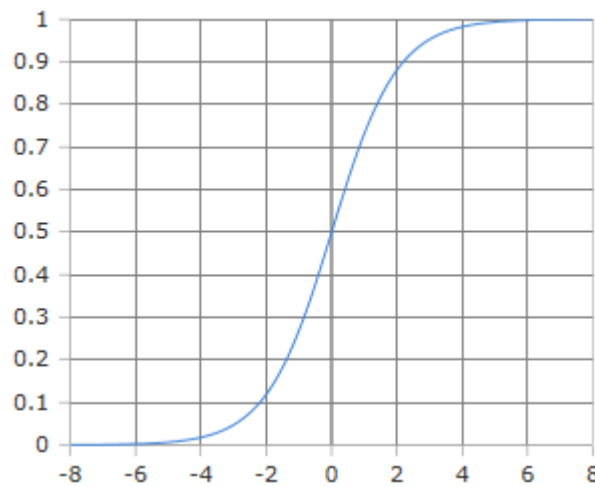
The Sigmoid AF is also referred to as the logistic function or squashing function in some studies. Its research resulted in three sigmoid AF variants for use in DL applications. The Sigmoid is a nonlinear adaptive function that is frequently used in feedforward neural networks. It is a bounded differentiable real function with positive derivatives in all directions and a certain amount of smoothness, defined for real input values. The equation for the Sigmoid function is[61]:

$$\text{sigmoid}(x) = f(x) = \left( \frac{1}{1+e^{-x}} \right) \dots\dots\dots (2.25)$$

where  $x$  is the input. With a range between 0 and 1 as shown in Figure (2.9), the sigmoid function can be used to predict posterior probabilities [62].

The sigmoid function is typically used as the output prediction in DL architectures and is found in the output layers. The concept has proved to be useful in a variety of classification tasks, including binary classification, modeling logistic regression tasks, and other neural network problems [46].

The main advantages of sigmoid functions are that they are simple to understand and that they are commonly used in shallow networks. Neural networks with random starting weights should avoid the Sigmoid AF [62].



**Figure (2.9):** Sigmoid Function [62].

### B- Hyperbolic Tangent Function (Tanh)

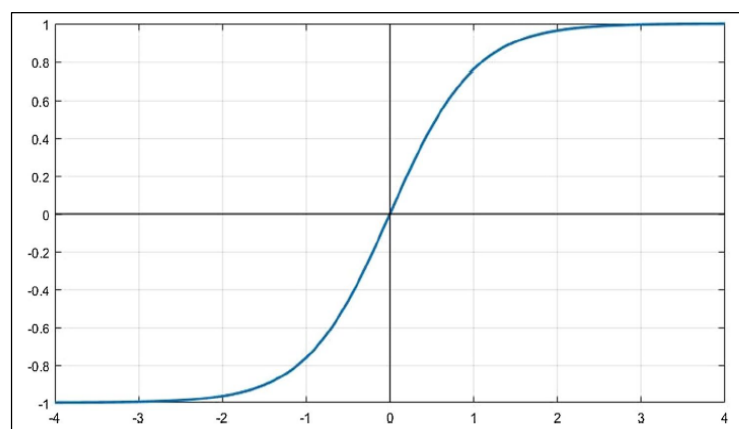
Another type of AF used in DL is the hyperbolic tangent function, DL application variations of which are used Tanh is a zero-centered hyperbolic tangent function, whose range is from -1 to 1. as shown in Figure (2.10). The tanh function's output is given by the equation:

$$f(x) = \left( \frac{e^x - e^{-x}}{e^x + e^{-x}} \right) \dots\dots\dots (2.26)$$

In comparison to the sigmoid function, the tanh function became the preferred function because it provides better training performance for multi-layer NN. The tanh function, on the other hand, was unable to solve the sigmoid functions' vanishing gradient problem. The function's main benefit is that it produces zero-centered output, which aids the back-propagation process[62].

The tanh function has the property of only being able to achieve when  $x$  is set to 0, the gradient is set to 1. During computation, as a result of this, the tanh function produces some dead neurons. The activation weight, which is rarely used as a result of zero gradient, is called a dead neuron. The tanh function's limitation prompted more AF research to find a solution, resulting in the rectified linear unit (ReLU) AF[62].

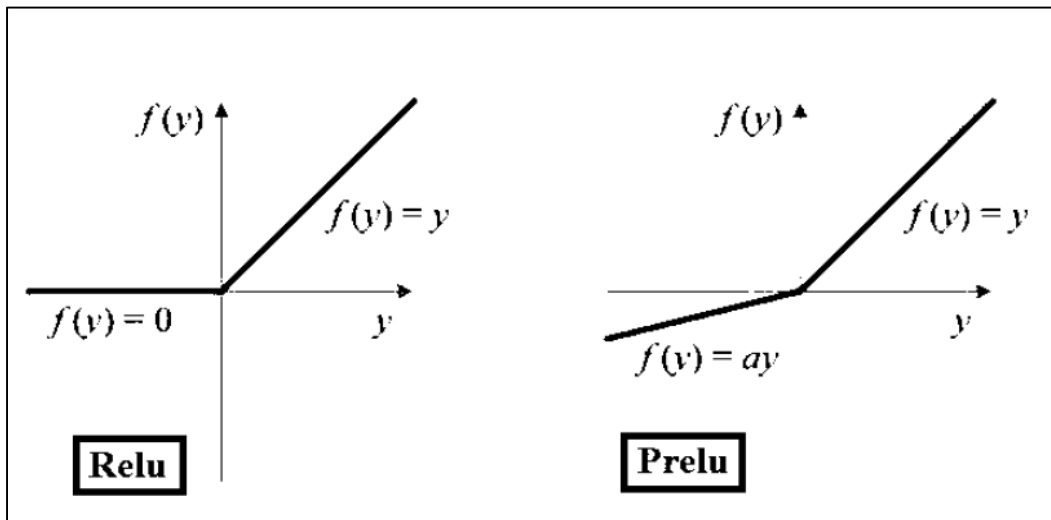
The tanh functions have primarily applied to natural language processing and speech recognition tasks using recurrent neural networks (RNN) [62].



**Figure (2.10):** Hyperbolic Tangent Function [62].

### C- Rectified Linear Units (ReLU):

Nair and Hinton proposed the ReLU activation function in 2010, and It has the most widely used AF for DL applications, with cutting-edge results. The ReLU is the most popular and successful fast learning AF. In DL, it outperforms the Sigmoid and Tanh AF in terms of performance and generalization. Figure (2.11) shows the two AF ReLU and Parametric ReLU(PReLU), which preserves the features of linear models, making them easy to optimize by means of gradient-descent methods [64].



**Figure (2.11):** types of ReLU Transformation [64].

Each input element is subjected to a threshold operation, and values less than zero are set to zero, giving the following equation for the ReLU:

$$\mathbf{ReLU}(X) = f(x) = \mathbf{max}(0, x) = f(x) = \begin{cases} x_i & x_i \geq 0 \\ 0 & x_i < 0 \end{cases} \quad \dots\dots\dots (2.27)$$

This function corrects inputs that are less than zero by forcing them to zero and thus eliminating the vanishing gradient problem that plagued previous types of AFs. With typical applications in object classification and speech recognition, the ReLU function has been combined with another AF in the network's output layers to form the hidden units of DNN[65].

The primary advantage of using rectified linear units in computation is that they ensure faster computation by omitting exponentials and divisions, resulting in an overall increase in computation speed. Another feature of the ReLU is that it squishes the values between zero and maximum, resulting in sparsity in the hidden units. However, when compared to the sigmoid function, the ReLU has the disadvantage of easily overfitting, despite the fact that the dropout technique has been used to reduce the effect of overfitting in ReLUs and rectified networks have improved DNN performance[62].



However, Nair and Hinton (2010) asserted that the ReLU has proven itself in multiple DL architectures due to its simplicity and reliability, including restricted Boltzmann machines and CNN architectures[62].

When compared to sigmoid and tanh, ReLU is significantly more reliable and significantly speeds up convergence by six times, but it is far more delicate in real-world applications. While this disadvantage cannot be completely eliminated, it can be countered by adjusting the learning rate to the desired level. [62].

#### D - Softmax Function

In neural computing, there are several AFs including the Softmax function. A big part of computer programming involves dealing with probability distributions. Softmax, a function that can be applied to multiple classes to generate an output of a range of probabilities between zero and one, with the total probability sum equal to 1. A variant of the Softmax function is implemented by way of the equation below:

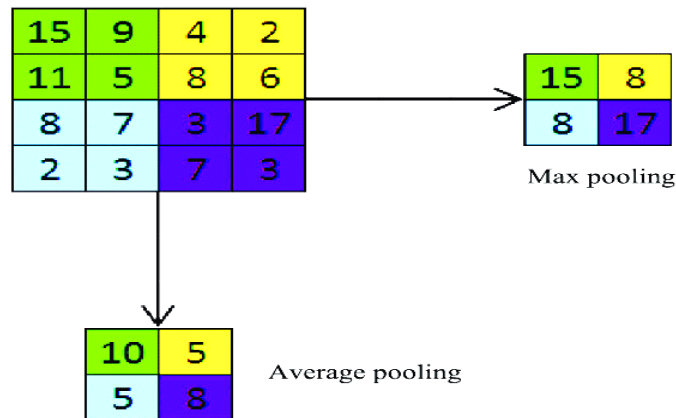
$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}} \quad \dots\dots\dots (2.28)$$

where  $X_i$  is the input layer value, this normalization of the total amount of outputs to one can be defined as the likelihood of the input belonging to class  $i$ ,  $j=1,2,3,\dots,k$ ,  $k$  is the number of classes in the multi-class classifier. and therefore the softmax output  $f(x_i)$ [62].

Multi-class models calculate class probabilities using the Softmax function, with the target class holding the highest likelihood. In nearly all of the output layers of DL architectures, the Softmax function is used. The primary difference between the Sigmoid and Softmax classifiers is that the Sigmoid classifier is a binary classifier, whereas the Softmax classifier is a multivariate classifier [62].

**4. Pooling layers** The max function or average function is the pooling layer, as shown in Figure (2.12), and the max pooling function is the most common function utilized in this layer. In a local window, the max function computes the

high-level feature. The pooling layer was employed to lower the size of the features and the amount of time it took to compute them [66].



**Figure (2.12):** Two Classic Pooling Methods [67].

Max pooling is a common pooling approach that takes sub regions of the feature map, preserves their highest value, and discards all other values [21].

### 5. Fully Connected Layer (FC)

As with a standard multilayer neural network, CNNs are composed of one or more fully connected convolutional layers [68].

Dense or fully connected layers classify the features recovered by the convolutional layers, which are then down sampled by the pooling layers. Every node in a dense layer is connected to every other node in the previous layer [21].

The feature map matrix is represented as a vector. All of the levels are completely interconnected. These feature maps are then integrated to form the final CNN model [69].

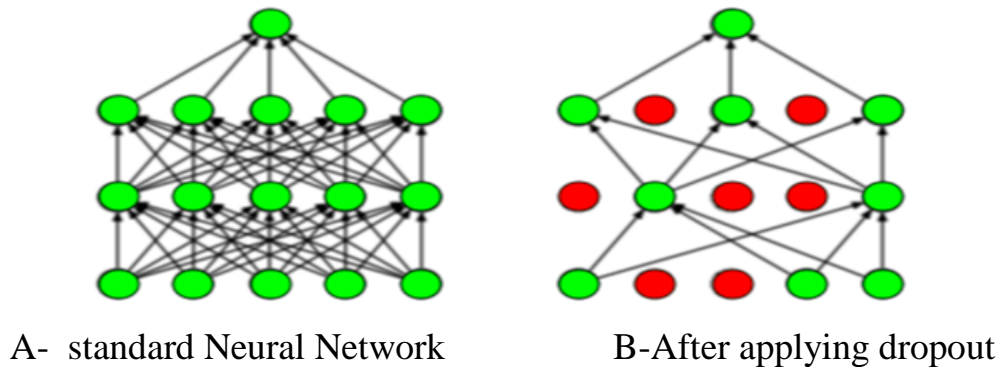
#### 2.7.3 The Network Training

Training is a weight-contact process. Most training systems start with random numbers for the matrix of weight. Then it is adjusted on a weight basis. The weight adjustment process is repeated in an acceptable way until the error limit is acceptable. The training aims, when applied to the network inputs, to produce the desired outcomes (or at least consistent) [70].

Dropout regularizes neural network training by preventing co-adaptation of model parameters. Thus reducing overfitting with limited training data [70].

Dropout can also be used as an ensemble averaging method. Instead then training multiple independent networks and weighing the results together to reduce variance in the estimation as shown in Figure (2.13), dropouts have the same tendency but with a single, larger network [71].

In addition, the fully linked layers have been subjected to a dropout learning technique with the same goal of reducing overfitting ([72], [73]).



**Figure (2.13):** Dropout Neural Network [71].

#### 2.7.4 Back Propagation Algorithm (BP)

Backpropagation is used to simulate how internal machine parameters change over time and are applied to demonstrate each layer in turn. The deep convolutional network (DCN) has made significant advancements in the processing of video, images, audio, voice, and text [9], BP methodologies help for the easy working of models [44].

The optimization algorithms are used to perform back-propagation in deep neural network learning. Taking one training sample at a time and passing it to the neural network. Furthermore, each iteration records the fault[74].

DL architectures have benefited greatly from the innovation and perfection of new optimization algorithms. In large part, NN is an optimization problem in which we use a stable training trajectory and rapid convergence to find the global optimum[74].

Optimization algorithms reduce the error function, and they employ a numeric scanning function of the model's constitutional responsibility parameters to compute objective values from the set of predictors accessed

within it. The bias values and neural network weights used in computing the output values are internal learnable parameters. Optimizers are crucial in lowering the loss incurred throughout the network training process, as well as in the neural network model during training. There are a variety of enhancers, and one of them was used in this thesis[74]:

#### 2.7.4.1 Adaptive Momentum (Adam)

Adam Optimizer is a stochastic enhancement method that requires first-order gradients and a small memory specification. From the examination of the first and second moments of the gradients, the Optimizer derives discrete flexible learning rates for various parameters [74].

The mathematical notation for Adam are as equation follows:

$$\mathbf{x}_t = \delta_1 + \mathbf{x}_{t-1} - (1 - \delta_1) * \mathbf{g}_t \quad (1)$$

$$\mathbf{y}_t = \delta_2 + \mathbf{y}_{t-1} - (1 - \delta_2) * \mathbf{g}_t^2 \quad (2)$$

$$\Delta \mathbf{w}_t = \eta \frac{\mathbf{x}_t}{\sqrt{\mathbf{y}_t + \epsilon}} * \mathbf{g}_t^2 \quad (3) \quad \dots\dots\dots(2.29)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta \mathbf{w}_t \quad (4)$$

where  $\eta$ : 'Initial learning rate',  $\mathbf{g}_t$ : 'Gradient at time t along  $\omega_j$ ',  $\mathbf{x}_t$ : Exponential average of gradient along  $\omega_j$ ,  $\mathbf{y}_t$ : Exponential average of squares of gradient along  $\omega_j$ , and  $\delta_1, \delta_2$ : Hyper parameters[75].

#### 2.7.4.2 Nesterov-accelerated Adaptive Moment (Nadam)

Nesterov and Adam's optimizer are abbreviated as Nadam. In contrast to the implementation that the company initially implemented, the Nesterov component is a more efficient modification. The model training process is accelerated using an exponential decay that depends on the moving average of the gradients. The Nadam optimizer converges more rapidly and is preferable for the pre-training phase when the Adam optimizer is not yet fully trained. Nadam adds Nesterov steps ahead to the gradient to bring the equations up to date by using the equations in the following:

$$\hat{g}_t \leftarrow \frac{g_t}{1 - \prod_{i=1}^t \beta_{1i}} \quad \dots\dots\dots (2.30)$$

$$m_t = \beta_1 m_{t-1} + (1 + \beta_1) g_t \quad \dots\dots\dots (2.31)$$

$$\hat{m}_t = \frac{m_t}{1 - \prod_{i=1}^{t+1} \beta_{1i}} \quad \dots\dots\dots (2.32)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad \dots\dots\dots (2.33)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad \dots\dots\dots (2.34)$$

where  $(m)$  and  $(v)$  initialized to zero and default values (taken from Keras) is  $\alpha = (0.002)$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$  [76], [77].

### 2.7.4.3 Root Mean Square Propagation (RMSProp)

Geoff Hinton proposed RMSprop, which is an adaptive learning rate approach. To address Adagrad's declining learning rates, RMSprop and Adadelta were created. RMSprop is used to update Adadelta's vector using the following equation:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \quad \dots\dots\dots (2.35)$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad \dots\dots\dots (2.36)$$

The learning rate is also divided by an exponentially decaying average of squared gradients by RMSProp. Hinton suggests  $\epsilon$  to be 0.9. Furthermore, 0.001 is a decent default number for the learning rate  $\alpha$  [77].

The error for the present state of the model must be estimated repeatedly as part of the optimization method. This necessitates the selection of an error function, sometimes referred to as a loss function, that may be used to estimate the model's loss and update the weights to lower the loss on the next assessment. There are a lot of loss functions to pick from, and it can be difficult to know which one to use, or even what a loss function is and what role it plays in neural network training [78].

The cost error or loss function is a metric for measuring CNN model error in Deep neural networks. When training neural network models, the two primary types of loss functions to use are cross-entropy and mean squared error [78].

#### ❖ The Cross-Entropy Error:

The cross-entropy error is often utilized in classification problems since misclassification penalties are significant, The input pattern to be examined is  $n$ , while the output node's index is  $J$  The training algorithm's goal is to minimize the function represented in the equation below:

$$J = \sum_{i=1}^M \{-d_i \ln(y_i) - (1 - d_i) \ln(1 - y_i)\} \quad \dots\dots\dots (2.37)$$

where  $y_i$  is the softmax layer's output value,  $d_i$  is the training data's right output value, and  $M$  is the output node number [78].

#### ❖ The Mean Squared Error (MSE):

MSE is the most commonly used loss function for regression. The loss is the mean overseen data of the squared differences between true and predicted values, the squaring is necessary to remove any negative signs. It also gives more weight to larger differences. It's called the mean squared error as you're finding the average of a set of errors calculated as equation[78]:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad \dots\dots\dots (2.38)$$

In other words, the MSE is the *mean*  $\frac{1}{n} \sum_{i=1}^n$  of the *squares of the errors*  $(y_i - \hat{y}_i)^2$ .

Cross-entropy loss is useful because it has two major advantages. The first is that depth ambiguities no longer cause mixed depth pixels to be preferred. Second, optimizing cross-entropy leads to much faster convergence than optimizing MSE, which is hampered by gradients that go to zero near the solution [78].

## 2.8 Evaluation Measures

The performance of the baseline of the ML algorithms was assessed using a variety of measurements. A confusion matrix (CM) is an ML construct that stores information about a classification system's actual and expected classifications. A CM has two dimensions: one is indexed by the object's actual class, and the other is indexed by the classifier's predicted class [61]. The basic shape of a CM for a multi-class classification problem is shown in Figure (2.14) [79], with the classes  $A_1$ ,  $A_2$ , and  $A_n$ . The number of samples categorized as class  $A_j$  but really belonging to class  $A_i$  is represented by  $N_{ij}$  in the confusion matrix[79].

		Predicted			
		$A_1$	... $A_j$ ...	$A_n$	
Actual	$A_1$	$N_{11}$	$N_{1j}$	$N_{1n}$	
	$\vdots$		$\vdots$		
	$A_i$	$N_{i1}$	... $N_{ij}$ ...	$N_{in}$	
	$\vdots$		$\vdots$		
	$A_n$	$N_{n1}$	$N_{nj}$	$N_{nn}$	

**Figure (2.14):** A confusion matrix [79].

A number of measures of classification performance can be defined based on the confusion matrix. Some common measures of performance are calculated as follows: [79]

### A. Accuracy or Classification Rate:

Accuracy refers to the relationship between the actual true classification numbers and the total number of test samples applied during training and testing, and the calculation equation as[80]:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \dots\dots\dots (2.39)$$

**B. Recall:**

The recall indicates (also called Sensitivity) the measure of the completeness of the classifiers, which is the relationship between the correct positive prediction numbers and the total positive prediction number, and the calculation equation as[81]:

$$\text{Recall} = \frac{TP}{TP+FN} \quad \dots\dots\dots (2.40)$$

**C. Specificity:**

Specificity is denoted The ratio of true negatives to total negatives in the data, and the calculation equation as:

$$\text{Specificity} = \frac{TN}{TN+FP} \quad \dots\dots\dots (2.401)$$

**D. Precision:**

Precision denotes a classifiers' exactness measure, and it is the ratio of true positives to the total predicted positives, and the calculation equation (2.26) is [81]:

$$\text{Precision} = \frac{TP}{TP+FP} \quad \dots\dots\dots (2.42)$$

**E. F1-score:**

F1 score (F-measure) is used to evaluate the detection results, the balance between precision and recall is also transmitted.

Only when the Precision and Recall numbers are both high does the F1-score become high, the calculation equation (2.27) as following [81]:

$$\text{F1 - SCORE} = \frac{2 \times (\text{Preceision} \times \text{Recall})}{\text{Preceision} + \text{Recall}} \quad \dots\dots\dots (2.43)$$

where truth positive is the TP, false positive is FP, the true negative is TN, and false negative is FN, each one has a specific meaning in the confusion matrix as it is shown below:

- True Positive (TP): is the number of class examples correctly recognized.
- True Negative (TN): is the number of examples correctly identified as not belonging to the class.



- False Positive (FP): is the examples that were either incorrectly assigned to the class or were not assigned to the class at all.
- False Negative (FN): those that were not identified as class examples, see ([80], [81]).

# Chapter Three

## PROPOSED SYSTEM DESIGN

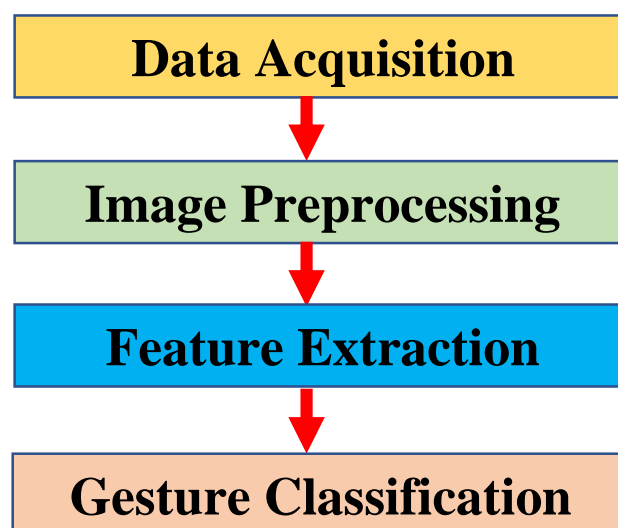
## CHAPTER THREE

### PROPOSED SYSTEM DESIGN

#### 3.1 Introduction

This work tries to contribute to providing the two proposed hand gesture recognition (HGR) systems using Machine Learning (ML) and Deep Learning (DL) algorithms will be discussed in this chapter, using the Convolution Neural Network (CNN) and Support Vector Machine (SVM). It will begin by presenting and discussing the system's general block diagram. In addition, this chapter will go through the system's architecture in detail, as well as the proposed algorithms for the various stages of the system.

Data acquisition, image preprocessing, feature extraction, and gesture recognition are the stages that the techniques used in this thesis are divided into as viewed in Figure (3.1), the progress of the algorithm, they are critically examined and their merits are evaluated at each step overall, the purpose of this thesis is to introduce readers to the field of automatic gesture and sign language recognition and to serve as a springboard for future research in this field.



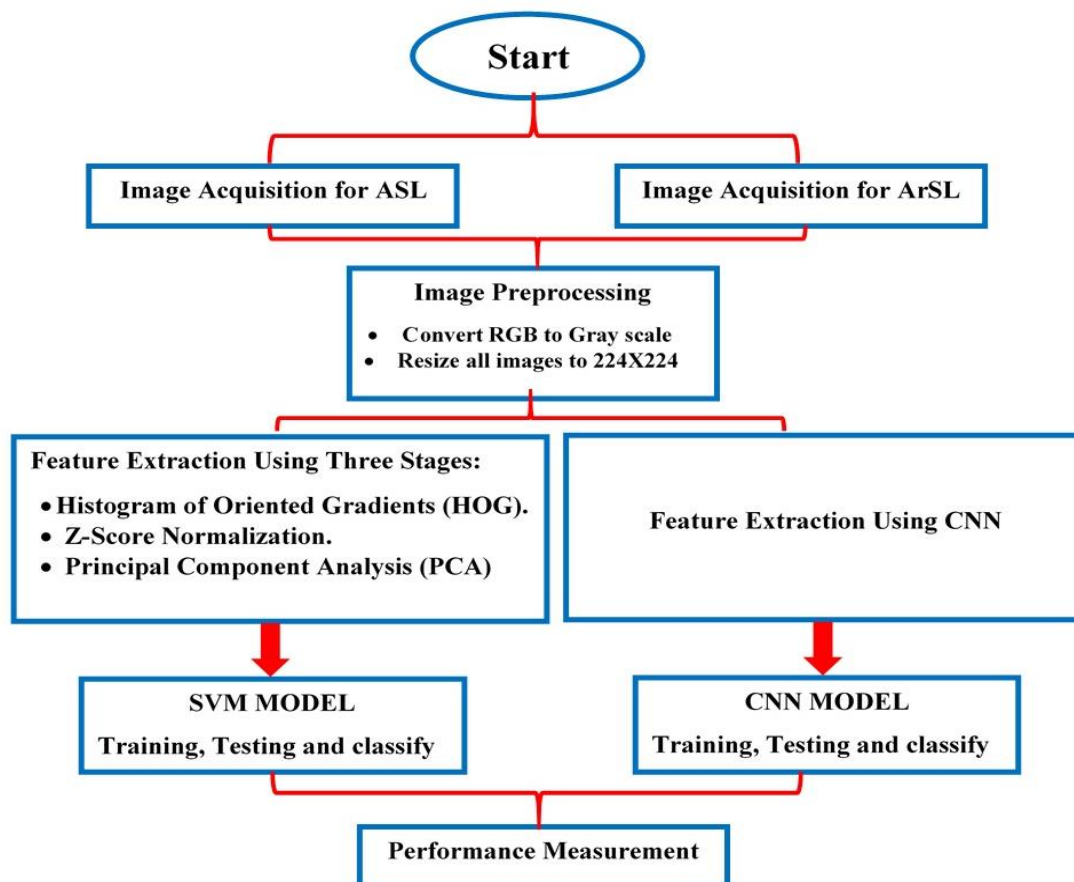
**Figure (3.1):** Stages of Hand gesture recognition (HGR).

### 3.2 The Proposed System Design

The two proposed systems aim to Static HGR for American sign language (ASL) and Arabic sign language (ArSL), Two different datasets are used to evaluate each proposed system.

In human-computer interaction, the two offered systems play a major role due to the increasing numbers of deaf and hard-of-hearing people in the community.

According to Figure (3.2), the system will be divided into two parts depending on the classification method. Each proposed system used a different dataset. Through this system and at the end, hand gestures for ASL and ArSL will be recognized and classified by applying the stages of the proposed methodology of the system, the general block diagram of the system for Static HGR and it will be explained later.



**Figure (3.2):** Block Diagram of General Proposed Systems

Figure (3.2) illustrates the major stages of a general diagram that will be applied to the two used proposed systems. In this general diagram, the stage (image acquisition) and (image pre-processing) are common stages between the two proposed systems and will be discussed in detail later. In the first proposed system, there are several stages that will have been carried out after image acquisition and image pre-processing, these include feature extraction and the recognition stage.

In the second proposed system, after the image acquisition and image pre-processing stage, the CNN structure has been built, and after that training and testing operations will use the dataset to complete the recognition process. Before explaining the two proposed, the joint stage between these two proposed systems stages will be explained.

### **3.3 Image Acquisition Stage**

The difficulty of this work is to create a dataset that requires the availability of high-quality cameras at very high prices and also the difference in the representation of the gesture from one person to the other because of the difference in skin color and lighting, these are the main reasons that make it very difficult to create a data set consisting of thousands of the different images of hand gestures. For these reasons, we resort to the famous and approved website safe, which is freely available to all researchers, and which contains thousands of images such as kaggle and mendeley.

In order to classify hand gesture images by using the proposed systems, the dataset was collected from different sources, the first data set is a collection of images of 26 alphabets from ASL and the second dataset images for the 32 ArSL.

The image for a computer is a three-dimensional matrix (width x Height channel), and values range between (0-255), The input images that used in the

proposal system for hand gesture recognition whose size has been 200 x 200 x 3 for ASL, and 64 x64 x3 for ArSL.

The input layer consists of three layers because the dataset is RGB color images are red, green, and blue in JPG file format, so each color has a specific layer.

### **3.4 Image Preprocessing Stage**

The image size is resized before it enters the ML algorithms for classification, the images are resized from 200x 200 to 224 x 224 image size for ASL and from 64x 64 to 224 x 224 image size for ArSL, the process of optimizing both input and variables aids in the acceleration of the training process. While preserving the integrity of the data without compromising the original image data, image conversion must be performed as the Algorithm (3.1), convert all of the image values in our system to grayscale by dividing all of the RGB values by 255.

**Algorithm (3.1):** Conversion RGB image to gray-scale

**Input: Color Image (RGB image)**

**Output: Gray-Scale Image**

**Steps:**

**Begin**

**Step (2): Read hand gesture image**

**Step (3):** Rows = image. Height

**Step (4):** Columns = image. Width

**Step (5):** Loop for i=0 to Rows

**Step (6):** Loop for j=0 to Columns

**Step (7):** Find pixel =Image (i , j)

**Step (8):** Compute the GP by using the equation (2.1)

**Step (9):** End for j

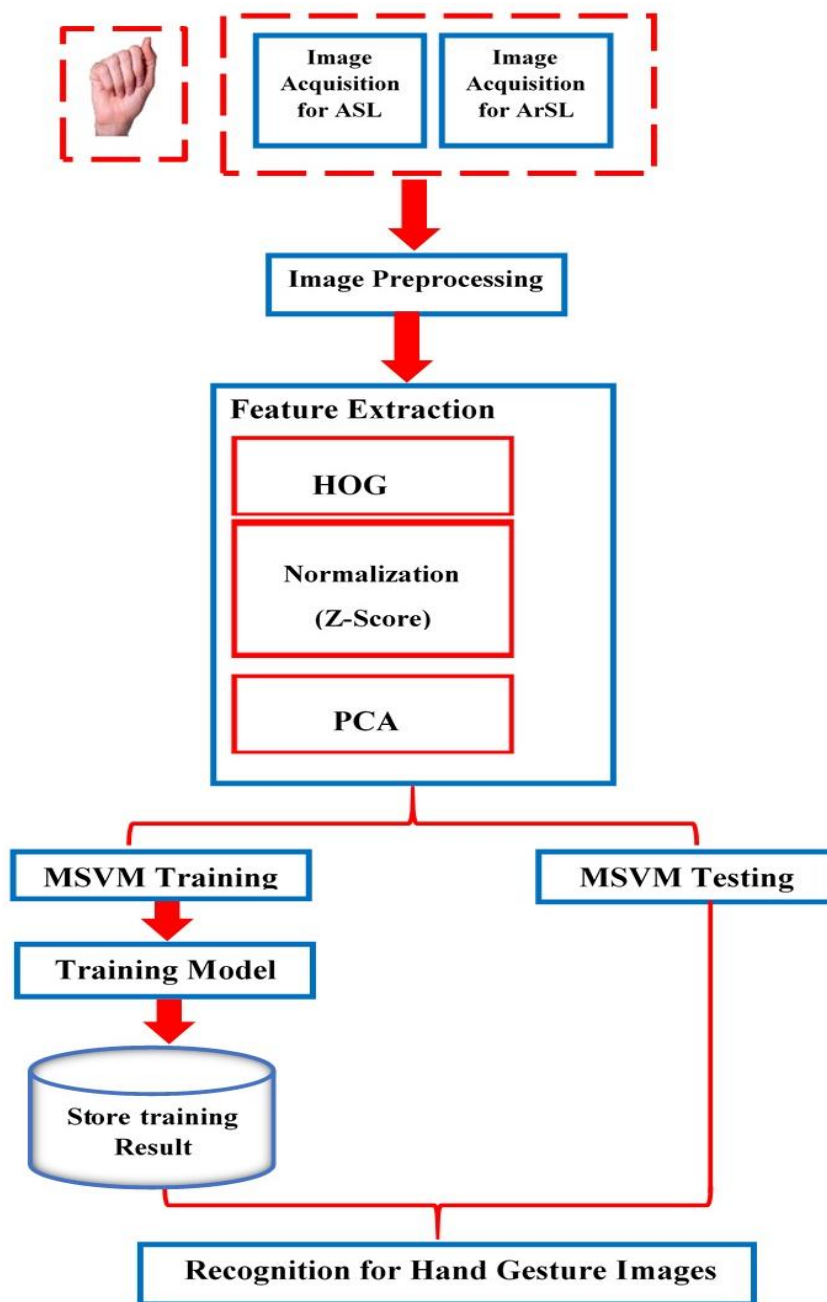
**Step (10):** End for i

**Step (11):** Return Gray Scale image (GI)

**End Algorithm**

### 3.5 The First Proposed system: (Multi-Class Support Vector Machine Algorithm (MSVM))

MSVM is a ML algorithm that is considered to be one of the most popular. depending on chapter two, the definition, details, and equations are presented (2.6.1.2). This section employs MSVM for classification after all stages of the system have been completed; these stages are depicted in Figure (3.3) as a block diagram.



**Figure (3.3):** The block diagram for the first proposed system.

**3.5.1 Feature Extraction**

The hand gesture images use to have a uniform background and are lighter in color than the hand gesture image, which made it easier to work with the hand and separate it from the background.

After converting RGB images to grayscale, work begins to determine the region of interest (ROI) in the hand gesture image using Histogram of Oriented Gradients (HOG) analysis is detected, the proposed system applies HOG algorithm on Dark gradient areas to extract HOG features as illustrated in Algorithm (3.2). The input of HOG algorithm is hand gesture image after applied to it a preprocessing operation.

The actual process of determines ROI involves separating the hand gesture from its background image. The complexity of this step would depend on the type of background that we are dealing with.

ROI is defined as the border region of the fingers, which is highlighted in white while the rest of the image is highlighted in black as shown in the next chapter.

The set of features extracted from the ROI of the hand gesture image using HOG algorithm are normalized using the Z-Score technique and then saved in the dataset. In addition, the proposed system calculated sigma for each feature's columns, and then saving in the dataset as shown in Figure (3.4), which representing Standard Deviation, that is a measure of the extent to which data varies from the mean, which has an important role to increase the accuracy of classification.



**Algorithm (3.2)** Feature Extraction using HOG Algorithm.**Inputs:** Data Set Image**Outputs:** Features**Begin****Step1:** For each image in data set do**Step2:** Resize Image to 224X224.**Step3:** Calculating Gradients Magnitude and direction (x and y)    **A-**Calculating Magnitude

For each Pixel in image do

Get neighbor pixel call [left, right, top, bottom]

 $dx = \text{right} - \text{left}$  //according to equation (2.2)             $dy = \text{bottom} - \text{top}$  //according to equation (2.3)            Magnitude= $\sqrt{dx^2+dy^2}$  //according to equation (2.4)

Endfor

**B-**Calculating Orientation

For each Pixel in image do

Get dx , dy from pixel

            Ori = $\tan^{-1}(dy/dx)$  //according to equation (2.5)

Endfor

**Step4:** Create Histograms using Orientation    **A-** split image to blocks, each block size 8\*8 pixel    **B-** Use (“unsigned” gradient) when the orientation bins(bin) are evenly spaced over 0°– 180°, size of each bin is (4), therefore, number of bin is (45).    **C-** PI= 3.14

For each block from image do

For each pixel in block do

                bin =  $N * (Q_{ori} + PI) / (PI*2)$  //bin represent (x-axis)

if (bin &lt; N) then bin = bin

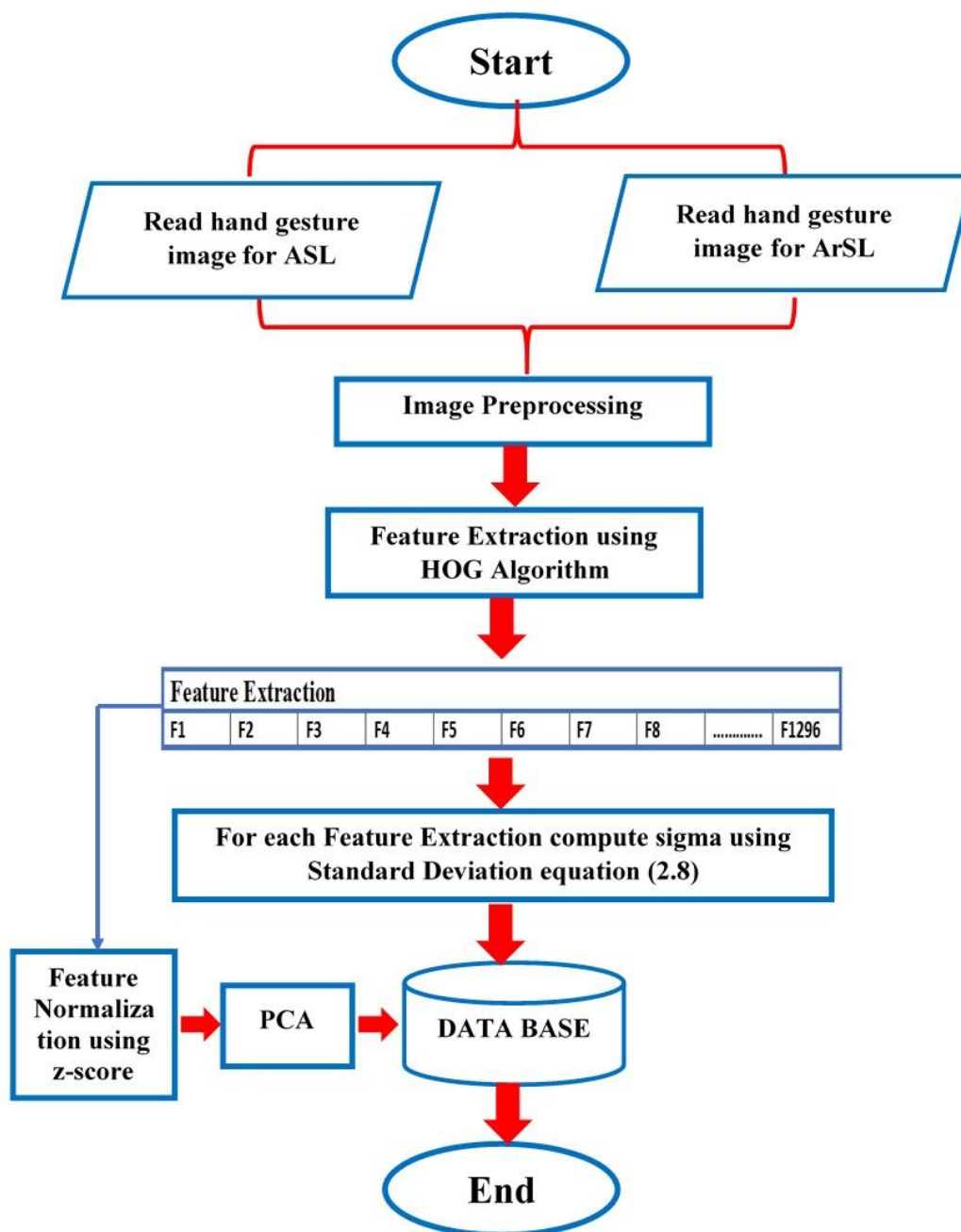
else bin = 0;

hist[bin] += 1; //hist[bin] represent (y-axis)

Endfor

Endfor

**Step5:** Normalize gradients using L2-norm Block normalization, by finding the summation of all hist[bin] then divided each value of hist[bin] on the summation according to equation (2.6) call result.**Step6:** Save result to data set features**End for****End**



**Figure (3.4):** Block Diagram of Extraction Features using HOG Algorithm.

### 3.5.2 Z- Score Normalization

The goal of Z-Score Normalization is to normalize features HOG to bring their values closer together; the results of this stage are saved in the dataset and are displayed in chapter four.

### 3.5.3 Principal Component Analysis (PCA)

The main purpose of PCA is to reduce the dimensionality of large datasets reducing a large set of variables into a smaller set that still includes most of the information from the large set.

The accuracy of a data set is reduced as the number of variables is reduced; nonetheless, the answer to dimensionality reduction is to exchange some accuracy for simplicity. Because ML algorithms can evaluate data more readily and rapidly without having to deal with superfluous factors, smaller data sets are easier to examine and display.

After stage HOG. The extracted features were then subjected to dimensionality reduction using PCA was used to extract the principal variable among the random variables illustrated in Algorithm (3.3), and as displayed in the next chapter.

#### Algorithm (3.3): PCA to Extract Feature.

**Inputs:** number of features, Y =number class , k =dimension reduction space

**Outputs:** Features that have the highest contrast

**Begin**

**Step 1:** Compute the mean vectors for the different classes from the dataset

**Step 2:** Computing the scatter matrices:

A- Computing the Covariance Matrix call A as equation (2.9)

B- Between-class scatter matrix  $S_B$

$$S_B = \sum_{x \in D} N_i (m_i - m)(m_i - m)^T$$

where  $m$  is the overall mean,  $m_i$  is sample mean, and  $N_i$  sizes of the respective classes.

**Step 3:** Compute the eigenvectors ( $e_1, e_2, \dots, e_d$ ) and corresponding eigenvalues ( $\lambda_1, \lambda_2, \dots, \lambda_d$ ) for the scatter matrices.  $Av = \lambda v$

and  $A = S_w^{-1} S_B$ ,  $V = \text{Eigenvector}$ ,  $\lambda = \text{Eigenvalue}$  as equation (2.10) and (2.11)

**Step 4:** Sort the eigenvectors by decreasing eigenvalues and choose  $k$  eigenvectors with the largest eigenvalues to form a  $d \times k$  dimensional matrix  $W$  (where every column represents an eigenvector)

**Step 5:** Calculate the eigenvalues for  $C = [\lambda_1 > \lambda_2 \dots > \lambda_N]$

**Step 6:** Use this  $d \times k$  eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication  $Y = X \times W$  where  $X$  is an  $n \times d$  dimensional matrix representing the  $n$  samples, and  $Y$  is the transformed  $n \times k$ -dimensional.

**END**

### 3.6 Recognition Stage Using MSVM Algorithm

This stage is the last stage in the first proposed system. The Recognition process is the summary of the work through which the decision is made. The MSVM algorithm is chosen, as it is one of the well-known traditional algorithms of supervised learning of machine learning algorithms. This algorithm is explained in detail with its equations in chapter two, section (2.6). The extracted features from the previous stages will be adopted at this stage. In the previous stage, 1296 important features were extracted from each image in the dataset. They were saved in a database.

In this database, the 1296 features that were extracted before the HOG method are also stored and after that used PCA to reduce these features and MSVM to classify for 26 classes ASL and 32 classes for ArSL.

**Algorithm (3.4):** Multi-Class Support Vector Machine Classifier

**Input:** Training set Features for Hand gesture images.

**Output:** Class name.

**Begin**

**Step 1:** Establish the training label for all training sets and identify 26 classes for ASL and 32 ArSL.

**Step 2:** Specify the kernel function ((Linear kernel), as calculated in the second chapter in the equation (2.19).

**Step 3:** Segregated data to 1 against all classes

**Step 4:** Passes testing set on MSVM depended on the hyperplane to identify the class name.

**Step5:** Return the class name

**End**

### 3.6.1 MSVM Training

At this stage, the proposed system has 26 classes of different types of hand gesture images for ASL, and 32 classes of different types of hand gesture images for ArSL to be recognized. In general, the SVM algorithm classifies only two classes, but here it has more than two classes. For this reason, the multiclass MSVM method type One Versus Rest (1VR) system is used, which is explained in section (2.6.1.B.1). The main idea is to express the problem oppositely: rather than writing "class A vs. class B vs. class c " the issue can be written "class A against the rest, class B against the rest," ..., n, ultimately n differential learning issues are the inverse to "class n against the rest. In the training phase, the images obtained on the MSVM algorithm are entered, in order to be classified in any class for any hand gesture. Each SVM binary classifier is trained to utilize a vector of training data, every row associates with features extracted as an investigation from a class. Following the training stage,

the multi-class MSVM model is able to decide the right class for the input features vector.

### **3.6.2 MSVM Testing**

The classifier was a trained MSVM model that produced an independent test dataset that was categorized according to the classifier's test accuracy. A specific group is used randomly for this stage, which is the testing stage, in order to estimate and evaluate the quality of the model training that has been proposed. Each row of the previously extracted feature vector was categorized and unlabeled in this stage, whereas the labeled rows are in the training stage. The classified system was developed using data from the training stage as well as the feature vector. Each feature is associated with a specific column in this sample, resulting in a matrix of samples. Because the number of columns represent the number of features, the sample size must be divided by the same batch size as the training data.

### **3.6.3 Recognition for ASL and ArSL hand gesture image in the First Proposed system (Performance Measurement)**

To perform the classification process, there are several forms available online or offline. Based on the previous steps and the training dataset, the Offline form was used to perform the recognition of static hand gesture images for ASL and ArSL to find the accuracy as shown in chapter four.

## **3.7 The Second Proposed System using CNN Algorithm**

CNN is one of the most effective methods for a higher-level representation of image data. CNN learns how to extract features from image pixel data and attempts to return inference about pixels. CNN processes the input image and classifies it into different classes.

After passing the image through a sequence of convolutional, nonlinear, pooling, and fully connected layers, the output is generated. The first layer's

output is used as the second layer's input. This happens with each new convolutional layer.

The diagram for the second proposed system to classify and recognize the hand gesture image will be displayed in Figure (3.5). Then the Algorithm (3.5) for CNN training will be shown. In this algorithm, the processes that occurred to classify and recognize the hand gesture image with preprocessing were presented starting from the first stage.

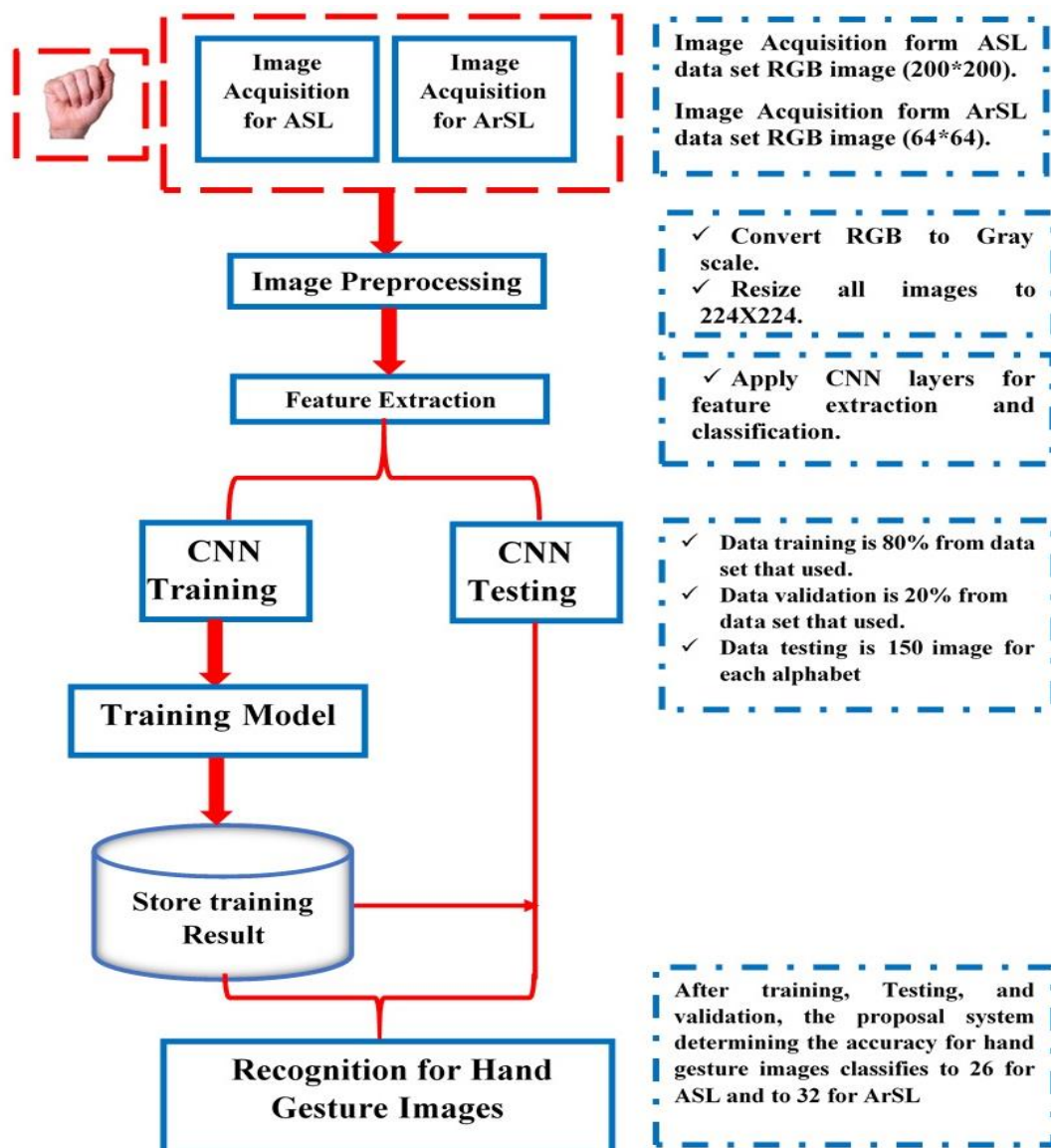


Figure (3.5): Block Diagram of Second Proposed System

**Algorithm (3.5):** CNN Training Algorithm to Classify Hand Gesture Images

**Input: Hand Gesture Images After Preprocessing**

**Output: Classification For Hand Gesture Images**

**Begin**

**Step (1):** Splitting the data into two parts, 80% for training and 20% for the testing.

**Step (3):** Implement CNN algorithm.

**Step (4):** CNN design which consisting 4 CNN with several layers, each CNN consists as:

- a. Input layer: RGB image after preprocessing.
- b. Convolution layer: Multiple filters were used with size  $3 \times 3$ .
- c. Nonlinear layer (Activation layer): Using Rectified linear units (ReLU) .....as Equation (2.2).
- d. Pooling layer: Using Max-pooling layer with size  $(2 \times 2)$ .
- e. Dropout layer to exclude 25%,50%, 25%, and 25% Respectively of neurons to reduce overfitting.
- f. Flatten layer: converts the two-dimensional matrix data to a vector, thereby allowing the final output to be processed by standard fully connected layers.
- g. two Dense or fully connected layers perform classification on the features extracted,
- h. Dropout layer.
- i. Softmax layer ....by using Equation (2.28).

**Step (5):** For each pattern in the training dataset:

- a. Input current pattern (input Image with Label).
- b. Calculate the real output of the CNN through Softmax layer
- c. Calculate the error rate by comparing the real output with the desired output.
- d. Compare the performance goal with the error rate:
  1. If the performance goal was not meet, change the connection weights by using the back-propagation learning algorithm.
  2. Else, go to next image pattern.
- e. Stop condition:
  1. If the performance goal was meeting with validation data or the maximum iteration was achieved, go to step (7).
  2. Else, repeat step (5).

**Step (6):** Return the CNN with the optimum weight.

**END.**



We also practically increased the data by training the model on more images.

After completing from image preprocessing stage, split the data set 80%, 70% and 60% for training and 20%, 30%, 40% for validation data For comparison and selection of the best results as we will see the variance of the results in Chapter (4).

### **3.7.1 Feature Extraction Stage**

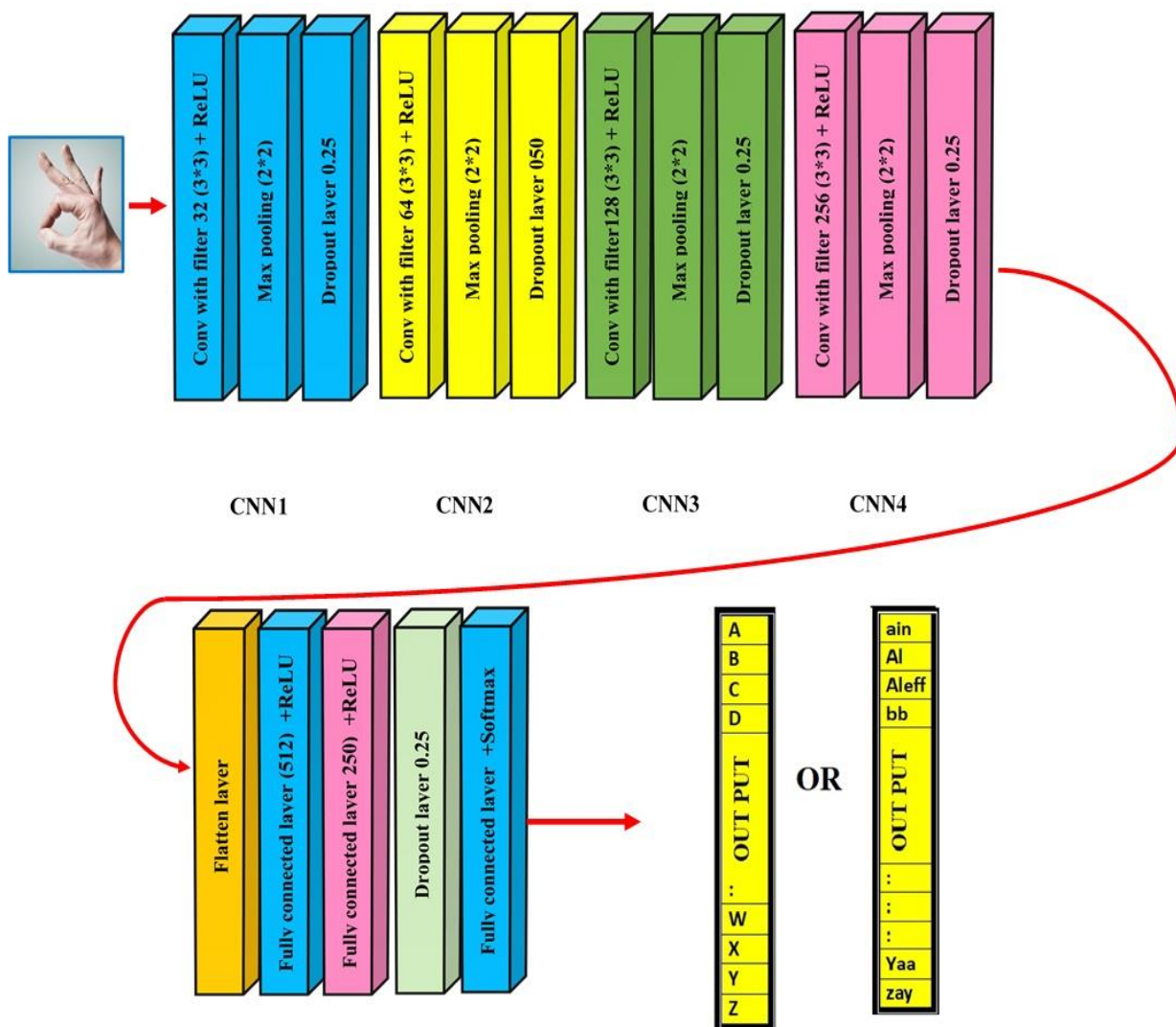
The CNN had already been used successfully for automatic feature learning. It performed admirably in image categorization, object recognition, and even recognition of human behavior, and used for extracting features in an automated way as its outcome is very satisfiable because it has several dynamic parameters to train up the machine easily.

The Convolution layer used Multiple filters with size 3X3 to repeatedly gather information about the image, however, the information obtained is only a small portion of the image area each time.

CNN's apply a variety of filters to an image's raw pixel data to extract and acquire higher-level features, which the system can ultimately employ for classification.

### **3.7.2 Design Convolution Neural network (CNN) Structure**

The structure is designing in Figure (3.6) to suit the proposed system and after changing many of the parameters and testing it, by choosing this design for the network for obtaining the best result.



**Figure (3.6):** Structure of the CNN algorithm

As shown in Figure (3.6) above, the structure of the CNN algorithm consists of several layers for each layer a specific task and a different structure, the structure is designed as follows:

1. **Input Layer:** RGB image in JPG file format after preprocessing stage.
2. **Convolution Layer:** The convolution layer is often called the extractor layer because the image features are extracted from it. First and foremost, part of the image is connected by sliding the filter to the next receiving field of the same input image via a stride and performs the same operation again with the Convolution Layer. We repeat the same process over and over until the whole

image is passed through. The output is the next layer's input; stride shows how many steps we take every step of the process. It's one by default.

The values are calculated according to the equation that was calculated according to equation (2.23), that clarifies the work of the convolution layer.

In the second proposed system, four convolution layers are used. In the first convolution layer, 32 filters were used with dimension  $3 \times 3$ , the second 64 filters were used to dimension  $3 \times 3$ , the third convolution layer, 128 filters were used dimension  $3 \times 3$ , and in the fourth convolution layer, 256 filters were used to  $3 \times 3$  dimension, as shown in Figure (3.6). Choosing the number of filters in each of the convolution layers, based on several experiments that prove the best result that obtained of these numbers that were used in each level.

### **3. Non-linear Layer (Activation Function)**

A non-linear transformation is implemented to the input by the CNN, it is also called Activation function (AF), and the node AF describes the node output provided by the input or input set. Rectified Linear Unit (ReLU), which was explained previously in section (2.7.2.3). In this function, negative values in the matrix resulting from the previous step are converted to 0 and positive values remain the same. It also showed with its calculation in the previous chapter in the equation (2.27). We are used in the proposed system after each level of the convolution layers, and with fully connected layers after flatten layer were shown in Figure (3.6).

### **4. Pooling Layer**

Another building block to the CNN is a pooling layer; the purpose of this layer used to decrease feature size and calculation time required, and the number of parameters and the calculation of the network.

In the interpretation, pooling layers are invariant as being convolution layers, since their calculations have been explained in detail in section (2.7.2.4). Average pooling and max-pooling layers are the most frequently used systems. The max pooling layer is used in the proposed system with the size  $(2 \times 2)$ .

Max pooling was chosen because it gives better results. It takes the highest value in the matrix specified for pooling and passes the process to all the values of the matrix and therefore has another matrix with fewer dimensions.

The two-dimensional matrix data is then converted to a vector by a layer called Flatten, allowing the final output to be processed by standard fully connected layers to obtain the next layers.

### **5. Fully Connected Layer**

The final layers of a CNN are frequently fully connected layers, the major difference is that the inputs would be in the form that CNN earlier stages would build. In the two neighboring layers, the neurons in a fully connected network were connected directly to each other.

The first fully connected layer with the ReLU AF contains 512 neurons. The second fully connected layer with the ReLU AF contains 250 neurons, followed by a dropout layer that excludes 25% of neurons to reduce overfitting.

### **6. Softmax Layer**

The output layer, which has a Softmax activation function and has 26 neurons for ASL and 32 neurons for ArSL, one for each hand gesture recognition class, is the final stage of the CNN structure. The mapping of the data to the final classes for hand gesture recognition is the output. The Softmax AF is according to equation (2.28).

So, what this implies is that every neuron in the fully connected layers can collect input data components over time that would help it to predict the right class value in the softmax layer afterward are shown in Figure (3.6). The objective of this layer is to summarize the weights of the features from the prior layers and show the value of per class, as was explained in section (2.7.2.). In the proposed system, 26 classes for ASL dataset and 32 classes for the ArSL dataset, will be produced from this process because according to the data used for training and determining the number per classes that have been extracted from this stage. These classes will be in the form of value for each class linked

to the fully connected layer image and Softmax will be made for them in the last step as seen in Figure (3.6).

**Algorithm (3.6):** Softmax layer function

**Input:** Fully Connected Layer Values  $Z_i$  .

**Output:** Softmax probabilities value for 32 classes for ArSL, and for 26 classes for ASL.

**Begin**

**Step (1):** Calculate the exponential for every input in fully connected layer

$$e^{z_i} \leftarrow e$$

**Step (2):** Calculate the exponential summation for two class of input fully connected layer  $\sum_{j=1}^2 e^{z_i}$

**Step (3):** Calculate the soft max function ( $y_i$ ) by using equation ( 2.28) for classes after calculating the exponential for each class in step (2), and divide each of them by the sum of the classes after calculating the exponential for them in step (3), to predicate a true class that has the highest probability.

**Step (5):** Return softmax probabilities value for the classes of hand gesture recognition ( $y_i$ )

End algorithm

### 3.7.3 CNN Training

Training a network is the process of obtaining kernels in convolution layers and weights in fully connected layers that reduce differences on a training dataset between output predictions and specified area truth labels. The training process begins with reading the model name, epoch number, and batch size from the user. The system then reads the dataset and generates the dataset augmentation. The system begins training the network using the number of epochs specified by the user earlier. The training will generate a probability value for each of the 26 ASL classification classes and 32 ArSL classification

classes, with the class with the highest probability value being the classification class predicted by the algorithm. The training results are then saved as a model file for further use. After completing the training, the system will store the model and plot the training results.

There are three parameters in this training that run continuously during the procedure: learning rate, batch size, and optimizer. The learning rate is 0.0001, and this option specifies the network layer constants for learning speed. While the batch size option determines the overall quantity of data used in a single training batch, the size of the batch that was applied in the proposed system of various sizes.

The memory capacity of the device that is used to conduct the training process is used to determine batch size. Also the optimizer is (Nadam) as explained in section (2.7.4.2).

The training stage needs three things for training, which are the training set that is obtained from the dataset, the layers in which the network was built, and the different training options that were made for training. Also, this structure has a very important function for evaluating the training process, which is the Loss function, which will also be used.

#### **a. Training Set**

The data are in two categories: training data and validation data. The data are divided using the Python function " validation split=0.20)", which splits the data randomly according to the percentages calculated by the user. To achieve a better method, it was randomly divided and not chained, and data were randomly taken from the dataset to make identification and classification later better and stronger. After checking all ratios, the data were divided in the proposed method so that the training data would be 80% and the validation data would be 20%. As will be seen in the chapter (4).

**b. Layers (designing)**

The layers are supposed to be developed during the training stage of the CNN structure, and Images will be passed over all these layers to extract features and learn from them so that the classification is done using the extracted features.

**c. Loss Function**

A loss function, also known as a cost function, can be used to determine the accuracy between the network's performance estimations and given region truth labels., which aids in the optimization of CNN parameters. The main goal here is to reduce the failure of a CNN by optimizing its parameters (weights). A CNN combines the target (actual) result and the expected value with errors to calculate the loss using the loss function. In this thesis, we use the cross-entropy error as explained in section (2.7.4.3).

**d. Training Option (training algorithm)**

In order for the training process to be completed, that need multiple options that were used in this process using a Python program by parameters that are created in the training process these options are:

**1. Nesterov-accelerated Adaptive Moment (Nadam)**

The method is utilized in the data training process, due to its good properties that fulfill the purpose of training. It is an optimization method utilized to train CNN and ML systems. Nadam is an optimization algorithm that can be used to iteratively change network weights depending on training data, and its tool helps move vectors of gradients in the true directions and thus contributes to faster convergence. It is one of the most famous optimization algorithms and it is used to train several states of the art systems. They were explained in the previous chapter in section (2.7.3.2).

**2. Max Epoch**

The “Epoch” is a metric of how many times all training vectors are once utilized to update the weights. Concurrently in one epoch in the learning

algorithm, before the weights are upgraded. The maximum number of epochs used for training is 150 epochs. In addition, tried with a different number of epochs with the different batch sizes, as shown in the next chapter.

### **3. Validation Data**

It is the details that will be used to validate the training throughout. This may be an image data store with categorical labels. A Mini-Batch data store with defined responses, or a table with a cell array X, Y. Where X is a numerical array with the input data and Y is a return array, unless image paths or images are in the first column. This option was used for validation data in the proposed system. The testing set refers to the community that was used in the subsequent testing process, while the dataset refers to a collection of datasets that have been labeled.

#### **3.7.4 CNN Testing**

The testing dataset is utilized to offer an unbiased final design fit assessment based on the training data set. The system now employs the groups that were trained in the previous phase of CNN, and the features were extracted from learning the network when the dataset was transmitted through the hand gesture image on this network. The dataset that was assigned to the research phase was also used.

The classification of hand gesture images has been completed. The training stage comes before the testing stage, which means that the network is trained when some image of a hand gesture is inserted. Since the network was previously learned and practiced, the types of hand gestures can be determined. So, the key distinction between training and testing is that test data is unlabeled, while training data is classified. This feature in Python is used in the proposed framework, `scores = model.Evaluate (X_test, y_test)` assigns each row of the dataset to one of the training classes. Both the sample and training arrays must have the same column size. Training the Group is a group element, and the individual values decide groups, and each factor specifies the group the related



training row belongs to. We have arrived at the final stage, which is the classification and recognition of static hand gesture images.

# **Chapter Four**

**EXPERIMENTAL RESULTS  
AND DISCUSSION**

## **CHAPTER FOUR**

### **EXPERIMENTAL RESULTS AND DISCUSSION**

#### **4.1 Introduction**

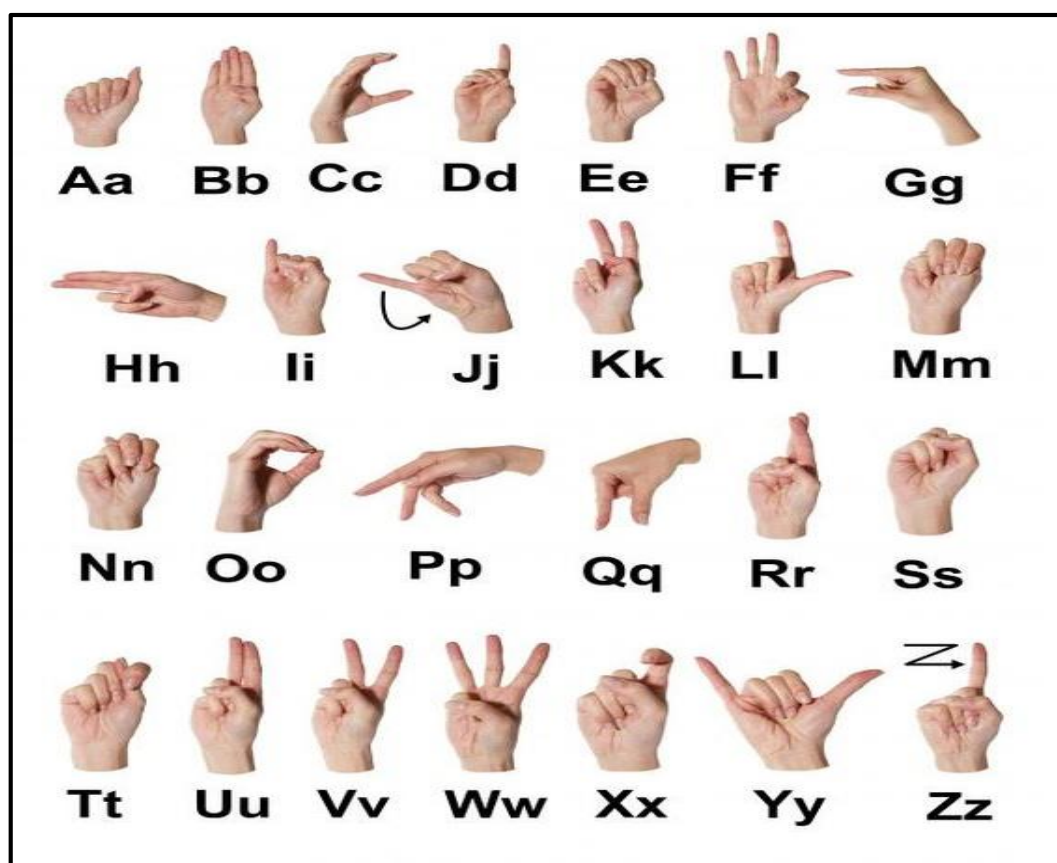
In this chapter, the implementation and experimental results of the proposed system were presented and described. This proposed system "hand gesture recognition based on machine learning techniques" is divided into two different subsystems according to the used algorithms. The first proposed system that was previously explained consists of several stages; the outcomes of these stages will be presented with the final results of classification using the multi-class support vector machine (MSVM) algorithm, the second proposed system is based on the convolution neural network (CNN) algorithm. In addition to the results of these two systems will be presented and a comparison will be made between the results of the two systems. Moreover, a comparison with related works will be presented. The same dataset was used in the two subsystems.

#### **4.2. Implementation Environment**

Hand gesture image classification system using CNN and MSVM is implemented under a specific system requirement such as the Windows-10 operating system, Hardware processor: Core i7- CPU 8550U, 200 GHz, and (8GB) RAM. Python (3.7.10 64-bit) programming language with Tensor Flow backend, CNN programs implemented on Kaggle server.

### 4.3 Dataset Acquisition

The proposed system used two data set images, the first data set is a collection of images of 26 classes from the American Sign Language (ASL) as shown in Figure (4.1) from “Kaggle is a Google subsidiary that operates as a community of data scientists and developers and is one of the best data collection sites. It also has a large community where you can discuss data, find available code, and create your own projects. It was founded in 2010 with the goal of becoming the first platform to host predictive analytics and data mining challenges and competitions”, Every downloaded image is recorded to RGB color space at 200 \* 200 sizes and saved in JPG format in file named (ASL data) and stored on the computer.



**Figure (4.1):** The Alphabets of the American Sign Language[82].

The second dataset images of the 32 Arabic sign language (ArSL) and alphabets as shown in Figure (4.2) from “Mendeley Data is a safe cloud-based repository where you can keep your data and share, view, and cite it from anywhere”. Every downloaded image is recorded to RGB color space at 64\* 64 sizes and saved in JPG format in an uncompressed file named (ArSL data) and stored in the computer.



**Figure (4.2):** The Alphabets of the Arabic Sign Language[83]

The two datasets that are used in the two proposed system have distributed as Table (4.1) shows the distribution of hand gesture images.

**Table (4.1):** The Distribution of Hand Gesture Images.

Date Set Type	Number Of Class	Number Images For Each class For Training	Number Images For Each class For Testing	Number Images For Training	Number Images For Testing
ASL	26	750	150	19500	3900
ArSL	32	750	150	24000	4800

#### 4.4 Evaluation of First Proposed System

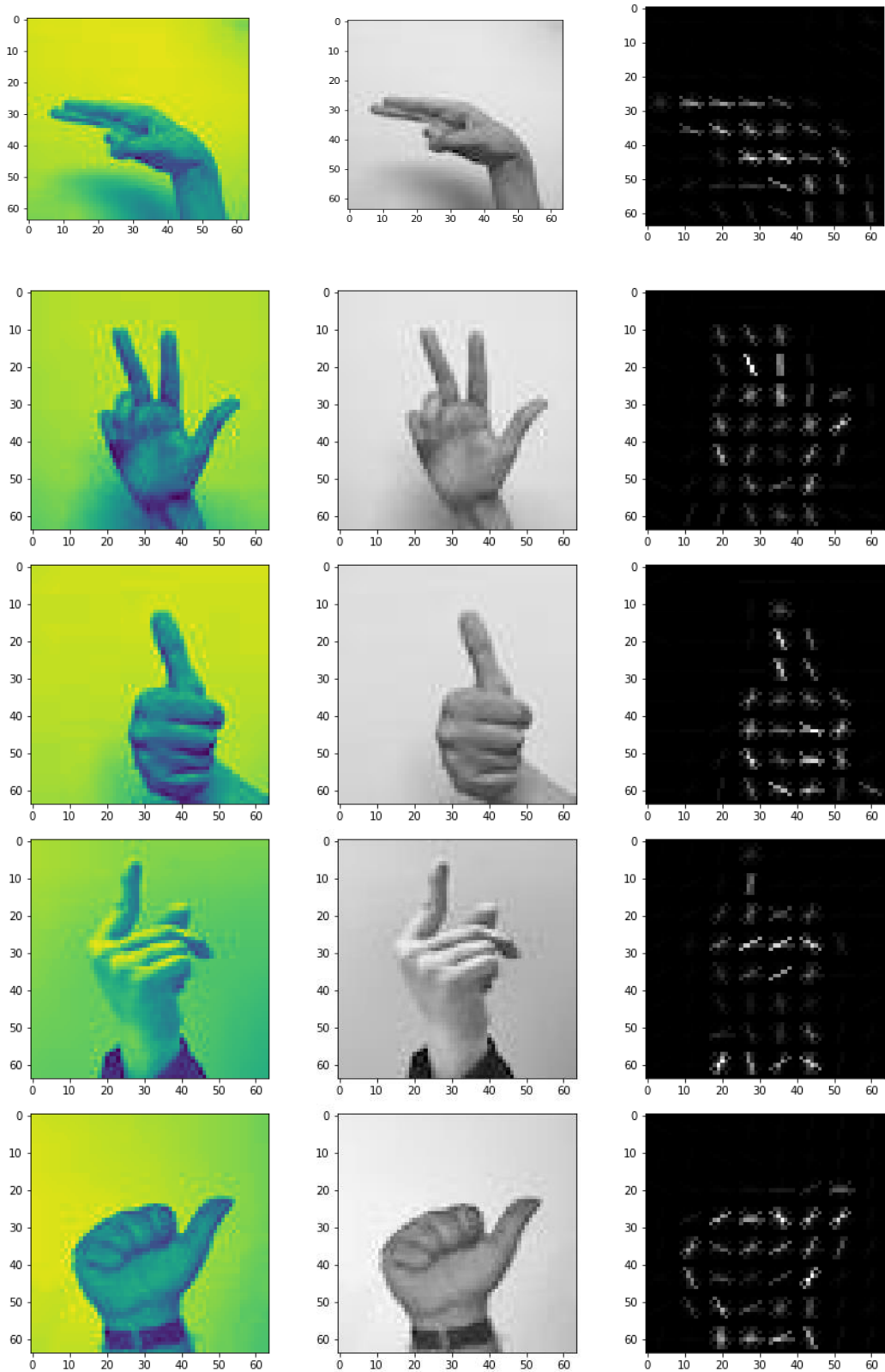
The first proposed system that was explained in the third chapter, which consists of several stages, each stage, and its results, up to the final results will be shown, in the classification stage with the MSVM algorithm.

##### 4.4.1 Result of Image Pre-Processing

At this stage, the images will be converted from RGB to grayscale and resize all images in ASL and ArSL to 224x224 size.

##### 4.4.2 Results of Implementation the Feature Extraction using Histogram of Oriented Gradients (HOG) Algorithm

After convert the RGB image to grayscale, then at this stage using the HOG to determine the border region of the fingers, which is highlighted in white while the rest of the image is highlighted in black as the Figure (4.3) that shown randomly samples from the data set , after that using HOG algorithm as illustrated in the algorithm (3.2) extract 1296 feature, compute sigma for 10 feature for one image in each class of 26 for ASL are shown in Table (4.2) and Figure (4.4). Also, compute sigma for 10 features for one image in each class of 32 for ArSL are shown in Table (4.3) and Figure (4.5), using equation (2.8).



**A: Original images      B: Gray scale      C: Afer using HOG**  
**Figure (4.3): Randomly Samples After Preprocessing**

Table (4.2) Example of HOG Features for ASL Images

	Fe1	Fe2	Fe3	Fe4	Fe5	Fe6	Fe7	Fe8	Fe9	Fe10
A	0.2356	0.0367	0.014	0.0859	0.3033	0.1831	0.0101	0.0006	0.0019	0.0063
B	0.1958	0.03	0.0117	0.0603	0.2746	0.1081	0.0132	0	0.0007	0.0172
C	0.2194	0.0472	0.014	0.0744	0.303	0.2436	0.005	0.0037	0.0029	0.0063
D	0.3055	0.0362	0.0195	0.0818	0.3366	0.2014	0.0072	0.0012	0.0011	0.0098
E	0.2197	0.035	0.0105	0.071	0.297	0.2041	0.0063	0.0013	0	0.0048
F	0.1163	0.022	0.0056	0.0448	0.2238	0.0902	0.0047	0.0006	0	0.0027
G	0.2417	0.014	0.0205	0.0074	0.2802	0.0527	0.007	0.0019	0	0.0425
H	0.144	0.0711	0.0823	0.054	0.2657	0.2092	0.0082	0.0003	0.0663	0.0023
I	0.357	0.062	0.037	0.0157	0.357	0.0607	0.0111	0.0029	0.0438	0.0056
J	0.3251	0.0541	0.0274	0.0084	0.328	0.0763	0.02	0.0093	0.1123	0.0053
K	0.1404	0.0218	0.0202	0.0177	0.2282	0.057	0.0121	0.1441	0.1635	0.0378
L	0.294	0.0691	0.0417	0.0092	0.366	0.123	0.0115	0.0026	0.2059	0.0073
M	0.2394	0.022	0.0262	0.0088	0.2785	0.07	0.0089	0.0025	0.1078	0.0066
N	0.2064	0.0196	0.0257	0.0106	0.297	0.0597	0.0066	0.0025	0.1267	0.0097
O	0.3358	0.0335	0.043	0.0103	0.3571	0.0856	0.0153	0.0027	0.2081	0.0099
P	0.2504	0.0677	0.0377	0.0101	0.3654	0.0754	0.0107	0.0028	0.24	0.0035
Q	0.2687	0.0507	0.0255	0.0689	0.3475	0.1082	0.0091	0.0031	0.14	0.0068
R	0.3456	0.058	0.0388	0.0908	0.3456	0.0548	0.0098	0.0062	0.0737	0.0082
S	0.1817	0.0558	0.0305	0.0249	0.3358	0.0954	0.0333	0.01	0.2428	0.0062
T	0.1922	0.0411	0.0234	0.0106	0.2908	0.0756	0.0157	0.0132	0.1344	0.0041
U	0.2606	0.0455	0.0293	0.0546	0.2662	0.0582	0.0105	0.007	0.0852	0.1706
V	0.3244	0.0487	0.0308	0.0589	0.3299	0.0329	0.0096	0.0021	0.0639	0.0057
W	0.2203	0.0353	0.0215	0.0306	0.3248	0.0538	0.0063	0.002	0.0452	0.005
X	0.2647	0.0387	0.0224	0.0609	0.2904	0.05	0.0099	0.0026	0.0246	0.0057
Y	0.3497	0.0583	0.035	0.1127	0.3506	0.0848	0.0077	0.0032	0.0703	0.005
Z	0.3571	0.0505	0.0306	0.0679	0.3571	0.0585	0.0107	0.0023	0.0014	0.0075

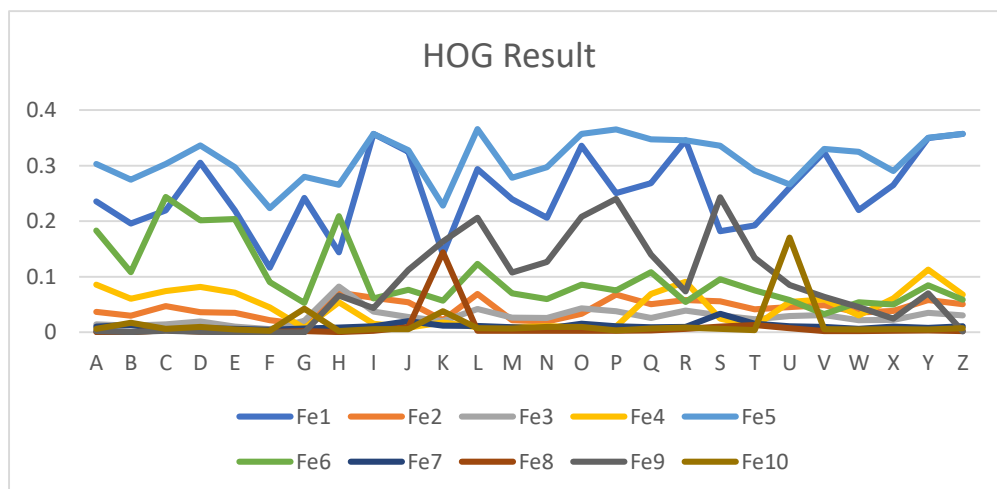
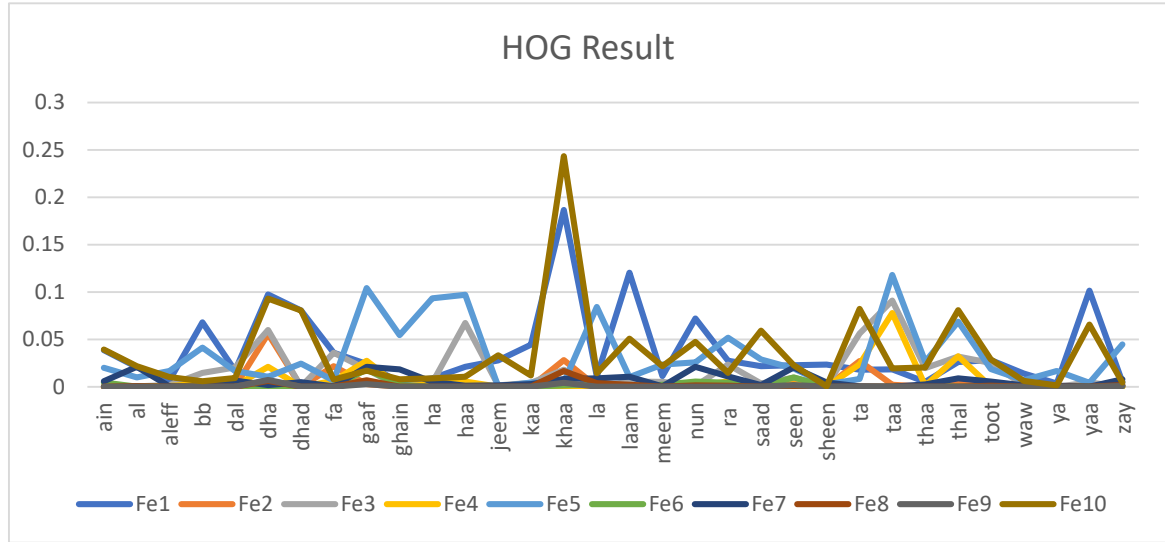


Figure (4.4): HOG Features for ASL Images.



**Table (4.3)** Example of HOG Features for ArSL Images

	Fe1	Fe2	Fe3	Fe4	Fe5	Fe6	Fe7	Fe8	Fe9	Fe10
ain	0.0386	0	0.0007	0	0.0201	0.0046	0.0058	0	0	0.0391
al	0.0207	0	0.0006	0	0.0098	0	0.021	0.0009	0	0.0216
aleff	0.007	0	0.0018	0.0007	0.0168	0	0.0004	0	0	0.0102
bb	0.068	0	0.0146	0.0033	0.0414	0	0	0	0	0.0059
dal	0.0171	0	0.0203	0.0009	0.0167	0	0.0066	0	0	0.0093
dha	0.0973	0.0558	0.0598	0.0211	0.0104	0	0.0016	0.006	0.0073	0.093
dhad	0.0805	0	0	0	0.0245	0	0.0049	0	0	0.0805
fa	0.036	0.0217	0.036	0.006	0.0065	0	0.0003	0	0	0.0076
gaaf	0.0242	0.0029	0.0184	0.0273	0.1042	0.0185	0.0212	0.0067	0.0027	0.0169
ghain	0.0045	0	0	0	0.0545	0.0017	0.0184	0	0	0.0078
ha	0.009	0	0.0042	0.0027	0.0932	0	0.0051	0	0	0.009
haa	0.021	0	0.0674	0.0054	0.0969	0	0.0011	0	0	0.0105
jeem	0.0277	0	0.0013	0	0	0	0.0013	0	0	0.0333
kaa	0.0445	0	0.0043	0	0.0046	0	0.0027	0	0	0.012
khaa	0.1866	0.0282	0.0179	0	0.0046	0.0013	0.0081	0.0167	0.0042	0.2431
la	0.0108	0	0	0	0.0843	0.0025	0.0089	0.0038	0	0.0142
laam	0.1204	0	0.0053	0	0.0103	0	0.0106	0.0021	0	0.0508
meem	0.0113	0	0.0051	0	0.0234	0.0022	0.0006	0	0	0.0218
nun	0.0719	0.0013	0.0008	0	0.0258	0.0054	0.0212	0.0013	0	0.0473
ra	0.0276	0.0043	0.0233	0.0057	0.0518	0.0041	0.011	0.0011	0	0.0145
saad	0.0215	0	0.0034	0	0.0287	0	0.0017	0	0	0.0593
seen	0.0227	0.0045	0.0094	0.0025	0.0193	0.01	0.0203	0.0007	0.0017	0.0226
sheen	0.0234	0	0.0008	0	0.0026	0	0.0045	0	0	0.0003
ta	0.0181	0.027	0.056	0.0238	0.0082	0	0.0003	0	0	0.0823
taa	0.0185	0.002	0.0908	0.0777	0.1179	0	0	0	0	0.0194
thaa	0.0051	0	0.0195	0.001	0.0265	0.001	0.0026	0	0	0.0202
thal	0.0261	0.0052	0.0318	0.0322	0.0686	0	0.009	0	0	0.0807
toot	0.028	0	0.0251	0	0.0186	0	0.0057	0.001	0	0.0289
waw	0.014	0	0.0015	0	0.0069	0	0.0015	0	0	0.0063
ya	0.0033	0	0	0	0.0167	0	0.0009	0	0	0.0013
yaa	0.1015	0.0023	0.0043	0	0.0041	0	0	0	0	0.0656
zay	0.0049	0	0	0	0.0448	0.0006	0.0077	0.0019	0.0009	0.0046



**Figure (4.5):** HOG Features for ArSL Images.

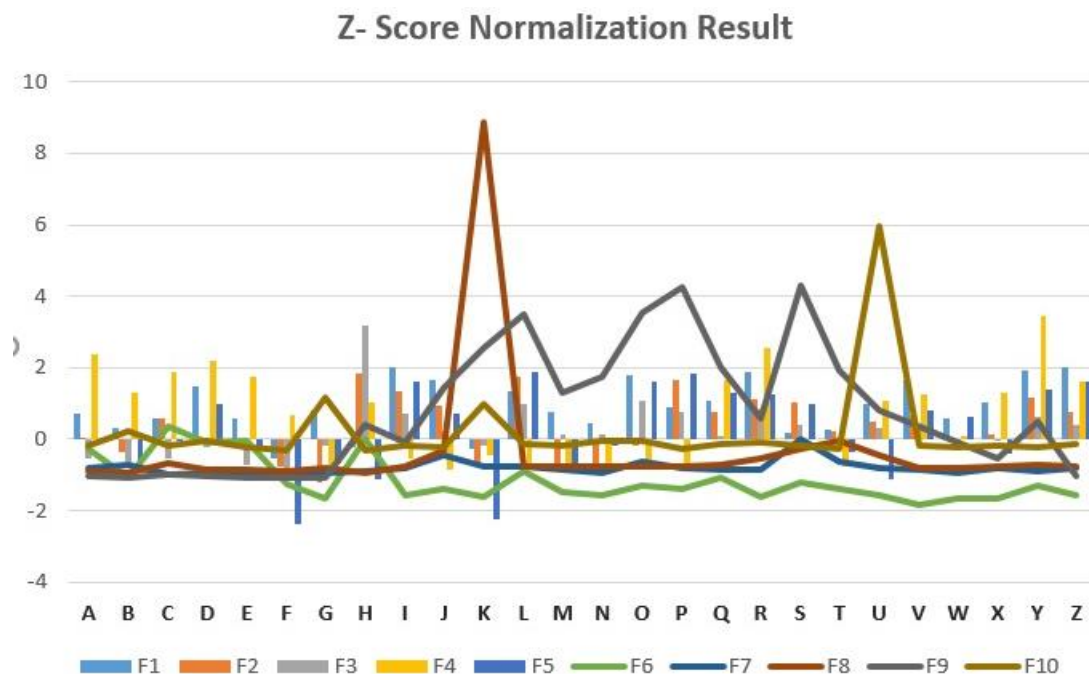
#### 4.4.3 Results of Implementation of Z- Score Normalization

As discussed in section (3.5.2), Z-score normalization aims to make the features and their sigma that are belonging to one class to be more closely related, but at the same time separate them from the other class, to avoid the overlapping of features classes and increase the accuracy(AC) of the proposed system in the classification stage. Table (4.4) and Figure (4.6) clarify the histogram for original features that extract after using Z-score normalization for ASL images of 26 classes and Table (4.5) and Figure (4.7) clarifies the histogram for original features that extract after using Z-score normalization for ArSL of 32 classes.

**Table (4.4):** Example of Histogram for Original Features to ASL.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
A	0.73	0.01	-0.5	2.36	-0	-0.3	-0.8	-0.9	-1	-0.2
B	0.31	-0.3	-0.6	1.3	-0.9	-1.1	-0.7	-0.9	-1.1	0.23
C	0.56	0.57	-0.5	1.88	-0	0.37	-1	-0.7	-1	-0.2
D	1.47	-0	-0.2	2.19	0.98	-0.1	-0.9	-0.9	-1.1	-0
E	0.56	-0.1	-0.7	1.74	-0.2	-0	-1	-0.9	-1.1	-0.2
F	-0.5	-0.8	-1	0.65	-2.4	-1.3	-1	-0.9	-1.1	-0.3
G	0.8	-1.2	-0.2	-0.9	-0.7	-1.7	-0.9	-0.8	-1.1	1.17
H	-0.2	1.83	3.2	1.04	-1.1	0.01	-0.9	-0.9	0.4	-0.3

I	2.01	1.35	0.74	-0.6	1.59	-1.6	-0.8	-0.7	-0.1	-0.2
J	1.67	0.93	0.21	-0.9	0.73	-1.4	-0.5	-0.3	1.42	-0.2
K	-0.3	-0.8	-0.2	-0.5	-2.2	-1.6	-0.8	8.86	2.55	1
L	1.35	1.73	0.99	-0.8	1.86	-0.9	-0.8	-0.8	3.49	-0.1
M	0.77	-0.8	0.14	-0.8	-0.7	-1.5	-0.9	-0.8	1.32	-0.2
N	0.43	-0.9	0.12	-0.8	-0.2	-1.6	-0.9	-0.8	1.74	-0.1
O	1.78	-0.2	1.06	-0.8	1.6	-1.3	-0.6	-0.8	3.54	-0
P	0.89	1.65	0.77	-0.8	1.84	-1.4	-0.8	-0.8	4.25	-0.3
Q	1.08	0.75	0.11	1.65	1.31	-1.1	-0.9	-0.7	2.03	-0.2
R	1.89	1.14	0.83	2.56	1.25	-1.6	-0.8	-0.5	0.56	-0.1
S	0.17	1.02	0.38	-0.2	0.96	-1.2	-0	-0.3	4.31	-0.2
T	0.28	0.24	-0	-0.8	-0.4	-1.4	-0.6	-0	1.91	-0.3
U	0.99	0.48	0.31	1.06	-1.1	-1.6	-0.8	-0.5	0.82	5.96
V	1.67	0.65	0.39	1.24	0.79	-1.9	-0.8	-0.8	0.34	-0.2
W	0.57	-0.1	-0.1	0.07	0.63	-1.6	-1	-0.8	-0.1	-0.2
X	1.04	0.12	-0.1	1.32	-0.4	-1.7	-0.8	-0.8	-0.5	-0.2
Y	1.93	1.16	0.62	3.47	1.4	-1.3	-0.9	-0.7	0.49	-0.2
Z	2.01	0.74	0.38	1.61	1.59	-1.6	-0.8	-0.8	-1	-0.1



**Figure (4.6)** Histogram for Original Features to ASL.

Table (4.5) :Example of Histogram for Original Features to ArSL.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
ain	0	-0.2	-0.5	-0.3	-0.8	-0.1	-0.4	-0.3	-0.2	0.02
al	-0.3	-0.2	-0.5	-0.3	-0.9	-0.3	-0	-0.2	-0.2	-0.3
aleff	-0.5	-0.2	-0.5	-0.3	-0.8	-0.3	-0.5	-0.3	-0.2	-0.5
bb	0.5	-0.2	-0.2	-0.1	-0.5	-0.3	-0.5	-0.3	-0.2	-0.6
dal	-0.4	-0.2	-0	-0.2	-0.8	-0.3	-0.4	-0.3	-0.2	-0.5
dha	1	3.44	0.98	1	-0.9	-0.3	-0.5	0.19	0.11	1.04
dhad	0.71	-0.2	-0.6	-0.3	-0.7	-0.3	-0.4	-0.3	-0.2	0.8
fa	-0	1.19	0.37	0.07	-0.9	-0.3	-0.5	-0.3	-0.2	-0.6
gaaf	-0.2	-0	-0.1	1.39	0.34	0.81	-0	0.25	-0.1	-0.4
ghain	-0.6	-0.2	-0.6	-0.3	-0.3	-0.2	-0.1	-0.3	-0.2	-0.6
ha	-0.5	-0.2	-0.4	-0.1	0.2	-0.3	-0.4	-0.3	-0.2	-0.5
haa	-0.3	-0.2	1.17	0.04	0.25	-0.3	-0.5	-0.3	-0.2	-0.5
jeem	-0.2	-0.2	-0.5	-0.3	-1	-0.3	-0.5	-0.3	-0.2	-0.1
kaa	0.1	-0.2	-0.4	-0.3	-1	-0.3	-0.5	-0.3	-0.2	-0.5
khaa	2.52	1.62	-0.1	-0.3	-1	-0.3	-0.3	1.07	-0	3.86
la	-0.5	-0.2	-0.6	-0.3	0.08	-0.2	-0.3	0.01	-0.2	-0.4
laam	1.39	-0.2	-0.4	-0.3	-0.9	-0.3	-0.3	-0.1	-0.2	0.24
meem	-0.5	-0.2	-0.4	-0.3	-0.7	-0.2	-0.5	-0.3	-0.2	-0.3
nun	0.57	-0.1	-0.5	-0.3	-0.7	-0	-0	-0.2	-0.2	0.18
ra	-0.2	0.05	0.04	0.05	-0.3	-0.1	-0.3	-0.2	-0.2	-0.4
saad	-0.3	-0.2	-0.5	-0.3	-0.7	-0.3	-0.5	-0.3	-0.2	0.4
seen	-0.3	0.06	-0.3	-0.1	-0.8	0.28	-0	-0.2	-0.1	-0.3
sheen	-0.3	-0.2	-0.5	-0.3	-1	-0.3	-0.4	-0.3	-0.2	-0.7
ta	-0.3	1.54	0.88	1.17	-0.9	-0.3	-0.5	-0.3	-0.2	0.84
taa	-0.3	-0.1	1.78	4.48	0.52	-0.3	-0.5	-0.3	-0.2	-0.3
thaa	-0.6	-0.2	-0.1	-0.2	-0.7	-0.3	-0.5	-0.3	-0.2	-0.3
thal	-0.2	0.1	0.26	1.69	-0.1	-0.3	-0.3	-0.3	-0.2	0.81
toot	-0.2	-0.2	0.09	-0.3	-0.8	-0.3	-0.4	-0.2	-0.2	-0.2
waw	-0.4	-0.2	-0.5	-0.3	-0.9	-0.3	-0.5	-0.3	-0.2	-0.6
ya	-0.6	-0.2	-0.6	-0.3	-0.8	-0.3	-0.5	-0.3	-0.2	-0.7
yaa	1.07	-0.1	-0.4	-0.3	-1	-0.3	-0.5	-0.3	-0.2	0.52
zay	-0.5	-0.2	-0.3	-0.2	-1	-0.3	-0.5	-0.3	-0.2	-0.6

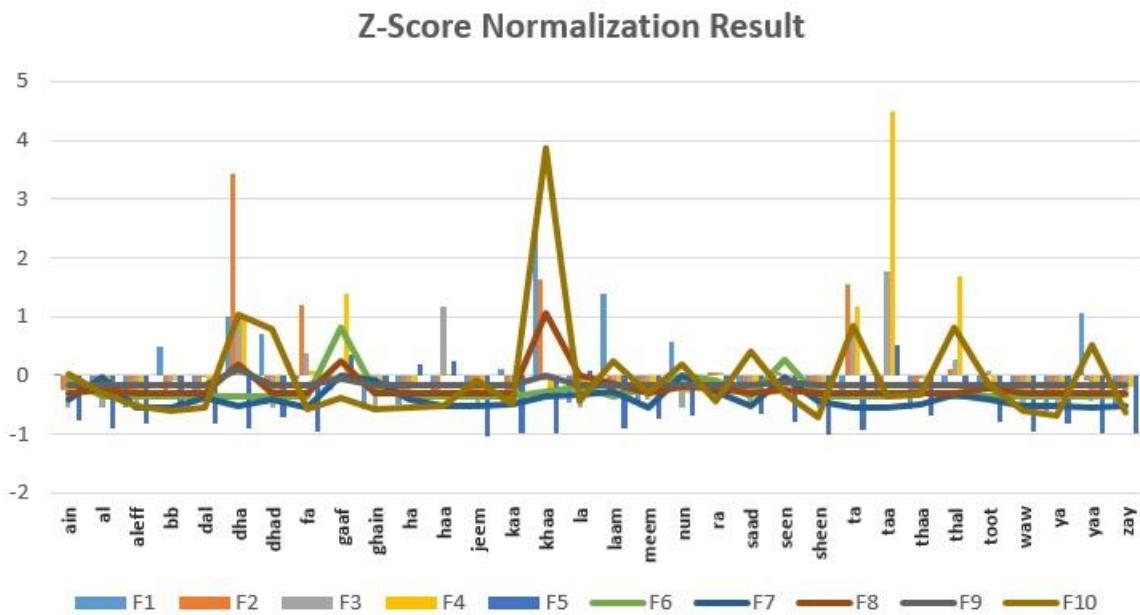


Figure (4.7): Histogram for Original Features for ArSL

#### 4.4.4 Results of Principal Component Analysis (PCA)

As discussed in section (3.5.3), Dimensionality reduction using PCA is a well-established and effective technique, but it has the limitation of requiring knowledge of the data statistics as illustrated in Algorithm (3.3). The features can be obtained in the data matrix sorted eigenvectors matrix as Table (4.6) and Figure (4.8), also Example of PCA Features for ASL Images, and Table (4.7) Example of PCA Features for ArSL Images, and Figure (4.9):

Table (4.6): Example of PCA Features for ASL Images

	Fe1	Fe2	Fe3	Fe4	Fe5	Fe6	Fe7	Fe8	Fe9	Fe10
A	0.731	0.0138	-0.52	2.36	-0.0075	-0.269	-0.821	-0.903	-1.03	-0.177
B	0.314	-0.339	-0.648	1.3	-0.862	-1.06	-0.71	-0.943	-1.06	0.228
C	0.561	0.566	-0.518	1.88	-0.0148	0.373	-1	-0.688	-1.01	-0.18
D	1.47	-0.0119	-0.218	2.19	0.984	-0.0749	-0.922	-0.86	-1.05	-0.0465
E	0.564	-0.077	-0.713	1.74	-0.194	-0.0467	-0.954	-0.854	-1.07	-0.236
F	-0.522	-0.765	-0.979	0.654	-2.37	-1.25	-1.01	-0.902	-1.07	-0.313
G	0.796	-1.19	-0.165	-0.896	-0.695	-1.65	-0.931	-0.814	-1.07	1.17
H	-0.231	1.83	3.2	1.04	-1.13	0.0077	-0.886	-0.922	0.399	-0.327
I	2.01	1.35	0.735	-0.554	1.59	-1.57	-0.787	-0.745	-0.0991	-0.203
J	1.67	0.934	0.21	-0.854	0.728	-1.4	-0.472	-0.31	1.42	-0.215
K	-0.269	-0.776	-0.184	-0.469	-2.24	-1.6	-0.751	8.86	2.55	0.998
L	1.35	1.73	0.992	-0.821	1.86	-0.906	-0.772	-0.765	3.49	-0.142

M	0.772	-0.766	0.144	-0.839	-0.746	-1.47	-0.863	-0.775	1.32	-0.168
N	0.425	-0.888	0.118	-0.761	-0.193	-1.58	-0.944	-0.772	1.74	-0.0517
O	1.78	-0.159	1.06	-0.778	1.6	-1.3	-0.638	-0.757	3.54	-0.0444
P	0.887	1.65	0.771	-0.786	1.84	-1.41	-0.799	-0.755	4.25	-0.284
Q	1.08	0.754	0.107	1.65	1.31	-1.06	-0.854	-0.729	2.03	-0.158
R	1.89	1.14	0.831	2.56	1.25	-1.63	-0.832	-0.523	0.563	-0.108
S	0.165	1.02	0.38	-0.172	0.962	-1.2	-0.002	-0.261	4.31	-0.183
T	0.276	0.243	-0.0081	-0.762	-0.38	-1.41	-0.625	-0.0422	1.91	-0.259
U	0.994	0.479	0.313	1.06	-1.11	-1.59	-0.807	-0.463	0.818	5.96
V	1.67	0.648	0.394	1.24	0.785	-1.86	-0.838	-0.803	0.344	-0.201
W	0.571	-0.0597	-0.113	0.0673	0.634	-1.64	-0.956	-0.806	-0.0697	-0.228
X	1.04	0.12	-0.0604	1.32	-0.39	-1.68	-0.826	-0.763	-0.525	-0.199
Y	1.93	1.16	0.622	3.47	1.4	-1.31	-0.904	-0.727	0.487	-0.228
Z	2.01	0.741	0.384	1.61	1.59	-1.59	-0.8	-0.783	-1.04	-0.133

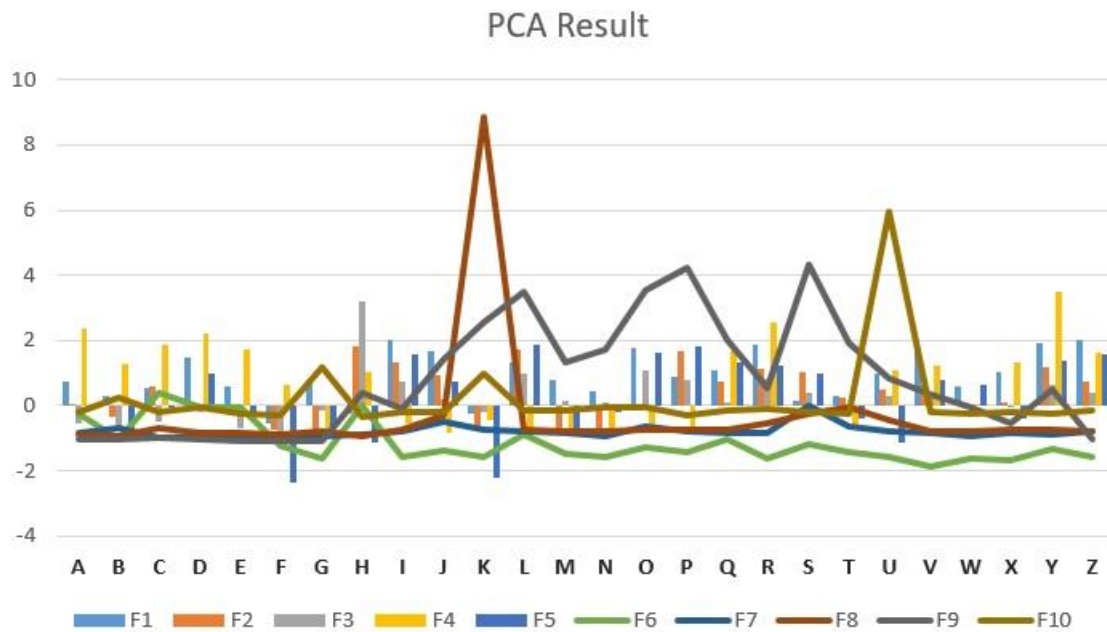
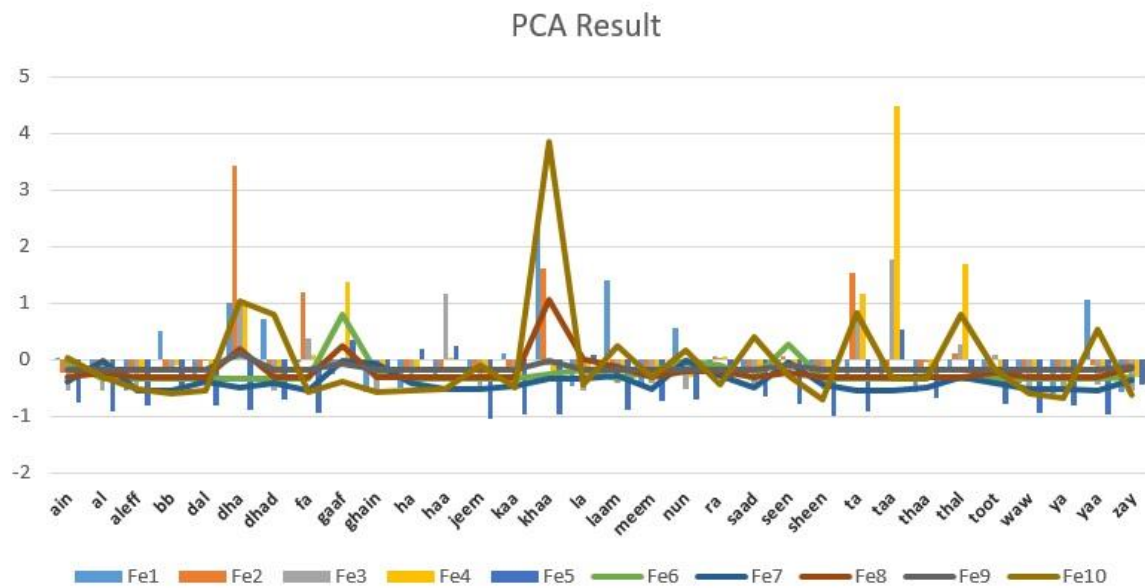


Figure (4.8): PCA Features for ASL

Table (4.7): Example of PCA Features for ArSL

	Fe1	Fe2	Fe3	Fe4	Fe5	Fe6	Fe7	Fe8	Fe9	Fe10
ain	0.002	-0.238	-0.535	-0.296	-0.768	-0.058	-0.4	-0.304	-0.166	0.0241
al	-0.303	-0.238	-0.539	-0.296	-0.904	-0.346	-0.018	-0.232	-0.166	-0.306
aleff	-0.537	-0.238	-0.507	-0.252	-0.811	-0.346	-0.535	-0.304	-0.166	-0.519
bb	0.5025	-0.238	-0.178	-0.092	-0.486	-0.346	-0.546	-0.304	-0.166	-0.6
dal	-0.365	-0.238	-0.032	-0.242	-0.813	-0.346	-0.381	-0.304	-0.166	-0.537

<b>dha</b>	1.001	3.4367	0.9798	1.0005	-0.896	-0.346	-0.505	0.1904	0.1095	1.0372
<b>dhad</b>	0.7141	-0.238	-0.553	-0.296	-0.71	-0.346	-0.422	-0.304	-0.166	0.8007
<b>fa</b>	-0.043	1.1925	0.3688	0.0745	-0.948	-0.346	-0.538	-0.304	-0.166	-0.568
<b>gaaf</b>	-0.244	-0.046	-0.081	1.3865	0.343	0.806	-0.014	0.2471	-0.064	-0.393
<b>ghain</b>	-0.58	-0.238	-0.553	-0.296	-0.313	-0.242	-0.085	-0.304	-0.166	-0.565
<b>ha</b>	-0.503	-0.238	-0.445	-0.131	0.1981	-0.346	-0.419	-0.304	-0.166	-0.543
<b>haa</b>	-0.298	-0.238	1.1745	0.0374	0.2468	-0.346	-0.518	-0.304	-0.166	-0.514
<b>jeem</b>	-0.184	-0.238	-0.52	-0.296	-1.033	-0.346	-0.514	-0.304	-0.166	-0.086
<b>kaa</b>	0.1016	-0.238	-0.443	-0.296	-0.973	-0.346	-0.479	-0.304	-0.166	-0.487
<b>khaa</b>	2.5212	1.6221	-0.095	-0.296	-0.973	-0.266	-0.342	1.0653	-0.008	3.8582
<b>la</b>	-0.471	-0.238	-0.553	-0.296	0.0802	-0.187	-0.324	0.0096	-0.166	-0.444
<b>laam</b>	1.3934	-0.238	-0.417	-0.296	-0.897	-0.346	-0.279	-0.131	-0.166	0.2428
<b>meem</b>	-0.464	-0.238	-0.422	-0.296	-0.724	-0.21	-0.532	-0.304	-0.166	-0.302
<b>nun</b>	0.5683	-0.15	-0.532	-0.296	-0.693	-0.011	-0.014	-0.194	-0.166	0.178
<b>ra</b>	-0.186	0.0472	0.0434	0.0542	-0.349	-0.087	-0.271	-0.215	-0.166	-0.439
<b>saad</b>	-0.289	-0.238	-0.467	-0.296	-0.654	-0.346	-0.504	-0.304	-0.166	0.4022
<b>seen</b>	-0.27	0.0559	-0.312	-0.14	-0.779	0.2798	-0.037	-0.243	-0.102	-0.288
<b>sheen</b>	-0.257	-0.238	-0.532	-0.296	-0.999	-0.346	-0.433	-0.304	-0.166	-0.706
<b>ta</b>	-0.347	1.5428	0.8827	1.1703	-0.926	-0.346	-0.538	-0.304	-0.166	0.8353
<b>taa</b>	-0.341	-0.108	1.776	4.4832	0.5242	-0.346	-0.546	-0.304	-0.166	-0.348
<b>thaa</b>	-0.569	-0.238	-0.052	-0.232	-0.684	-0.282	-0.481	-0.304	-0.166	-0.332
<b>thal</b>	-0.211	0.1018	0.2623	1.6873	-0.127	-0.346	-0.321	-0.304	-0.166	0.8058
<b>toot</b>	-0.18	-0.238	0.0907	-0.296	-0.787	-0.346	-0.404	-0.222	-0.166	-0.169
<b>waw</b>	-0.418	-0.238	-0.514	-0.296	-0.942	-0.346	-0.508	-0.304	-0.166	-0.593
<b>ya</b>	-0.599	-0.238	-0.553	-0.296	-0.813	-0.346	-0.523	-0.304	-0.166	-0.687
<b>yaa</b>	1.072	-0.087	-0.442	-0.296	-0.979	-0.346	-0.546	-0.304	-0.166	0.5214
<b>zay</b>	-0.573	-0.238	-0.553	-0.296	-0.442	-0.306	-0.353	-0.146	-0.131	-0.625



**Figure (4.9):** PCA Features for ArSL.

#### 4.4.5 Result of Training and Testing Using MSVM Algorithm

After the process of extracting features, the system reached the classification stage. In this stage, the static hand gesture images are classified according to the features that were extracted, and the MSVM algorithm was previously explained in the Algorithm (3.4). In order to complete the classification process, the system needs two stages that have been explained previously, which are the training stage and the testing stage. The results of these two stages will be displayed in order to them recognition and classify the hand gesture images. The training results will be displayed first and then the test results.

##### 4.4.5.1 Result of MSVM Training

This operation was explained previously in section (3.6.1). Training process for images is done in MSVM method after performing the operations and to recognize static hand gesture images for ASL and ArSL. The average



AC of all classes in the training stage for ASL is equal to (95.58%), and for ArSL is equal to (96.16%).

It is important to note that the percentage used in our thesis was not chosen at random; instead, different data ratios were tested, with the best result being 80% to training and 20% of the test data set, which is especially important for the testing process that relies on it in the classification process. Comparing accuracy with data ratios for ASL is shown in Table (4.8), and comparing accuracy with data ratios for ArSL is shown in Table (4.9).

The AC and AC of each class in training were calculated according to equation (2.38).

**Table (4.8):** Comparing accuracy with data ratios for ASL

Dataset (%) (training: testing)	The AC for training	The AC for testing	The Total time
80:20	95.58%	96%	2H
70:30	95.23%	95%	2:30H
60:40	94.47%	94%	3H

**Table (4.9):** Comparing accuracy with data ratios for ArSL

Dataset (%) (training: testing)	The AC for training	The AC for testing	The Total time
80:20	96.16%	96%	3:30H
70:30	96.125%	96%	4H
60:40	96%	96%	4:30H

#### 4.4.5.2 Result of MSVM Testing

This stage is an examination of the system by testing it with the remainder of the data that is not labeled in order to classify the static hand gesture images as explained in the previous chapter. At this stage also the results display as shown in Table (4.10) each class's results accuracy for ASL, and Table (4.11) each class's results accuracy for ArSL. the AC for ASL of testing is equal to (96%) and the AC for ArSL of testing also is equal to (96%).

**Table (4.10):** The AC of test data for all classes for ASL.

class No.	Precision	Recall	F1- Score	Support
A	0.94	0.96	0.95	140
B	0.92	0.95	0.94	142
C	0.99	1.00	1.00	139
D	0.96	0.97	0.97	162
E	0.94	0.94	0.94	136
F	0.99	1.00	0.99	148
G	0.96	0.99	0.97	157
H	0.98	0.99	0.98	149
I	0.97	0.97	0.97	144
J	0.99	0.96	0.98	142
K	0.96	0.96	0.96	157
L	0.99	1.00	1.00	164
M	0.94	0.91	0.93	150
N	0.93	0.93	0.93	144
O	0.98	0.96	0.97	165
P	1.00	1.00	1.00	158
Q	1.00	0.99	1.00	156
R	0.89	0.96	0.92	158
S	0.94	0.95	0.94	157
T	0.98	0.98	0.98	154

<b>U</b>	0.85	0.85	0.85	150
<b>V</b>	0.89	0.85	0.87	149
<b>W</b>	0.96	0.93	0.94	147
<b>X</b>	0.97	0.93	0.95	146
<b>Y</b>	0.96	0.97	0.97	136
<b>Z</b>	1.00	0.99	1.00	150
<b>Accuracy</b>	0.96			3900

**Table (4.11):** The AC of test data for all classes for ArSL.

<b>class No.</b>	<b>Precision</b>	<b>Recall</b>	<b>F1- Score</b>	<b>Support</b>
<b>ain</b>	0.94	0.99	0.96	140
<b>al</b>	0.99	0.98	0.98	143
<b>aleff</b>	0.98	0.98	0.98	143
<b>bb</b>	0.98	0.99	0.98	161
<b>dal</b>	0.94	0.98	0.96	146
<b>dha</b>	0.91	0.95	0.93	156
<b>dhad</b>	0.98	0.99	0.98	160
<b>fa</b>	0.9	0.95	0.92	136
<b>gaaf</b>	0.95	0.99	0.94	150
<b>ghain</b>	0.99	0.95	0.97	157
<b>ha</b>	0.94	0.93	0.94	153
<b>haa</b>	0.97	0.96	0.96	160
<b>jeem</b>	0.97	0.94	0.95	155
<b>kaa</b>	0.95	0.9	0.93	145
<b>khaa</b>	0.97	0.94	0.95	149
<b>la</b>	0.94	0.98	0.96	154
<b>laam</b>	0.99	0.96	0.97	149
<b>meem</b>	0.97	0.99	0.98	157
<b>nun</b>	1.00	0.97	0.99	155
<b>ra</b>	0.96	0.94	0.95	164
<b>saad</b>	0.97	0.94	0.95	153

seen	0.93	0.98	0.95	134
sheen	0.99	0.98	0.98	145
ta	0.94	0.95	0.95	167
taa	0.94	0.96	0.95	140
thaa	0.92	0.92	0.92	154
thal	0.99	0.99	0.99	154
toot	0.98	0.97	0.97	140
waw	0.98	0.98	0.98	139
ya	0.99	0.99	0.99	141
yaa	0.99	0.99	0.99	156
zay	0.98	0.92	0.95	144
Accuracy	0.96			4800

A confusion matrix (CM) is a summary of prediction results on a classification problem, the number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix as shown in Figure (4.10) CM for ASL and Figure (4.11) CM for ArSL.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
A	134	3	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	3	135	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
C	0	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	1	0	0	157	1	0	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0
E	2	1	0	3	128	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	155	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	2	147	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	1	0	0	0	0	140	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	4	1	0	137	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	1	0	0	0	0	0	0	3	0	150	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	1	1	0	0	0	1	0	0	0	0	0	0	137	8	1	0	0	1	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	7	134	1	0	0	0	0	0	0	1	0	0	1	0	0
O	1	0	0	1	1	0	0	0	0	0	0	0	2	0	159	0	0	0	0	0	1	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	158	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	155	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	151	1	0	4	0	0	0	0	0	0
S	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	3	0	0	0	0	3	0	
T	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	151	0	0	0	0	0	0	0
U	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	9	2	0	128	5	1	1	1	0	
V	0	2	0	0	0	0	0	0	0	1	0	0	1	0	0	0	2	0	0	12	126	3	2	0	0	0	
W	0	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	7	136	0	0	0	0	
X	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	3	0	4	1	0	136	0	0	0	
Y	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1	132	0	0	
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	149	

Figure (4.10): CM for ASL Using MSVM

	ain	al	aleff	bb	dal	dha	dhad	Fa	gaaf	ghain	ha	haa	jeem	kaaf	khaa	la	laam	meem	nun	ra	saad	seen	sheen	ta	taa	thaa	thal	toot	waw	ya	yaa	zay
ain	138	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
al	0	140	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
aleff	0	0	140	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
bb	0	0	1	159	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
dal	1	0	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
dha	0	0	0	1	0	148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0
dhad	0	0	0	0	0	0	158	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
Fa	0	0	0	1	0	0	1	129	3	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
gaaf	0	0	0	0	0	0	3	140	0	3	0	0	0	1	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
ghain	3	0	1	0	0	1	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	
ha	3	0	0	0	0	0	0	2	1	0	143	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
haa	0	0	0	0	1	0	0	0	0	0	0	153	1	0	2	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	
jeem	0	0	0	0	2	0	1	1	0	0	0	145	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	3
kaaf	1	0	1	0	0	0	0	0	0	0	1	0	0	131	0	1	0	0	0	0	0	3	0	0	0	7	0	0	0	0	0	0
khaa	0	0	0	0	0	1	0	0	0	0	1	3	0	0	140	1	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0
la	0	0	0	0	0	0	0	0	0	0	0	0	1	0	151	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
laam	0	2	0	0	0	1	0	0	0	0	0	0	0	0	143	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0
meem	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	155	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nun	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	1	0	0	151	0	0	0	0	0	0	0	0	0	0	0	0	0
ra	0	0	0	0	3	2	0	0	0	1	0	0	1	0	0	0	0	1	0	154	0	0	1	1	0	0	0	0	0	0	0	0
saad	0	0	0	0	0	0	2	2	1	0	2	0	1	0	0	0	0	0	0	144	0	0	0	1	0	0	0	0	0	0	0	0
seen	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	131	0	0	0	0	0	0	0	0	0	0
sheen	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	142	0	0	0	0	0	0	0	1	0	0
ta	0	0	0	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	159	0	0	0	0	0	0	1	0	0
taa	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	134	3	0	0	0	0	0	0	0
thaa	0	0	0	0	0	1	0	2	1	0	0	0	0	2	0	1	0	0	0	0	0	0	0	4	142	0	1	0	0	0	0	0
thal	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0
toot	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	2	0	0	136	0	0	0	0	0
waw	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	136	0	0	0	0	0
ya	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	140	0	0
yaa	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	155	0
zay	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	4	0	0	1	1	0	1	1	0	0	0	0	133

Figure (4.11): CM for ArSL Using MSVM

#### 4.4.6 Hand Gesture Recognition in the First Proposed System

Following conducting the training and testing processes of the system, the process of static HGR is carried out. The recognition mechanism in the first proposed system using MSVM algorithm for classification. Then conducting all the previous operations that were mentioned, hand gesture is recognized and classified. However, the AC in detection is not high due to the lack of AC of the test and training well. The reasons have been explained, and for this reason, a second proposed system has been built that is better than the first system.

## 4.5 Evaluation of the Second Proposed System

The second proposed system that was explained in the third chapter, which consists of several stages, each stage and its results, up to the final results will be shown by CNN as shown in Figure (3.1) and the procedure shown in Algorithm (3.5). This section explores the performance results of CNN. Table (4.12) shows the main CNN structure that has been used in the proposed system for ASL and Table (4.13) show the main CNN structure that has been used in the proposed system for ArSL.

**Table (4.12):** Proposed System Design CNN Layers for ASL.

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_4 (MaxPooling2)	(None, 111, 111, 32)	0
dropout_5 (Dropout)	(None, 111, 111, 32)	0
conv2d_5 (Conv2D)	(None, 109, 109, 64)	18496
max_pooling2d_5 (MaxPooling2)	(None, 54, 54, 64)	0
dropout_6 (Dropout)	(None, 54, 54, 64)	0
conv2d_6 (Conv2D)	(None, 52, 52, 128)	73856
max_pooling2d_6 (MaxPooling2)	(None, 26, 26, 128)	0
dropout_7 (Dropout)	(None, 26, 26, 128)	0
conv2d_7 (Conv2D)	(None, 24, 24, 256)	295168
max_pooling2d_7 (MaxPooling2)	(None, 12, 12, 256)	0
dropout_8 (Dropout)	(None, 12, 12, 256)	0
flatten_1 (Flatten)	(None, 36864)	0
dense_3 (Dense)	(None, 512)	18874880
dense_4 (Dense)	(None, 250)	128250
dropout_9 (Dropout)	(None, 250)	0
dense_5 (Dense)	(None, 26)	6526

- The first layer is the convolution layer, it has 32 filters, each with three channels, which has a filter size of 3x3 and a stride of 1. It is meaning every 3x3 square of an input image is treated as a separate filter. There is no zero-padding in this layer, so the number of outputs equals the

number of inputs; note that the number of parameters in this layer can be calculated using equation (2.23).

- Conv1: number of input channels is 3, the number of output channels equal 32. No. of parameters=  $32*(3*(3*3) +1) =896$ .
- Conv2: number of input channels is 32, the number of output channels is 64.

No. of parameters= $64*(32*(3*3) +1) =18496$ . The rest of Convolution layers applied the same formula.

- We used the image from the first layer first and divided it into 3 x 3 squares in step 1. There are still 32 filters as before, even though the 224 x 224 matrix is now 111 x 111. Notice that there is no parameter in this layer, as mentioned in chapter three. So, these processes repeated for layer (3 to 12).
- Finally, in layer 13 we flatten the network takes inputs from the previous pooling layer  $12*12*256= 36864$ , and that need 512 nodes for the first denes layer, while in second denes need 250 nodes and after that dropout (0.5) for calculated classification by softmax function to 26 classes

**Table (4.13):** Proposed System Design CNN Layers for ArSL

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_4 (MaxPooling2)	(None, 111, 111, 32)	0
dropout_5 (Dropout)	(None, 111, 111, 32)	0
conv2d_5 (Conv2D)	(None, 109, 109, 64)	18496
max_pooling2d_5 (MaxPooling2)	(None, 54, 54, 64)	0
dropout_6 (Dropout)	(None, 54, 54, 64)	0
conv2d_6 (Conv2D)	(None, 52, 52, 128)	73856
max_pooling2d_6 (MaxPooling2)	(None, 26, 26, 128)	0
dropout_7 (Dropout)	(None, 26, 26, 128)	0
conv2d_7 (Conv2D)	(None, 24, 24, 256)	295168
max_pooling2d_7 (MaxPooling2)	(None, 12, 12, 256)	0
dropout_8 (Dropout)	(None, 12, 12, 256)	0
flatten_1 (Flatten)	(None, 36864)	0
dense_3 (Dense)	(None, 512)	18874880
dense_4 (Dense)	(None, 250)	128250
dropout_9 (Dropout)	(None, 250)	0
dense_5 (Dense)	(None, 32)	8032

The explanation of Proposed System Design CNN Layers for ArSL is the same as the explanation of Proposed System Design CNN Layers for ASL above except calculated classification by soft max function to 32 classes.

#### 4.5.1 Result of the Second Proposed System (Using CNN)

The experiment was conducted by setting a different number of training epoch to get the most accurate result. As in Table (4.14) for ASL and Table (4.15) for ArSL, it can be seen that from epoch 1 to 10 for example it shows that the AC of selection and verification increased while the loss and verification loss decreased.



**Table (4.14):** The AC and loss for each training in 10-Epoch for ASL

Epoch	Time	Loss	Training AC	Validation Loss	Validation AC
1	20s 655ms/step	3.2609	0.0436	3.2580	0.0282
2	14s 467ms/step	3.2301	0.0589	3.2243	0.0641
3	12s 410ms/step	3.0899	0.0977	3.1264	0.0908
4	15s 488ms/step	2.8174	0.1647	2.8856	0.1733
5	17s 567ms/step	2.3330	0.2848	2.4617	0.2618
6	11s 379ms/step	1.9682	0.3730	2.1750	0.3285
7	18s 592ms/step	1.5320	0.4965	1.9823	0.3708
8	11s 374ms/step	1.2730	0.5718	1.5165	0.4905
9	12s 396ms/step	1.0436	0.6478	1.4587	0.4954
10	11s 376ms/step	0.8797	0.6960	1.4192	0.5203

**Table (4.15):** The AC and loss for each training in 10-Epoch for ArSL

Epoch ARSL	Time	Loss	Training AC	Validation Loss	Validation AC
1	80s 2s/step	3.4716	0.0336	3.4659	0.0312
2	41s 1s/step	3.4593	0.0392	3.4234	0.0640
3	14s 356ms/step	3.1734	0.0985	2.8529	0.1513
4	12s 300ms/step	2.5103	0.2410	2.0817	0.4046
5	15s 374ms/step	1.7688	0.4475	1.7890	0.5535
6	13s 315ms/step	1.1995	0.6177	1.1967	0.6392
7	12s 310ms/step	0.8231	0.7349	0.9526	0.6969
8	12s 312ms/step	0.6314	0.8062	0.7692	0.7435
9	13s 319ms/step	0.4871	0.8468	0.6839	0.7735
10	13s 324ms/step	0.4076	0.8732	0.6833	0.7771

The main problem that occurred during training is that it takes a long time, and also the speed and characteristics of the computer play a big role in the time spent

on network training. In Table (4.14) and Table (4.15), the (Time) column represents for training the size the data set in each epoch.

The structure that was explained in chapter three, section (3.7.2), shows the use (4CNN) layers, which leads to reduce the number of transactions and increasing the AC of the training, testing, and validation gradually to get the best results, as shown in Table (4.16) for ASL, and Table (4.17) for ArSL.

**Table (4.16):** Comparison of the layers of the CNN and the AC rate for ASL

CNN numbers	Number of Parameters	The AC rate for all classes in training	The AC rate for all classes in testing	The AC rate for all classes in validation
CNN1	15,881,208	99.10%	96.20%	64.41%
CNN2	6,577,208	99.28%	97.43%	68.33%
CNN3	2,587,832	99.02%	97.07%	69.51%
<b>CNN4</b>	<b>1,047,992</b>	<b>98.76%</b>	<b>98%</b>	<b>77.21%</b>

**Table (4.17):** Comparison of the layers of the CNN and the AC rate for ArSL

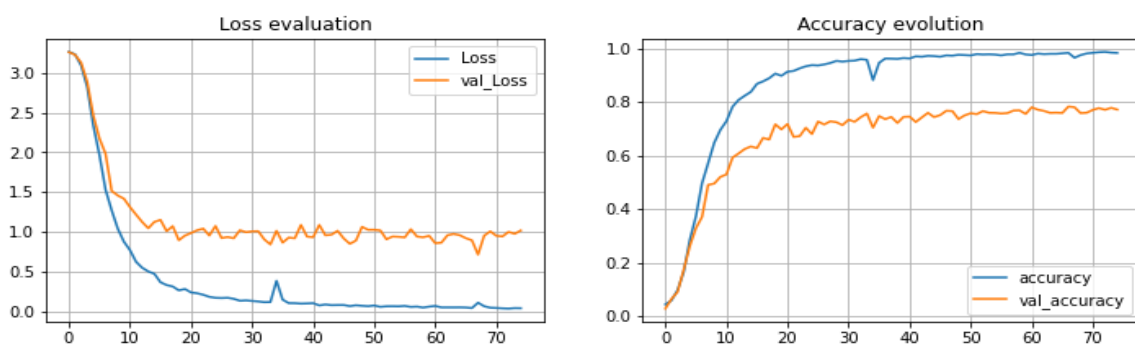
CNN layers number	Number of Parameters	The AC training	The AC for testing	The AC for validation
CNN1	15,882,714	99.89%	81,54%	71,35%
CNN2	6,578,714	99.64%	86.5%	79,60%
CNN3	2,589,338	99.68%	89.27%	84.21%
<b>CNN4</b>	<b>1,049,498</b>	<b>99.03%</b>	<b>89.38%</b>	<b>86.69%</b>

### 4.5.2 Result of the CNN Training

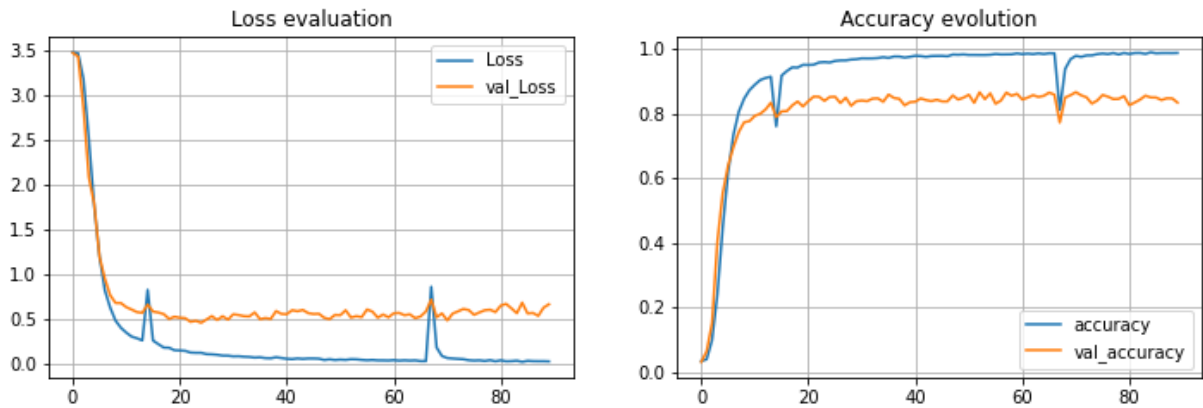
At this stage, the network training process was conducted to learn about the features of each static hand gesture image and recognize it at the recognition stage. When the network is trained and all images pass on the CNN of all layers in order to teach this network, this is the main purpose of the training process.

In the training process also, there is the process to calculate the consistency between the network's output estimates by forwarding propagation and assigned area truth labels by using the loss function that has been explained in the previous two chapters. In Figure (4.12) a chart will be displayed the loss function and the AC for ASL, and in Figure (4.13) for ArSL. The loss function which is descending to the bottom, in contrast to the AC that goes from the bottom to the up. In addition, the number of epochs was used in the training process is 75 epochs, as for the number of iterations is 30 for each epoch iterations for ASL, to extract the total number of iterations by multiplying  $75 * 30$  and the result was 2250 iterations.

After that, it was noted that one of the related work s had obtained a higher AC of the proposed system, so we increased the number of epochs to reach the closest result, and according to the Table (4.18), where 150 epochs were used epochs, as for the number of iterations is 75 for each epoch iterations for ASL, to extract the total number of iterations by multiplying  $150 * 75$  and the result was 11,250 iterations while the related work used 33,000 iterations.



**Figure (4.12):** AC and Loss Validation Change Against Training Epochs (75) for ASL.

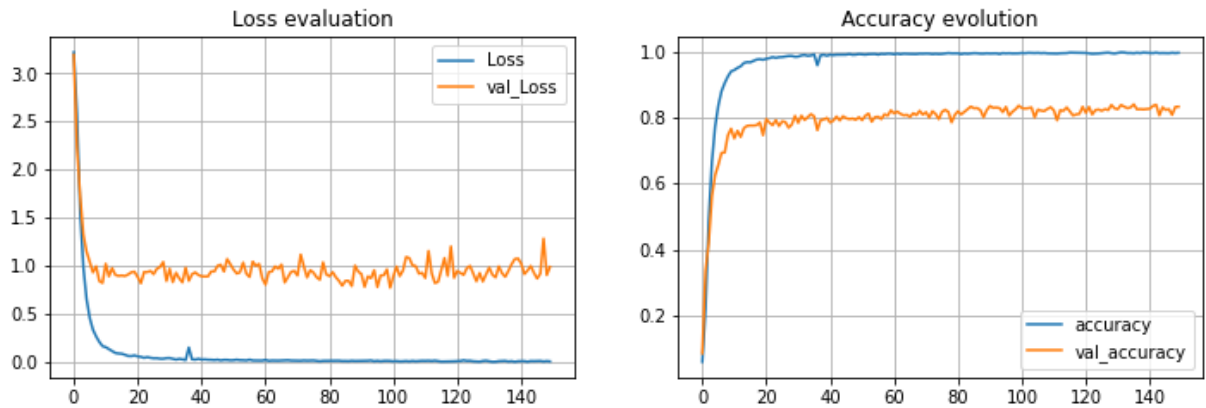


**Figure (4.13):** AC and Loss Validation Change Against Training Epochs for ArSL.

In Figure (4.14) a chart will be displayed the loss function and the AC for ASL when used epoch (150).

**Table (4.18):** Comparing the AC with number of epoch for ASL.

Epoch	Iteration for Epoch	The AC training	The AC for testing	The AC for validation	Total Run Time
75	30	98.76%	98%	78.33%	21m
90	40	99.08%	96.85%	80.03	40m
100	50	99.43%	98.179%	81.03	54m
120	60	99.52%	97.615%	81.21%	1H
<b>150</b>	<b>75</b>	<b>99.71%</b>	<b>98.717%</b>	<b>83.95%</b>	<b>2H</b>



**Figure (4.14):** AC and Loss Validation Change Against Training Epochs (150) for ASL.

In addition, the number of epochs was used in the training process is 90 epochs, as for the number of iterations is 40 for each epoch iterations for ArSL, to extract the total number of iterations by multiply  $90 * 40$  and the result was 3600 iterations.

### 4.5.3 Result of the CNN Testing

In this stage, the results of the testing were displayed. This process was explained in the previous chapter and how the convolutional neural network is tested in order to be used for the process of Recognition of static hand gesture images. In the testing stage, excellent results were obtained and presented as a confusion matrix (CM) as well as shown in Figure (4.15), which shows the results of the AC of each class of the 26 classes in the testing stage for ASL, and in the Figure (4.16), which shows the results of the AC of each class of the 32 classes in the testing stage for ArSL.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
A	148	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	3	0	134	0	0	0	0	0	0	0	0	0	0	0	11	0	0	2	0	0	0	0	0	0	0	0
E	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	1	0	0	149	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	146	4	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	7	131	0	0	0	6	0	0	0	4	0	0	2	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	149	0	0	1	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	138	0	2	0	1	3	5	0	
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	123	2	2	1	0	0	
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	149	0	0	0	0	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	145	0	0	0	
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	

Figure (4.15): CM for ASL Using CNN.

	ain	al	aleff	bb	dal	dha	dhad	Fa	gaaf	ghain	ha	haa	jeem	kaaf	khaa	la	laam	meem	nun	ra	saad	seen	sheen	ta	taa	thaa	thal	toot	waw	ya	yaa	zay		
ain	133	0	0	0	0	0	0	0	0	3	0	9	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
al	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
aleff	0	0	149	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
bb	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
dal	0	0	0	0	124	2	0	0	0	1	3	4	0	2	0	2	0	0	7	0	0	0	0	0	2	1	0	1	0	1	0	1		
dha	0	1	2	1	1	125	0	0	0	0	0	0	1	0	0	0	2	1	4	3	0	0	0	8	1	0	0	0	0	0	0	0		
dhad	2	0	0	0	0	0	136	3	2	0	0	2	1	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	
Fa	0	0	0	0	0	0	1	127	4	0	5	0	1	0	0	0	0	0	0	10	1	0	0	0	0	0	0	0	1	0	0	0	0	
gaaf	0	0	0	1	0	0	0	12	122	0	5	1	2	0	0	0	0	0	0	4	0	1	0	0	1	0	0	1	0	0	0	0	0	
ghain	11	0	1	0	0	0	0	0	0	133	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	1	0	0	
ha	0	0	0	0	0	0	0	4	4	0	130	0	0	1	0	0	0	2	0	0	4	0	2	0	0	2	0	0	1	0	0	0	0	
haa	0	0	0	0	0	0	0	1	0	0	136	3	2	5	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
jeem	0	0	0	0	2	0	0	2	0	0	3	2	129	2	0	0	0	0	1	0	2	1	0	0	0	4	1	0	0	0	0	1	0	
kaaf	0	0	0	0	0	0	0	1	0	0	1	0	0	128	0	0	0	0	0	2	5	4	0	2	7	0	0	0	0	0	0	0	0	
khaa	0	0	0	0	0	0	0	0	0	0	13	0	0	130	0	2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	3	0	
la	0	0	0	0	10	0	0	0	0	1	0	0	0	0	0	134	0	0	1	0	0	0	2	0	0	0	0	1	0	1	0	0	0	
laam	0	3	3	3	0	0	0	0	0	0	0	0	0	0	0	134	0	4	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	
meem	0	0	3	0	0	2	0	1	0	1	0	0	0	0	0	0	132	3	0	0	0	0	4	1	0	0	0	0	0	3	0	0	0	
nun	1	0	0	0	7	0	0	0	0	0	0	0	0	0	2	0	0	136	1	0	0	0	0	1	0	0	0	2	0	0	0	0	0	
ra	0	1	0	0	3	4	0	1	0	0	0	0	0	0	0	0	1	0	0	136	0	0	0	1	0	0	0	2	0	0	0	1	0	
saad	0	0	0	0	0	0	4	5	0	0	2	0	0	0	0	0	0	0	0	0	136	0	1	0	0	0	0	0	0	1	0	0	1	0
seen	0	0	0	0	0	0	1	2	2	0	1	0	0	1	0	0	0	0	0	2	134	3	0	1	2	0	0	1	0	0	0	0	0	
sheen	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	1	1	0	141	0	0	1	0	1	0	0	0	0	0	
ta	1	0	2	0	1	7	0	0	0	4	0	0	0	0	0	0	0	0	2	1	0	0	130	0	0	0	0	0	2	134	0	1	0	
taa	0	3	3	2	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	134	1	0	5	0	0	0	0	0	0	
thaa	0	0	1	1	0	0	0	0	0	0	0	0	0	5	0	1	0	0	0	0	0	2	0	0	3	134	0	3	0	0	0	0	0	
thal	0	0	0	0	10	0	2	2																										

It is important to note that the percentage used in our thesis was not chosen at random; instead, different data ratios were tested, with the best result being 80% for training and 20% for the test data set, which is especially important for the testing process that relies on it in the classification process. Comparing accuracy with data ratios for ASL is shown in Table (4.19), and Comparing accuracy with data ratios for ArSL is shown in Table (4.20).

**Table (4.19):** Comparing accuracy with data ratios for ASL

Dataset (%) (training: testing)	The AC for training	The AC for testing	The AC for validation	Total Run Time
80:20	98.76%	98%	77.21%	21m
70:30	98.35%	95.89%	68.56	26m
60:40	98.96%	90.28%	38.63%	31m

**Table (4.20):** Comparing accuracy with data ratios for ArSL

Dataset (%) (training: testing)	The AC for training	The AC for testing	The AC for validation	Total Run Time
80:20	99.03%	89.38%	86.69%	22m
70:30	99.17%	89.02%	82.61%	28m
60:40	99.24%	89.20%	82.63%	32m

#### 4.5.4 Hand Gesture Recognition in the Second Proposed System

The last stage in the second proposed system is the Recognition stage of static hand gesture images, which is the most important stage of the system. When the network has been trained and tested so that any image that is chosen will be classified according to the 26 classes for ASL and 32 classes for ArSL,

furthermore after obtaining high and excellent results in the training and testing, thus Recognition AC will be very high and the errors are almost non-existent or very rare.

#### **4.6 Comparison, Between the First Proposed System with the Second Proposed System**

The comparison between the two proposed systems is very important in order to show the strengths and weaknesses of each of them. Initially, when choosing this topic which is the recognition of static hand gesture for ASL and ArSL, at first, the MSVM algorithm was used in the first proposed system, but after getting the results of AC, notes may be can obtain of the best result with another algorithm, then begin to searching for an alternative system is better than the first proposed system in order to obtain a high recognition AC and very few errors, unlike the first proposed system. After experiments and research, the CNN algorithm was chosen in the second proposed system for classification the hand gesture images. It is worth noting that the same dataset was used, as well as the same data division for training and testing, which is 80% for training and 20% for testing for both algorithms.

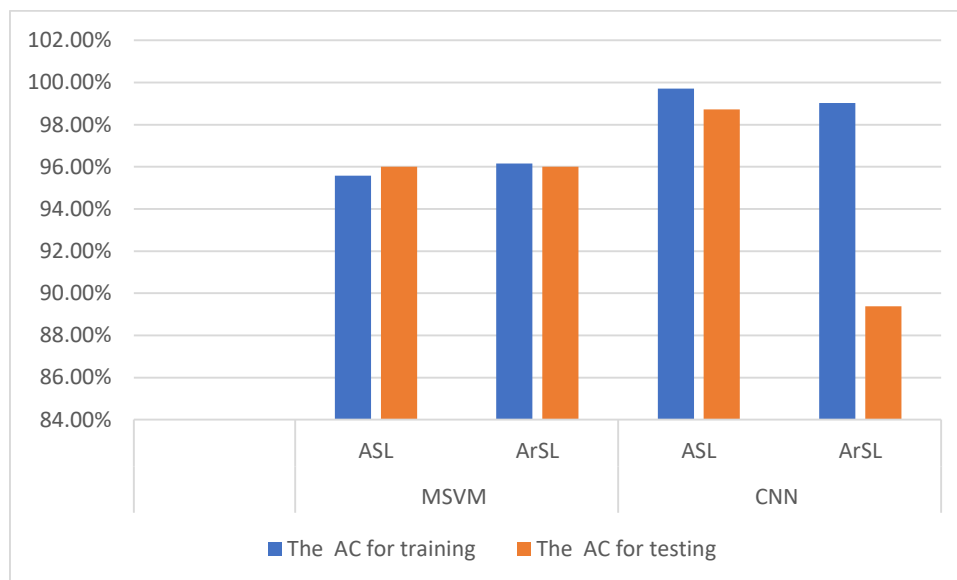
1- The high AC acquired from the second proposed system in the training and testing stages for each of the 26 ASL and 32 ArSL classes, as well as the AC rate in these two stages, as shown in the Table (4.21) and Figure (4.17) that shows the AC comparison between MSVM and CNN algorithms.

2- Because the first proposed system requires a long additional time to extract the features from the images, the time taken for the training process in the first proposed system using the MSVM method is longer than the time taken in the second proposed system using the CNN algorithm. This is also a benefit of the CNN algorithm, which extracts its features inside, reducing the need for additional time.



**Table (4.21):** The AC comparison between MSVM and CNN algorithms

DATA SET	Algorithm	The AC for training	The AC for testing	Total Run time
ASL	MSVM	95.58%	96%	2H
ArSL		96.16%	96%	3:30H
ASL	CNN	99.71%	98.717%	2H
ArSL		99.03%	89.38%	22m

**Figure (4.17):** The chart shows the AC comparison between MSVM and CNN algorithms

#### 4.7 Proposed system vs. Previous Studies

Following the achievement of high AC results on a large dataset of hand gesture images and the classification of many classes, a comparison was made between:

1- The first proposed system with our previous studies for ASL, Table (4.22) show comparing the first proposed system with previous studies.

2- The first proposed system with our previous studies for ArSL, Table (4.23) show comparing the first proposed system with previous studies.

3- The second proposed system with our previous studies for ASL, Table (4.24) show comparing the second proposed system with previous studies.

4- The second proposed system with our previous studies for ArSL, as the table (4.25) show comparing the second proposed system with previous studies.

**Table (4.22):** Comparing the first proposed system with previous studies for ASL

Researcher(s), year	Ref.No.	Classification algorithm	The size of dataset	AC
S. Nagarajan and T. S. Subashini (2013)	[5]	MSVM	720 images in 24 categories	93.75%,
A.Sharma et al. (2020)	[23]	MSVM	3000 images	85.25
<b>Our Proposal (2021)</b>		<b>MSVM</b>	<b>19,500 images 26 classes</b>	<b>95.58%</b>

**Table (4.23):** Comparing the first proposed system with previous studies for ArSL

Researcher(s), year	Ref.No.	Classification algorithm	The size of dataset	AC
Reema Alzohairi et al. (2018)	[18]	MSVM	210	63.5 %.
<b>Our Proposal (2021)</b>		<b>MSVM</b>	<b>24,000 images 32 classes</b>	<b>96.16%</b>

**Table (4.24):** Comparing the second proposed system with previous studies for ASL

Researcher(s), year	Ref.No.	Classification algorithm	The size of dataset	AC
O. K. Oyedotun and A. Khashman (2016)	[17]	CNN	1440 for training , 600 for testing	92.83%
V. Bheda and N. D. Radpour (2017)	[16]	CNN	650 Images , 25 images from 5 people for each alphabet	67%
S. Masood et al. (2018)	[19]	CNN	2524 ASL gestures	96%
R. Ahuja et al. (2019)	[19]	CNN	47,445 images for 24 classes	99.7%
T.Goswami and S. R. Javaji (2020)	[21]	CNN	27,455 images for 24 classes	99%
<b>Our Proposal (2021)</b>		<b>CNN</b>	<b>19,500 images 26 classes</b>	<b>99.71%</b>

**Table (4.25):** Comparing the second proposed system with previous studies for ArSL

Researcher(s), year	Ref.No.	Classification algorithm	The size of dataset	AC
S. Hayani et.al (2019)	[20]	CNN	5839 images of 28 class	90.02%
M. M. Kamruzzaman (2020)	[22]	CNN	100 images for each alphabet (32 classes)	90%
<b>Our Proposal (2021)</b>		<b>CNN</b>	<b>24,000 images 32 classes</b>	<b>99.03%</b>

Despite using more images for the two proposed systems, all of these tables show high AC.

# Chapter Five

**CONCLUSIONS AND SUGGESTIONS  
FOR FUTURE WORK**

## CHAPTER FIVE

# CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

### 5.1 Conclusion

In this chapter, the proposed system is summarized; the following conclusions were taken from a collection of test results. Some of those conclusions are listed in the following:

1-The testing results from experimentation on the American Sign Language (ASL) and Arab Sign language (ArSL) Datasets indicate that this method is very effective with high accuracy with significantly less time compared to other methods.

2- The proposed system is a reliable methodology to identify and classify static hand gestures with accurate and faster results, also the system succeeded in recognizing the dynamic hand gestures of the two letters j and z in which all previous studies failed to identify and by using CNN and MSVM.

3- The increase in the number of epochs was not random, but rather with the goal of increasing accuracy, which negatively affected the increase in the total run time.

4- In preprocessing stage in the second proposed system, to remove the imperfections in the structure of the image, the best value is chosen for the filter size that fits the image which is [3\*3] for two reasons, firstly is to speed up the operation, secondly is producing image more smoothing.

5- In the feature extraction stage by using the HOG algorithm and PCA, the accuracy of the system increased, when increasing the number of features extracts from each sample until access to 1296 features from each image.

6 - The proposed system has been improved classification accuracy by using the Z-score normalization method, which makes the features and their sigma that are belonging to one class to be more closely related, but at the same time separate them from the other class, to avoid overlapping all features classes that which effect on the accuracy rate of the proposed system negatively.

In addition, the comparison revealed the main reason for the low accuracy of the first proposed system, which is related to the number of classifiers that were classified as well as the large size of the data set. In conclusion of the comparison, it was proved that the second proposed system is the best, fastest, accurate, and most powerful way to recognize and classifies static hand gestures.

## **5.2 Suggestions for Future Work**

The proposed system of static hand gestures recognition of ASL and ArSL is a flexible system and there are many suggestions that can be performed for future work as follows:

1. Design an expert system that can recognize static hand gestures for ASL and ArSL in an automated way depending on multiple techniques and presents ways to use computer vision technologies to assist the deaf and hard-of-hearing communicate more effectively.
2. Applying the system to other various types of sign language as Spanish, Italian, German, and French ... etc., or another dataset as the dataset of digit number for Arabic and English to make the system more comprehensive.
3. Using various machine learning techniques such as (KNN, Native Bayes, etc...) with a comparison between them, also other optimizers for the proposed system to obtain a better accuracy rate.
4. Applying the system to recognize static hand gestures using one hand in Real-time.

5. Applying the system to recognize static hand gestures using two hands.
6. Applying the system to recognize dynamic gestures such as waving or wagging a finger can make HCI much more intuitive

# Reference

- [1] B. Abhishek, K. Krishi, M. Meghana, M. Daaniyaal, and H. S. Anupama, **“Hand gesture recognition using machine learning algorithms,”** *Comput. Sci. Inf. Technol.*, vol. 1, no. 3, pp. 1734–1737, 2020, doi: 10.11591/csit.v1i3.p116-120.
- [2] S. C. Mesbahi, J. Riffi, M.A. Mahraz and H. Tairi, **“Hand gesture recognition based on convexity approach and background subtraction”**, 2018 International Conference on Intelligent Systems and Computer Vision (ISCV). doi:10.1109/isacv.2018.8354074 .
- [3] M. Jin, C. Zaid, O. Mohamed, and H. Jaward, **“A review of hand gesture and sign language recognition techniques,”** *Int. J. Mach. Learn. Cybern.*, vol. 0, no. 0, p. 0, 2017, doi: 10.1007/s13042-017-0705-5.
- [4] M. A. Almasre and H. A. Al-Nuaim, **“Using The Hausdorff Algorithm to Enhance Kinect’s Recognition of Arabic Sign Language Gestures,”** *Nuaim Int. J. Exp. Algorithms*, no. 7, p. 1, 2017.
- [5] S. Nagarajan and T. S. Subashini, **“Static Hand Gesture Recognition for Sign Language Alphabets using Edge Oriented Histogram and Multi Class SVM,”** *International Journal of Computer Applications*, vol. 82, no. 4. pp. 28–35, 2013, doi: 10.5120/14106-2145.
- [6] A. Abraham, P. Krömer, and V. Snášel, **“SIFT-based Arabic Sign Language Recognition System,”** *Afro-European Conf. Ind. Adv. Proc. First Int. Afro-European Conf. Ind. Adv. AECIA 2014 Adv. Intell. Syst. Comput.*, vol. 334, no. November, 2014, doi: 10.1007/978-3-319-13572-4.



- [7] T. Aujeszky and M. Eid, “**A gesture recognition architecture for Arabic sign language communication system**”, *Multimed. Tools Appl.*, vol. 75, no. 14, pp. 8493–8511, 2016, doi: 10.1007/s11042-015-2767-2.
- [8] P. Shah, K. Pandya, H. Shah, and J. Gandhi, “**Survey on Vision based Hand Gesture Recognition**”, *International Journal of Computer Sciences and Engineering*, vol. 7, no. 5, pp. 281–288, E-ISSN: 2347-2693, 2019.
- [9] P. Parvathy, K. Subramaniam, G. K. D. P. Venkatesan, P. Karthikaikumar, J. Varghese, and T. Jayasankar, “**Development of hand gesture recognition system using machine learning,**” *J. Ambient Intell. Humaniz. Comput.*, 2020, doi: 10.1007/s12652-020-02314-2.
- [10] V. Gajjar, “**Hand Gesture Real Time Paint Tool – Box : Machine Learning Approach**” 2017 IEEE Int. Conf. Power, Control. Signals Instrum. Eng., pp. 856–860, 2017.
- [11] M. Oudah, M.,A.Al-Naji, & J.Chahl, (2020), “**Hand Gesture Recognition Based on Computer Vision: A Review of Techniques**”, *Journal of Imaging*, 6(8), 73. doi:10.3390/jimaging6080073.
- [12] M. K. Ahuja and A. Singh, “**Static vision based Hand Gesture recognition using principal component analysis,**” *Proc. 2015 IEEE 3rd Int. Conf. MOOCs, Innov. Technol. Educ. MITE 2015*, pp. 402–406, 2016, doi: 10.1109/MITE.2015.7375353.
- [13] M. M. Islam, S. Siddiqua, and J. Afnan, “**Real time Hand Gesture Recognition using different algorithms based on American Sign Language**” 2017 IEEE Int. Conf. Imaging, Vis. Pattern Recognition, *icIVPR 2017*, 2017, doi: 10.1109/ICIVPR.2017.7890854.

- [14] H. Hasan, and S. Abdul-Kareem, “**Static hand gesture recognition using neural networks**”, 2012 RETRACTED ARTICLE: Static hand gesture recognition using neural networks. *Artificial Intelligence Review*, 41(2), 147–181. doi:10.1007/s10462-011-9303-1
- [15] O. K. Oyedotun and A. Khashman, “**Deep Learning In Vision-Based Static Hand Gesture Recognition**”, *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3941–3951, 2017, doi: 10.1007/s00521-016-2294-8.
- [16] V. Bheda and N. Dianna Radpour, “**Using Deep Convolutional Networks for Gesture Recognition in American Sign Language**”, Available online <https://arxiv.org/abs/1710.06836v3> (accessed 15 August 2018).
- [17] S. Masood, H. C. Thuwal, and A. Srivastava, “**American sign language character recognition using convolution neural network**”, *Smart Innov. Syst. Technol.*, vol. 78, no. October, pp. 403–412, 2018, doi: 10.1007/978-981-10-5547-8\_42.
- [18] R. Alzohairi, R. Alghonaim, W. Alshehri, S. Aloqeely, M. Alzaidan, and O. Bchir “**Image based Arabic Sign Language Recognition System**”, *Int. J. Adv. Comput. Sci. Appl.*, vol. 09, no. 03, pp. 185–194, 2018, [Online]. Available: [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org).
- [19] R. Ahuja, D. Jain, D. Sachdeva, A. Garg, and C. Rajput, “**Convolutional neural network based American sign language static hand gesture recognition**”, *Int. J. Ambient Comput. Intell.*, vol. 10, no. 3, pp. 60–73, 2019, doi: 10.4018/IJACI.2019070104.
- [20] S. Hayani, M. Benaddy, O. El Meslouhi, and M. Kardouchi, “**Arab Sign language Recognition with Convolutional Neural Networks**”, *Proceedings of 2019 International Conference of Computer Science and Renewable Energies, ICCSRE 2019*, doi: 10.1109/ICCSRE.2019.8807586.

- [21] T. Goswami and S. R. Javaji, “**CNN Model for American Sign Language Recognition**”, in ICCCE 2020 Proceedings of the 3rd International Conference on Communications and Cyber Physical Engineering.
- [22] M. M. Kamruzzaman, “**Arabic Sign Language Recognition and Generating Arabic Speech Using Convolutional Neural Network**”, *Wirel. Commun. Mob. Comput.*, vol. 2020, doi: 10.1155/2020/3685614.
- [23] A. Sharma, A. Mittal, S. Singh, and V. Awatramani, “**Hand Gesture Recognition using Image Processing and Feature Extraction Techniques**”, *Procedia Comput. Sci.*, vol. 173, pp. 181–190, 2020, doi: 10.1016/j.procs.2020.06.022.
- [24] M. Al-Hammadi, G. Muhammad, W. Abdul, M. Alsulaiman, and M. S. Hossain, “**Hand Gesture Recognition Using 3D-CNN Model**”, *IEEE Consum. Electron. Mag.*, vol. 9, no. 1, pp. 95–101, 2020, doi: 10.1109/MCE.2019.2941464.
- [25] S. Veluchamy, L.R. Karlmarx and J. Jeya Sudha, “**Vision Based Gesturally Controllable Human Computer Interaction System**”, in 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), p. pp.8-15.
- [26] M.B. Hisham, S.N. Yaakob, R.A. Raof, A.B. Nazren, and N.M. Wafi, “**An Analysis of Performance for Commonly Used Interpolation Method**”, American Scientific Publishers Advanced Science Letters, United States of America, 2016 .

- [27] H. Badi, S. Kareem, and S. Husien, “**Feature Extraction Technique for Static Hand Gesture Recognition**”, Editor: A. Chaudhary, Recent Trends in Hand Gesture Recognition DOI: 10.15579/gcsr.vol3.ch2, GCSR Vol. 3, pp. 19-41, 2015.
- [28] M. Lorentzon, “**Feature extraction for image selection using machine learning**”, M.s.C Thesis in Electrical Engineering Department of Electrical Engineering, Linkoping University, 2017.
- [29] N. Dalal and B. Triggs, “**Histograms of oriented gradients for human detection**“, in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, vol. 1, pp. 886–893.
- [30] A. Challa, “**Automatic Handwritten Digit Recognition On Document Images Using Machine Learning Methods**”, thesis :Master of Science in Computer Engineering January 2019, Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden.
- [31] M. HafizurRahman and J. Afrin, “**Hand Gesture Recognition using Multiclass Support Vector Machine**”, International Journal of Computer Applications, vol. 74, no. 1. pp. 39–43, 2013, doi: 10.5120/12852-9367.
- [32] W. Zhang, “**Dynamic Hand Gesture Recognition Based on 3D Convolutional Neural Network Models**”, 2019 IEEE 16th Int. Conf. Networking, Sens. Control, pp. 224–229, 2019.
- [33] Y. Liu, Y. Ge, F. Wang, Q. Liu, D. Zhang, and G. Lu, “**A Rotation Invariant Hog Descriptor For Tire Pattern Image**“, Classification Center for Image and Information Processing , Xi an University of Posts & Telecommunications No . 618 , Chang ’ an West Road , Xi ’ an City , Shaanxi Province , China, pp. 2412–2416, 2019.

- [34] M. A. Hajizadeh and H. Ebrahimnezhad, “**Classification of age groups from facial image using Histograms of Oriented Gradients**”, 2011 7th Iran. Conf. Mach. Vis. Image Process. MVIP 2011 - Proc., pp. 0–4, 2011, doi: 10.1109/IranianMVIP.2011.6121582.
- [35] M. Turkoglu and D. Hanbay, “**Recognition of plant leaves: An approach with hybrid features produced by dividing leaf images into two and four parts**”, Appl. Math. Comput., vol. 352, pp. 1–14, 2019, doi: 10.1016/j.amc.2019.01.054.
- [36] H. Kamel, D. Abdulah, and J. M. Al-Tuwaijari, “**Cancer Classification Using Gaussian Naive Bayes Algorithm**”, Proc. 5th Int. Eng. Conf. IEC 2019, pp. 165–170, 2019, doi: 10.1109/IEC47844.2019.8950650.
- [37] X. Duanmu, “**Image retrieval using color moment invariant**”, ITNG2010 - 7th Int. Conf. Inf. Technol. New Gener., pp. 200–203, 2010, doi: 10.1109/ITNG.2010.231.
- [38] M. Zhang, W. Cheng, and Y. Wang, “**Multiple-Fault Classification for Hot-Mix Asphalt Production by Machine Learning**”, J. Constr. Eng. Manag., vol. 144, no. 5, p. 04018024, 2018, doi: 10.1061/(asce)co.1943-7862.0001470.
- [39] S. Dong, D. Sun, B. Tang, Z. Gao, W. Yu, and M. Xia, “**A fault diagnosis method for rotating machinery based on PCA and Morlet kernel SVM**”, Math. Probl. Eng., vol. 2014, 2014, doi: 10.1155/2014/293878.
- [40] D.B A and M.K , “**Image Processing Techniques for Hand Gesture and Sign Recognition**”, International Research Journal of Engineering and Technology (IRJET), vol. Volume: 06, no. Issue: 01, 2019, [Online]. Available: [www.irjet.net](http://www.irjet.net).

- [41] C. Maharani, D. A., Fakhrurroja, H., Riyanto, & Machbub, “**Hand Gesture Recognition Using K-Means Clustering and Support Vector Machine**”, IEEE Symp. Comput. Appl. Ind. Electron. (ISCAIE)., 2018, [Online]. Available: doi:10.1109/iscaie.2018.8405435.
- [42] K. P. Murphy, “**Machine Learning, a Probabilistic Perspective**”, J. homepage <https://www.tandfonline.com/loi/uclm20>, no.0933–2480, p. 1104 pages,[Online]. Available: <https://doi.org/10.1080/09332480.2014.914768>.
- [43] T. Le Duc, R. G. Leiva, P. Casari, and P. O. Östberg, “**Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey**”, *ACM Comput. Surv.*, vol. 52, no. 5, 2019, doi: 10.1145/3341145.
- [44] M. A. Almasre and H. Al-Nuaim, “**Comparison of four SVM classifiers used with depth sensors to recognize arabic sign language words,**” *Computers*, vol. 6, no. 2, 2017, doi: 10.3390/computers6020020.
- [45] E. García-Gonzalo, Z. Fernández-Muñiz, P. J. G. Nieto, A. B. Sánchez, and M. M. Fernández, “**Hard-rock stability analysis for span design in entry-type excavations with learning classifiers**”, *Materials (Basel)*., vol. 9, no. 7, pp. 1–19, 2016, doi: 10.3390/ma9070531.
- [46] M. Awad, Y. Motai, J. Näppi, and H. Yoshida, “**A Clinical Decision Support Framework for Incremental Polyps Classification in Virtual Colonoscopy**”, *Algorithms*, vol. 3, no. 1, pp. 1–20, 2010, doi: 10.3390/a3010001.
- [47] G. W. Naji, and J. M. Al-Tuwaijari, “**Satellite Images Scene Classification Based Support Vector Machines and K-Nearest Neighbor**”, *Diyala Journal for Pure Science* 15, No. 03, PP. 70-87, 2019.

- [48] F. F. Chamasemani and Y. P. Singh, “**Multi-class Support Vector Machine (SVM) classifiers - An application in hypothyroid detection and classification**”, Proceedings - 2011 6th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2011. pp. 351–356, 2011, doi: 10.1109/BIC-TA.2011.51.
- [49] M. Achirul Nanda, K. Boro Seminar, D. Nandika, and A. Maddu, “**A comparison study of kernel functions in the support vector machine and its application for termite detection**”, Information, 9(1),5, doi:10.3390/info9010005 .
- [50] B. Scholkopf, C. J.C. Burges, and A. J. Smola, “**Advances in kernel methods: support vector learning**”, book, 1999.
- [51] J. Patterson, and A. Gibson, “**Deep Learning**”, book, Released August 2017 Publisher(s): O'Reilly Media, Inc. ISBN: 9781491914250.
- [52] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, “**Deep learning applications and challenges in big data analytics. Journal of Big Data**”, <https://doi.org/10.1186/s40537-014-0007-7>Deep learning , Journal of Big Data, vol. 2, no. 1. p. 1, 2015, [Online]. Available: <http://www.journalofbigdata.com/content/2/1/1>.
- [53] M. Mustafa, “**A study on Arabic sign language recognition for differently abled using advanced machine learning classifiers**”, Journal of Ambient Intelligence and Humanized Computing. 2020, doi: 10.1007/s12652-020-01790-w.
- [54] L. T. Bhavanam and G. N. Iyer, “**On the Classification of Kathakali Hand Gestures Using Support Vector Machines and Convolutional Neural Networks**”, in 2020 International Conference on Artificial Intelligence and Signal Processing (AISP).

- [55] S. Albawi, T. A. M. Mohammed, and S. Alzawi, “**Understanding of a Convolutional Neural Network,**” Ieee. 2017, [Online]. Available: <https://wiki.tum.de/display/lfdv/>.
- [56] X. Kang, B. Song, and F. Sun, “**A deep similarity metric method based on incomplete data for traffic anomaly detection in IoT**”, Applied Sciences (Switzerland), vol. 9, no. 1. 2019, doi: 10.3390/app9010135.
- [57] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, “**Medical Image Analysis using Convolutional Neural Networks: A Review**” J. Med. Syst., vol. 42, no. 11, 2018, doi: 10.1007/s10916-018-1088-1.
- [58] G. Li et al., “**Hand gesture recognition based on convolution neural network**” Cluster Comput., 2017, doi: 10.1007/s10586-017-1435-x.
- [59] X. Jiang, Y. Wang, W. Liu, S. Li, and J. Liu, “**CapsNet, CNN, FCN: Comparative performance evaluation for image classification**”, International Journal of Machine Learning and Computing, vol. 9, no. 6. pp. 840–848, 2019, doi: 10.18178/ijmlc.2019.9.6.881.
- [60] S. Skansi, “**Introduction to Deep Learning from Logical Calculus to Artificial Intelligence**”, Springer, 2018, undergraduate Topics in Computer Science, <https://doi.org/10.1007/978-3-319-73004-2>.
- [61] G. Castaneda and P. Morri, and T. M. Khoshgoftaa, “**Evaluation of maxout activations in deep learning across several big data domains**”, J. Big Data, doi: <https://doi.org/10.1186/s40537-019-0233-0>.
- [62] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “**Activation functions: Comparison of trends in practice and research for deep learning**”, arXiv:1811.03378v1 [cs.LG] 8 Nov 2018.



- [63] W. Pedrycz and S. Chen, “**Deep Learning: Algorithms and Applications**”, vol. 865, no. 2. Springer Nature Switzerland AG 2020.
- [64] B. Xu, N. Wang, T. Chen, and M. Li, “**Empirical Evaluation of Rectified Activations in Convolutional Network**”, 2015, [Online]. Available: <http://arxiv.org/abs/1505.00853>.
- [65] A. Gupta and R. Duggal, “**P-TELU: Parametric Tan Hyperbolic Linear Unit Activation for Deep Neural Networks**” ,Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017, vol. 2018-Janua. pp. 974–978, 2017, doi: 10.1109/ICCVW.2017.119.
- [66] A. A. Alani, G. Cosma, A. Taherkhani, and T.M McGinnity, “**Hand Gesture Recognition Using an Adapted Convolutional Neural Network with Data Augmentation**”, 2018 4th International Conference on Information Management (ICIM), doi:10.1109/infoman.2018.8392660
- [67] W. Rawat and Z.Wang, “**Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review**”, Article in Neural Computation June 2017, doi: 10.1162/NECO\_a\_00990.
- [68] M. R. Islam, U. K. Mitu, R. A. Bhuiyan, and J. Shin, “**Hand gesture feature extraction using deep convolutional neural network for recognizing American sign language**”, 2018 4<sup>th</sup> ICFSP 2018, no. September, pp. 115–119, 2018, doi: 10.1109/ICFSP.2018.8552044.
- [69] M.ZahangirAlom, T.M.Taha, C. Yakopcic,S.Westberg, *P.Sidike*, *S.Nasrin* , B. C Van Esesn, A. A S. Awwal, and V. K. Asari, “**The History Began From Alexnet: A Comprehensive Survey On Deep Learning Approaches**” , <https://arxiv.org/abs/1803.01164> , 2018.

- [70] Han, M., Chen, J., Li, L., & Chang, Y. (2016). "**Visual hand gesture recognition with convolution neural network**", 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing.
- [71] A. El Korchi and Y. Ghanou, "**DropWeak: A novel regularization method of neural networks**", *Procedia Comput. Sci.*, vol. 127, pp. 102–108, 2018, doi: 10.1016/j.procs.2018.01.103.
- [72] A. Gummeson, "**Prostate Cancer Classification using Convolutional Neural Networks**", Master's Theses in Mathematical Sciences. 2016.
- [73] P. Bao, A.I. Maqueda, C. R. del-Blanco, and N. García, "**Tiny hand gesture recognition without localization via a deep convolutional network**", 2017 IEEE Transactions on Consumer Electronics, 63(3), 251–257. doi:10.1109/tce.2017.014971.
- [74] S. Vani and Dr. T. V. Madhusudhana, "**An Experimental Approach towards the Performance Assessment of Various Optimizers on Convolutional Neural Network**", in Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019), p. IEEE Xplore Part Number: CFP19J32-ART; ISBN: 978-1.
- [75] M.Yaqub , J.Feng , M. Sultan Zia, K. Arshid , K. Jia , Z. Ur Rehman and A. Mehmood, "**State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images**", *Brain Sciences*, 10(7), 427. doi:10.3390/brainsci10070427, 2020.
- [76] B. Xiao, Y. Liu, and B. Xiao, "**Accurate state-of-charge estimation approach for lithium-ion batteries by gated recurrent unit with ensemble optimizer,**" *IEEE Access*, vol. 7, pp. 54192–54202, 2019, doi: 10.1109/ACCESS.2019.2913078.

- [77] T.Thu-Huong Le, J. Kim, and H. Kim, “**An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization**” , Int. Conf. Platf. Technol. Serv, no. November, 2017, doi: 10.1109/PlatCon.2017.7883684.
- [78] S. Imran, Y. Long, X. Liu, and D. Morris, “**Depth Coefficients for Depth Completion**”, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, doi:10.1109/cvpr.2019.01273.
- [79] X. Deng, Q. Liu, Y. Deng, and S. Mahadevan, “**An improved method to construct basic probability assignment based on the confusion matrix for classification problem**” , Inf. Sci. (Ny)., vol. 340–341, pp. 250–261, 2016, doi: 10.1016/j.ins.2016.01.033.
- [80] K. Kurnianingsih, K. Hamed, L. Edi Nugroho, Widyawan, L. Lazuardi, A. Satria Prabuwo, and T. Mantoro, “**Segmentation and Classification of Cervical Cells Using Deep Learning**”, IEEE Access, Digital Object Identifier 10.1109/ACCESS.2019.2936017, vol. 7, 2019.
- [81] P. B. Shull, S. Jiang, S. Member, and Y. Zhu, “**Hand Gesture Recognition and Finger Angle Estimation via Wrist-Worn Modified Barometric Pressure Sensing**”, IEEE Trans. Neural Syst. Rehabil. Eng., vol. PP, no. c, p. 1, 2019, doi: 10.1109/TNSRE.2019.2905658.
- [82] <https://www.kaggle.com/grassknotted/asl-alphabet>.
- [83] <https://data.mendeley.com/datasets/y7pckrw6z2/1>.

## الخلاصة

ان عملية تحديد كل حرف على حدة مهمة جدا وبهذا ، أصبح التعرف على لغة الإشارة تقنية مهمة في الذكاء الاصطناعي والتعلم الآلي.

تقدم هذه الأطروحة نظامين مقترحين للتعرف على إيماءات اليد الثابتة بناءً على خوارزميات التعلم الآلي والتعلم العميق ، حيث يتم استخدام عدة خطوات في شكل مراحل ؛ الحصول على الصور والمعالجة المسبقة للصور واستخراج الميزات والتصنيف. في النظام الأول المقترح ، يتم استخدام الرسم البياني للتدرجات الموجهة (HOG) لاستخراج الميزات من كل صورة ثم يتم تطبيق آلة متجهية داعمة متعددة الفئات (MSVM) باستخدام نتيجة HOG للصور لأداء عملية التصنيف. في النظام الثاني المقترح ، يتم استخدام الشبكة العصبية الالتفافية (CNN) والتي يتم من خلالها التعرف على إيماءات اليد الثابتة وفقاً لهيكل خاص لهذه الخوارزمية التي تتكون من عدة طبقات.

كانت الأعمال والأبحاث السابقة في هذا المجال معقدة للغاية وبدقة مختلفة. النتائج التي تم الحصول عليها ، النظام الثاني المقترح الذي اعتمدت التعليم العميق باستخدام نموذج CNN يتفوق على النظام الأول من حيث الأداء والدقة ، وكان معدل الدقة الذي تم الحصول عليه من النظام الثاني المقترح (99.71%) للغة الإشارة الأمريكية (ASL) و ( 99.03% للغة الإشارة العربية (ArSL) ، بينما كانت نسبة الدقة التي تم الحصول عليها من النظام المقترح الأول (95.58%) لـ ASL، و (96.16%) بالنسبة لـ ArSL



جمهورية العراق  
وزارة التعليم العالي والبحث العلمي  
جامعة ديالى  
كلية العلوم  
قسم علوم الحاسوب



## مقارنة بين خوارزميات MSVM و CNN في التعرف على إيماءات اليد في وضع عدم الاتصال

رسالة مقدمة  
الى كلية العلوم في جامعة ديالى وهي جزء من متطلبات نيل  
شهادة الماجستير في علوم الحاسوب  
تقدمت بها الطالبة

هند ابراهيم محمد سبيع

بإشراف

أ.م.د. جمانة وليد صالح

2021 م

1443 هـ