Republic of Iraq

Ministry of Higher Education and Scientific Research

University of Babylon

College of Information Technology

Software Department

Deep Learning Classification Models for Diabetes Mellitus and its Complications from Multiple Sources

A Dissertation

Submitted to the Council of the College of Information Technology at the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctorate of Philosophy in Information Technology/ Software

By

Hussein Ali Ismael Hameed

Supervised by

Prof. Dr. Nabeel Hashem Al-A'araji

Prof. Dr. Baheeja Kudayir Shukur AL-helo

2023 A.C. **1445 A.H.**

بِسْـمِ اللَّهِ الرحمن الرَّحِيـمِ

﴿وَعَدَ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَعَمِلُوا الصَّالِحَاتِ لَيَسْتَخْلِفَنَّهُمْ فِي الْأَرْضِ كَمَا اسْتَخْلَفَ الَّذِينَ مِن قَبْلِهِمْ وَلَيُمَكِّنَنَّ لَهُمْ دِينَهُمُ الَّذِي ارْتَضَىٰ لَهُمْ وَلَيُبَدِّلَنَّهُم مِّن بَعْدِ خَوْفِهِمْ أَمْنًا ۚ يَعْبُدُونَنِي لَا يُشْرِكُونَ بِي شَيْئًا ۚ وَمَن كَفَرَ بَعْدَ ذَٰلِكَ فَأُولَٰئِكَ هُمُ الْفَاسِقُونَ﴾

صدق الله العلي العظيم

سورة النور / آية 55

Supervisor Certification

I certify that this dissertation was prepared under my supervision at the Department of Software / Collage of Information Technology / Babylon University, by **Hussein Ali Ismael** as a partial fulfillment of the requirements for the degree of **Ph.D. in Information Technology**.

Signature:
Name: **Prof. Dr. Nabeel Al-A'araji**
Title: **Professor**
Date: / / 2023

Signature:
Name: **Prof. Dr. Baheeja Kudayir Shukur AL-helo**
Title: **Professor**
Date: / / 2023

The Head of the Department Certification

In view of the available recommendation, we forward this dissertation for debate by the examining committee.

Signature:
Name: **Dr. Sura Z. Al-Rashid**
Title**:** **Assist. Professor**
Date: / / 2023

Declaration

I hereby declare that this dissertation, **Deep Learning Classification Models for Diabetes Mellitus and its Complications from Multiple Sources** submitted to the University of Babylon in partial fulfillment of requirements for the degree of Doctorate of Philosophy in Information Technology-Software has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for reports and summaries whose sources are appropriately cited in the references.

Signature:

Name: **Hussein Ali Ismael hameed**

Title: **Ph.D. Student**

Date: / / 2023

Acknowledgments

First, greatest praise and thanks to Allah almighty and with the blessings of His beloved Messenger, Abu Al-Qasim Muhammad Ibn Abdullah, and his good and pure family for supporting me in overcoming all the problems and difficulties that I faced in finishing this dissertation.

I would like to offer deep gratitude and sincere thanks to my supervisors **Prof. Dr. Nabeel Al-A'araji** and **Prof. Dr. Baheeja Kudayir Shukur AL-helo** for their support, guidance, and untiring efforts during this work.

I owe thanks to my family, especially my mother, my father, my brothers, and my wife for their support, prayers for me, encouragement, and care throughout my years of study and through researching and writing this dissertation.

I would like to express my thanks and respect to all My professors and staff of Information Technology College, especially **Prof. Dr. Eman Salih Al-Shamery, Asset. Dr. Sura Z. Al-Rashid, and Prof. Dr. Ahmed Saleem Abbas** for the great assistance they provided me.

Special thanks to **Laith Alzubaidi, Mohammed Abdulraheem Fadhel, and Simon Mueller** for their advice and assistance during the work.

Last but not least, most importantly, I would like to thank all the kind, helpful, and lovely people who helped me directly or indirectly to complete this work and apologize to them for not being able to mention them by name here, though they are in my heart.

Hussein Ali Ismael

Dedication

To the savior of mankind Al-Imam Al-Mehdi (peace be upon him) …
Who will fill the earth with justice and equity, after it has been filled with injustice and oppression? The survival of religion in the earth.

To my mother …

To my father …

To my brother and sisters …

To my wife and children **…**

Abstract

Diabetes mellitus is a chronic metabolic disorder that affects millions of individuals worldwide. With its complications, this disease poses a significant health risk. There are many successful models for classifying diabetes and its complications. Although, none of these models dealt with multiple data sources. This dissertation proposes a new approach to deal with the heterogeneity of data sources, including tabular, Electrocardiogram (ECG), and image data types. Moreover, modified deep learning models are used for classifying diabetes and its complications (diabetes foot ulcer, diabetes retinopathy) by utilizing multiple data sources.

The proposed approach processes each data type with a dedicated model, where each model has three key stages: data preprocessing, feature extraction, and classification stages. There are four proposed models. The first model, called Complete Autoencoder with Regulation and Deep Neural Network (CAER-DNN), is designed for classifying diabetes by handling tabular data sources. The main contribution of this model is using an unsupervised technique to regenerate the essential features that act as the input to the DNN classifier. The second model, called ECG-DNN, is suggested to deal with ECG data sources. This model is unique by leveraging statistical tools and Convolutional Neural Network (CNN) techniques to extract important features from the heart rate variability signal. The third model called (CNN-GLCM) addresses diabetic foot ulcer classification by utilizing image data sources. The origin of this model is a combination of the power of the Gray-Level Co-occurrence Matrix (GLCM) method and CNN for feature extraction. In addition to the classification, this model clusters the Diabetes Foot Ulcer (DFU) case into three clusters based on severity. The fourth model, called CLAHE-CNN, is specialized for Diabetic Retinopathy (DR) classification using an image data source. This model leverages Contrast-Limited Adaptive Histogram Equalization (CLAHE) to enhance image contrast, coupled with CNN for feature extraction.

Finally, an aggregated model is proposed to aggregate the model's classifications and get a final decision. This model is designed to take multiple data sources into account. The CAER-DNN model is used to process tabular data and achieves an accuracy of 98.90% compared with ten state-of-the-art based on the Pima Indian Diabetes (PID) dataset and 98.48% compared with one state-of-the-art based on the Mendeley Diabetes Dataset (MDD) dataset. To deal with ECG data, the ECG-DNN model is applied, with an accuracy score of 99.88% compared with two state-of-the-art. To address image data, two models are utilized (GLCM-CNN and CLAHE-CNN). First, DFU classification, the GLCM-CNN model achieves an accuracy of 97.43% compared with five state-of-the-art. Second, foot ulcer clustering, the Winner-Take-All Competitive Networks (WTA-CN) achieves an average silhouette of 0.429. The CLAHE-CNN model achieves an accuracy of 94.0% compared with two state-of-the-art for DR classification.

Table of Contents

List of Algorithms

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
ANN	Artificial Neural Network
BMI	Body Mass Index
BP	Back Propagation
CAER	Complete Autoencoder with Regularization
CAN	Cardiovascular Autonomic Neuropathy
CEB	Cross Entropy Binary
Chol	Cholestrol
CLAHE	Contrast Limited Adaptive Histogram Equalization
CM	Confusion Matrix
CNN	Convolutional Neural Network
CR	Creatinine
CV	Cross Validation
DA	Data Augmentation
DFU	Diabetic Ulcer Foot
DL	Deep Learning
DLC	Deep Learning Classification
DM	Diabetic Mellitus
DNN	Deep Neural Network
DR	Diabetic Retinopathy
DT	Decision Tree
ECG	Electrocardiogram
EHRs	Electronic Health Records
FANN	Forword Artificial Neural Network
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GD	Gradient Descent
GLCM	Gray-Level Co-Occurrence Matrix
HBA1C	Hemoglobin A1C
HDL	High-Density Lipoprotein
HRV	Rate Heart Variability
KNN	K Nearest Neighbor
LDL	Low-Density Lipoprotein
LR	Linear Regression

Abbreviation	Meaning
MDD	Mendeley Diabetes Dataset
ML	Machine Learning
MLP	Multilayer Perceptron
MLR	Multinomial Logistic Regression
MSE	Mean Square Error
NB	Naïve Bayes
NN	Neural Network
PCA	Principal Component Analysis
PID	Pima Indian Diabetes
PSD	Power Spectral Density
ReLU	Rectified Linear Unit
RF	Random Forest
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SD	Standard Deviation
SGB	Stochastic Gradient Boosting
SL	Supervised Learning
SLP	Single Layer Perceptron
SSL	Semi-Supervised Learning
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TG	Triglycerides
TN	True Negative
TP	True Positive
USL	Unsupervised Learning
VLDL	Very Low-Density Lipoprotein
WTA-CN	Winner-Take-All Competitive Network

List of Dissertation-Related Publications

(First Paper)

- **Name of Journal:** Karbala International Journal of Modern Science.
- **Paper Title:** An Enhanced Diabetic Foot Ulcer Classification Approach Using GLCM and Deep Convolution Neural Network
- **Scopus: Q2 site score 2.2**
- **Authors:**
    - Hussein A. Ismael.
    - Prof. Dr. Nabeel H. Al-A'araji.
    - Prof. Dr. Baheja khudair shukur.

Software Department, Information Technology College, Babylon University.

College of Computer Science & Information Technology, University of Kerbala, Kerbala, Iraq.

Email: husseinyessari@uobabylon.edu.iq, husseinyessari@hotmail.com.

(Second Paper)

- **Name of Journal:** Periodicals of Engineering and Natural Sciences.
- **Paper Title:** Enhanced the classification approach of diabetes using an autoencoder with regularization and deep neural network.
- **Scopus: Q2 site score 2.1**
- **Authors:**
    - Hussein A. Ismael.
    - Prof. Dr. Nabeel H. Al-A'araji.
    - Prof. Dr. Baheja khudair shukur.

Software Department, Information Technology College, Babylon University.

College of Computer Science & Information Technology, University of Kerbala, Kerbala, Iraq.

Email: husseinyessari@uobabylon.edu.iq, husseinyessari@hotmail.com.

CHAPTER ONE:

Introduction

Introduction

1.1 Overview

The healthcare sector has resisted the use of machines for a long time and has remained human-dominated. This was largely because, until recently, most machines could not match the breadth of information, finesse, precision, and competence that doctors possess. But when machines began to do things that had previously appeared nearly impossible, everything began to change. X-ray equipment, CT scanners, and other diagnostic tools were standard in all medical settings. However, it was still believed that humans would always be superior to machines. However, it was discovered that machines may become somewhat "intelligent" with the development of artificial intelligence. Additionally, research has demonstrated that a joint effort between doctors and AI increased performance in every area where this collaboration took place. As a result, the use of AI technologies such as Machine Learning (ML) and Deep Learning (DL) by scientists makes it possible to treat and complete a wide range of tasks, such as diagnostic and suitable drug selection, classification of risk and sickness stratification, reduction of medical error, and enhancement of productivity [1], [2].

ML is the essential technology to process data in a meaningful way that is larger than what the human brain can understand [3]. ML techniques permit computers to perform complex assignments. ML includes components from computer science, statistics, and math. Where ML computer-based models are trained using a large amount of data points. Instead of explicitly encoding the existing information about specific data items and relationships between data elements into the model, it is learned through the iterative mapping between outputs and inputs. Because of this, a collaboration between ML and clinical experts is essential, and several computational techniques incorporate model parameters with varying degrees of clinical experience [2], [4].

Presently, the four primary categories of ML model creation are supervised learning (SL), unsupervised learning (USL), reinforcement learning (RL), and semi-supervised learning (SSL). In this dissertation, two principles were discussed, namely SL and USL. In short, supervised ML models have known output labels, like a disease. The goal is to create a computational tool that can predict an output from the input dataset (the output is typically referred to as a class label, whereas the inputs are referred to as features). The model is learned by analyzing the training dataset, which consists of several observations. Each of these has values for its features and labels. A model that has been perfectly trained will generate an accurate output that matches the input from the training set. Classification and regression are common tasks in supervised ML. Usually, the integrity of the data representation as an input has a significant impact on how well an ML approach performs. It has been demonstrated that a perfect data representation outperforms a bad one in terms of performance. Consequently, feature engineering has been a prominent study trend in ML for a long time. The objective of this approach is to generate features directly from raw data. Moreover, it is highly specialized in a particular field and often demands substantial human involvement. For example, histograms of oriented gradients and scale-invariant-feature transform in computer vision. In contrast, the feature extraction process in DL is automated, unlike traditional ML techniques, with the least amount of human labor and technical expertise[2], [4], [5].

The DL, one of the several ML methods, is frequently used in medical applications. Due to the tremendous advancements achieved in hardware technologies, such as high performance-computing, and the unpredictably growing development in the ability to gather data, new studies in the disciplines of deep and distributed learning continue to be published. DL evolves from the traditional artificial neural network and performs noticeably better than its predecessors. DL is executed by Deep Neural Networks (DNNs). DNNs are the multi-layers neural networks that have two or more hidden-layers. A range of applications, including image and speech recognition, and natural language processing has seen excellent performance based on DL techniques. Currently, DL techniques outperform compared with traditional machine learning techniques and become more scalable, and the high precision can be obtained by using the huge training dataset or increasing the size of neural networks (more hidden layers). One of the most well-known and widely utilized DL networks is the Convolutional Neural Network (CNN). It incorporates the feature extraction operation into the training process instead of manually developing them. As it is widely used for image recognition, especially medical images, this is due to its great ability to extract important features compared to traditional ML methods [6],[7].

On the other hand, the training data for unsupervised ML consists of inputs without outputs (labels). Models of unsupervised ML are designed to find connections (e.g., relationships) in data that humans would not otherwise notice. Using similarities across data sets, its algorithms must output groups and classes. Thus, uncovers previously unknown patterns, making classifications about undiscovered outcomes. Clustering is an example of an unsupervised ML approach. The last one involves the process of grouping or clustering data. For example, in the medical field, to solve the problem of identifying people who display a high risk of dementia, unsupervised learning algorithms try to understand the underlying structure of people and separate them into different groups.

Such that members of the same group have things in common yet differ from those of other groups. Additionally, USL is used to enhance the performance of ML[8].

The most typical problems that data scientists encounter when developing solutions are as follows: First, inadequate labeled data, a large amount of labeled data is required, which is difficult and expensive to produce to employ supervised algorithms. Unlabeled samples can be automatically labeled using USL. Cluster all the instances, using the labels from the labeled samples in the same cluster, and apply them to the unlabeled examples within the cluster. There are several clustering algorithms such as K-mean, DBSCAN, and ANN with competitive learning. Second, curse of dimensionality, supervised algorithms must develop their ability to separate points and approximate functions to work well (e.g., perfect decisions) in a very high-dimensional domain. To solve this problem, it is needed to apply unsupervised techniques to reduce the dimensionality such as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Third, feature engineering is one of the most important things done by scientists to data. SL relies heavily on the extracted features[9]. In a supervised learning algorithm, instead of being given the original input data straight away, can input a new representation of the initial input data that we will produce by the USL technique such as an Autoencoder network. Most of the previously mentioned techniques are of great importance and effectiveness in the field of medical healthcare[9], [10].

1.2 Problem Statement

The prevalence of diabetes on a global scale has reached alarming levels, imposing a significant strain on healthcare systems and public health resources. Although attempts to enhance the diagnosis and treatment of diabetes, an important gap exists in classifying the diabetes and its complications accurately for those at high risk of diabetes. Since, early diagnosing and caring play a crucial role in reducing the advancement of the disease and the occurrence of incapacitating consequences, such as retinopathy, ulcer foot, cardiovascular disease, and nephropathy. Moreover, the existing classification models exhibit certain limits. These existing models is focusing in single data source to make the classification and it show incapability to utilize multiple data sources to come up with final decision. Specially, no previous works has linked ECG data to process of making the classification in region of Iraq. In addition to this, the accuracy achieved previously is not enough for the high risk of this disease and its complication. Therefore, propose a more complete methodology that encounter the limitations of multiple data source and increase the accuracy score is necessity to handle this disease.

1.3 Dissertation Objectives

The aim of this dissertation is to proposed an AI-based system that utilizes multi-deep learning and multi-source data to predict the occurrence of diabetes and its complications in a healthcare setting. The system will use various data modalities, such tabular data, ECG information data, and medical imaging, to create a comprehensive predictive model. This aim can be summarized as follows:

1. Design a proposed system to classify diabetes mellitus and its complications based on deep learning models with multiple data sources.

2. Design a model to classify diabetic mellitus based on an autoencoder technique and deep neural network with tabular data
3. Build a model using ECG data as an input and based on statistical techniques and a convolution neural network for the classification of diabetic mellitus.
4. Build a model that relies on convolution neural network and Gray Level Co-Occurrence Matrix (GLCM) techniques for the classification of diabetic ulcer foot with images data as an input.
5. Build a model based on Contrast Limited Adaptive Histogram Equalization (CLAHE) enhanced technique and convolution neural network technique for the classification of diabetic retinopathy with images data as an input.
6. Design a model for clustering the state of diabetic foot ulcers into three clusters, which represent the state of foot injury.

1.4 Dissertation Contributions

1. It is the first dissertation based on multiple data sources include tabular, ECG, image data type for classification the diabetes disease and its complications, which be executed depending on deep learning techniques.
2. This dissertation is the first that contributes in Iraq to the Classify of the diabetes mellitus based on ECG signal data and enhanced the accuracy metric of the classification by using statistical methods and CNN technique to extract important features.
3. The classification accuracy of GLCM-CNN model with DFU image data type is improved using GLCM and CNN to extract the important features that are used for the classification of the diabetic ulcer foot.

4. This thesis addressed the problem of knowing the specific condition of diabetic foot ulceration by clustering it into three clusters (Mild, Moderate, and Severe) to determine the condition of the injury through the use of competitive neural network with winner-take-all techniques.
5. The dissertation also addresses the problem of selection the best hyperparameters of the GLCM method based on GA technique
6. The classification accuracy of the CAER-DNN with tabular data is improved based on Complete Autoencoder with Regularization and Deep Neural Network through regenerate and select important features.

The contributions may enhance patient outcomes, permit proactive healthcare treatments, and help healthcare professionals deliver more focused and efficient care.

1.5 Dissertation Challenges

There are several challenges as follows:

1. Variety: The difference and diversity of data, which includes structured and unstructured data.
2. Data collection and lack of sufficient labeled data: The presence of a restricted amount of labeled data has the potential to result in overfitting or inadequate generalization of the model.
3. Feature extraction is a crucial step in the development of deep learning models, as it involves the extraction of significant patterns from raw data.

1.6 Related Works

Many research projects use artificial intelligence (AI) techniques to predict diabetes mellitus and its complications. It can be summarized as follows (i) constructing and developing approaches based on ML techniques, (ii) constructing and developing approaches based on DL techniques.

1.6.1 Related Works for Classification of Diabetes with Tabular Data:

Özmen and Özcan [11] proposed new approaches for classifying diabetes. First, (ANN-GA) approach depends on a genetic algorithm and Artificial Neural Networks (ANN). The second, (CART-GA) approach is based on classification and regression tree. This study used GA to enhance the accuracy and CART-GA gives a high accuracy of 93.63%.

Research by Orukwo and Kabari [12] developed a model to predict the type of diabetes (type I or type II) based on the ANN classifier. In this study, the ANN consists of an input layer with 18 neurons that represent the symptoms of diabetes, one hidden layer with two neurons, and an output layer with two outputs representing the type of diabetes. They used a private dataset consisting of 100 samples collected from diabetes patients in a survey.

Gupta, Varshney, Sharma, Pachauri, and Verma [13] proposed two techniques DL and Quantum Machine Learning (QML). They used MLP to train and test a dataset that consists of five hidden layers with (16, 32, 8, and 2) neurons respectively. QML used the advantage of quantum computers to solve the classification of diabetes. It has been demonstrated to be capable of efficiently resolving these issues with current technology by calculating several states concurrently. The authors proved the DL techniques achieved high performance as compared with QM.

Abedini, Bijari, and Banirostam [14] suggested a new model for diagnosing diabetes. In this work, there are two phases. First, used an ensemble model that depends on two classifiers Decision Tree (DT), and Logistic Regression (LR), which trains independently. Second, using an ANN classifier that trains the output of the first phase to enhance the final decision. It achieved 83% accuracy.

Chang, Bailey, Xu, and Sun [15] applied a new E-Diagnosis system for classifying type II diabetes. It relies on ML techniques that are implemented on the internet of things for the medical environment. This study, used Random Forest (RF), Naïve Bayes (NB), and J48 Decision Tree (J48DT) as classifiers to train and test the Pima Indian dataset (PID). Then choose the best one in terms of accuracy and performance. They prove the NB classifier with feature selection achieved high performance. While RF achieved high with more features.

Alasaady, Noranis, Aris, Sharef, and Hamdan [16] suggest an approach to the diagnosis of diabetes relies on an Adaptive Neuro-Fuzzy Inference System (ANFIS). This approach consists of several phases including, the preprocessing phase, classification phase, and evaluation. The first, phase includes normalization, outlier detection, and imputation. Second. Apply ANFIS classifier to predict the disease. Finally, evaluate the model. The authors proved the model achieved the highest performance in accuracy.

1.6.2 Related Work for Classification Diabetic Foot Ulcer

Alzubaidi, Abbood, Fadhel, Al-Shamma, and Zhang [17] proposed four models that make use of the hybrid convolution neural network to classify a patient's skin into either normal or abnormal. In their work, the models relied upon the network of multiple branches. These four deep aggregated models are built based on the combination of traditional and multiple-branch convolutional layers. Different filter sizes are applied to parallel conv-layers on the same input images. The obtained features were concatenated to get better features for the classification.

Alzubaidi, Fadhel, Oleiwi, Al-Shamma, and Zhang [18] applied the DFU QUTNet model, which is a new Deep Convolutional Neural Network. It has been introduced for categorizing skin automatically into normal (healthy) and abnormal (DFU) skin. In this study, the model is based on increasing the network width while retaining the network depth to extract the best features. These extracted features are used by SVM and K-Nearest Neighbors (KNN) classifiers.

Reddy, Mahesh, and Preethi [19], the authors proposed approaches that rely on ML methods to predict DFU. In their study, a new neural network called Extreme ML is suggested for classification. In addition, each of KNN, ANN, and SVM are applied to predict DFU. Such an extreme ML method achieved a high accuracy score.

Amin, Sharif, Anjum, Khan, Malik, and Kadry [20] proposed a model for classifying and localizing the types of DFU (Ischaemia) and (infection) into normal or abnormal. In their work, the CNN model is used for feature extraction using shallow classifiers (KNN, DT, Ensemble, NB, SoftMax), after which the best classifier is selected. The YOLOv2-DFU model is used for the detection and localization of infections or ischemia.

Das, Roy, and Mishra [21], the authors proposed a new network architecture that depends on unique stacked parallel convolution layers to classify a skin into DFU and normal skin. In their study, the model consists of three blocks of convolution layers that are parallel with different kernel sizes, to extract abstract features from local and global regions.

Goyal, Reeves, Rajbhandari, Ahmad, Wang, and Yap [22] suggest an approach based on the handcrafted ML method and ensemble CNN. In their work, a new feature descriptor called Superpixel Colour Descriptor is developed to extract color features from DFU image regions for identifying Ischaemia and infection in DFU images.

1.6.3 Related Work for Classification Diabetic retinopathy

Jadhav, Pati, and Biradar [23] have introduced a computer-aided diagnosis strategy for diabetic retinopathy (DR) using intelligent learning methods, such as neural networks, combined with an enhanced thresholding technique. This methodology consists of several stages: image preprocessing, blood vessel segmentation, feature extraction, and classification. They employ the Modified Levy Updated-Dragonfly Algorithm (MLU-DA) to enhance gray-level thresholding and neural network methods, resulting in improved segmentation accuracy and a reduction in classification errors.

Tsao, Chan, and Su [24] investigates risk factors associated with diabetic retinopathy (DR) in type 2 diabetes mellitus and aims to create a predictive model using data mining techniques including the support vector machines, decision trees, artificial neural networks, and logistic regressions. Support vector machines outperformed other algorithms, achieving 79.5% accuracy. By incorporating previous research methods, the study surpassed prior studies in most measures. Notably, the model identifies insulin use and diabetes duration as key features for identifying high-risk DR populations. Each additional year of diabetes duration increases the odds of DR by 9.3%, and insulin use raises the odds by 3.561 times. These findings can inform the development of clinical decision support systems for future practice.

1.6.4 Related Work for Classification Diabetic based on HRV in ECG data

Swapna, Vinayakumar, and Soman [25] proposed a methodology that utilizes deep learning architectures, specifically long short-term memory (LSTM) and convolutional neural network (CNN), for extracting complex temporal dynamic features from HRV signals that are produced from ECG signals. Then used (SVM) for classification. The proposed classification system achieves a high accuracy of 95.7%, which can aid clinicians in diagnosing diabetes using ECG signals.

Seyd, Joseph, and Jacob [26] presented an automated diagnostic system for diabetes mellitus (DM) using heart rate variability (HRV) measures and a feed-forward neural network. The results indicate a high accuracy of 93.08%, with a specificity of 96.92% and a sensitivity of 89.23%. This automated system shows promise in effectively identifying DM patients using HRV signal, providing a potential tool for early detection and intervention.

Aggarwal, Das, Mazumder, Kumar, and Sinha [27] proposed an approach for DM detection based on heart rate variability (HRV) parameters. In this study, artificial neural network (ANN) and support vector machine (SVM) algorithms. Thirteen nonlinear HRV parameters were computed from the ECG data and used for training and testing the ANN and SVM models. The results showed that the ANN achieved a classification accuracy of 86.3% using the nonlinear HRV parameters as inputs. The SVM model performed relatively better with an accuracy of 90.5%. Table 1.1 shows a recap of the earlier research with further information.

Table 1.1 summary of literature reviews

References	Year	Title	Methodology	Best Results
Tabular dataset for classification DM				
[11]	2020	Diagnosis of diabetes mellitus using artificial neural network and classification and regression tree optimized with genetic algorithm	ANN-GA CART-GA, DT-CART, ANN	CART-GA Acc: 93.63%
[14]		Classification of Pima Indian Diabetes Dataset using Ensemble of Decision Tree, Logistic Regression, and Neural Network	Ensemble of DT, LR, and ANN	Acc:83.08% Pre:98.57% Rec:83.13%
[12]	2020	Diagnosing Diabetes Using Artificial Neural Networks	ANN	Acc:90.00%
[13]	2021	Comparative performance analysis of quantum machine learning with deep learning for diabetes classification	DL and QML	DL Acc:95.00% Pre: 90.00% Rec:95.00%
[15]	2022	Pima Indians diabetes mellitus classification	RF, NB, and J48DT using three approaches (all	RF with all features: Acc:79.13% Pre: 89.40%

		based on machine learning (ML) algorithms	features, 3-factor selection, 5-factor selection)	fscor:85.17%
[16]	2022	A proposed approach for diabetes diagnosis using neuro-fuzzy technique	ANFIS	Acc:92.77% Sen: 97.00%
Image dataset for classification DFU				
[18]	2020	DFU_QUTNet: diabetic foot ulcer classification using novel deep convolutional neural network	DFU QUTNet, DFU QUTNet+SVM, DFU QUTNet+KNN	QUTNet+SVM, DFU Pre:95.4% Rec:93.6%
[20]	2020	An Integrated Design for Classification and Localization of Diabetic Foot Ulcer based on CNN and YOLOv2-DFU Models	CNN With dense neural network classifier, KNN, or DT	CNN With dense neural network classifier Acc: 97.00%
[22]	2020	Recognition of ischaemia and infection in diabetic foot ulcers: Dataset and techniques	CNNs+ SVM	Acc:90.30% Pre: 91.8%
[17]	2021	Comparison of hybrid convolutional neural networks models for diabetic foot	DCNN With 5,4,3,2 branches	DCNN with 4 branches Pre: 97.3% Rec: 94.50%

		ulcer classification		
[19]	2021	Exploiting Machine Learning Algorithms to Diagnose Foot Ulcers in Diabetic Patients	ELM model, KNN, ANN, and SVM	ELM model 96.15%
[21]	2021	DFU_SPNet: A stacked parallel convolution layers-based CNN to improve Diabetic Foot Ulcer classification	CNN architecture called DFU_SPNet	Acc:96.40% Pre: 92.40% Rec:98.40%
Image dataset for classification DR				
[24]	2018	Classifying diabetic retinopathy and identifying interpretable biomedical features using machine learning algorithms	SVM, DT, ANN, and LR	SVM Acc:83.90%
[28]	2020	Computer-aided diabetic retinopathy diagnostic model using optimal thresholding merged with neural network	MLU-DA-NN	Acc:92.0% Pre: 70.0% Rec:
ECG dataset for classification DM				
[26]	2012	Automated diagnosis of	FFNN	Acc:93.08% Spec: 96.92%

		diabetes using heart rate variability signals		
[25]	2018	Diabetes detection using deep learning algorithms	CNN-LSTM with SVM	Acc: 95.70%
[27]	2020	Heart rate variability features from nonlinear cardiac dynamics in identification of diabetes using artificial neural network and support vector machine	ANN, SVM	ANN Acc: 90.50%

1.7 Dissertation Outline

The dissertation's remaining sections are organized as follows:

Chapter Two: a thorough explanation of the fundamental concepts and techniques of deep learning and machine learning in terms of the classification process and decision optimization of diabetes classification. which will be referenced further in this dissertation.

Chapter Three: explains the stages of building the proposed model for diabetes classification and its characteristics.

Chapter Four: presents and discusses the experimental results obtained from applying the proposed model to a data set of different data types.

Chapter Five: shows the conclusions reached in this dissertation and offers suggestions for future works.

CHAPTER TWO:

Theoretical Background

Theoretical Background

2.1 Introduction

This chapter presents an introduction to diabetes mellitus and its complications and the dissertation's theoretical methodology, including techniques of preprocessing data for different data types (tabular, ECG, image) that include feature scaling, feature transform, data augmentation, dimensionality reduction, and image enhancement. Additionally, discuss the approaches of feature selection. After that, an introduction about artificial neural networks and activation functions. An introduction to Deep learning techniques and convolution neural network for classifying diabetes and its complications. Then mention how tunning hyperparameters includes (tuning the architecture of neural networks, optimization and learning, regularization techniques, and genetic algorithms). Introduction about unsupervised learning (winner-take-all and autoencoder). Finally, evaluation measures for classification diabetes and its implications are offered.

2.2 Diabetes Mellitus

Diabetic Mellitus (DM) is one of the most common diseases that occurs when the production of insulin in the body of a human is or when the body human is not able to utilize the produced insulin appropriately, as a result, this leads to high blood sugar. The number of people with diabetes increased from 108 million in 1980 to 451 million in 2017 according to the international diabetes Federation[29]. DM is a long-term problem with several risk factors, complexities, and rising prevalence. The danger that a person with diabetes will pose if it is not diagnosed early leads to major complications such as blindness, heart attack, lower limb amputation sometimes even death. Therefore, early diagnosis helps control the disease and choose the best treatments to avoid complications [14],[30]. Rate of

Heart Variability (HRV) is important indicator of diabetes, which plays an important role for diagnoses this disease.

Regardless of whether dyslipidemia or arterial hypertension may be present, diabetes-related hyperglycemia leads to alterations in the cardiovascular system. Diabetes results in cardiovascular autonomic neuropathy (CAN), which fully disorganizes the neural system and reduces HRV. Consequently, it is a sign to detect the presence of diabetic neuropathy. The time interval between two contiguous, successive QRS complexes in an ECG is known as the heart rate. HRV is a representation of RR-interval (e.g., the time elapsed between two successive R-waves of the QRS signal on the electrocardiogram) fluctuation. The fact that HRV tests are non-invasive. Thus, as a result of the spread of information technology and data science, which in turn introduced many effective methods for the classification of diabetes[31], [32].

The high levels of blood glucose can sorely harm parts of the human body, including the feet and the eyes. These are called “diabetes complications”. But can take activity to prevent or delay numerous of these issues. There are two types of complications of diabetes, The first type, which is a dangerous complication that constructs up over time called "chronic complications”, such as eye harm (diabetic retinopathy), is a medical case that harms the retina of the eye which causes visual destruction and foot harm (diabetic foot ulcer), it is the most complication of diabetes, which, if not appropriately treated, may lead to removal the feet, etc. The second type is called "intense complications" which can occur at any time and may cause “chronic complications”, such as Hypos and Hypers. The first occurs when the blood glucose is very low and the second occurs when the blood glucose is very high [33], [18].

A Diabetic Foot Ulcer (DFU) is considered one of the most common complications of diabetes. It can be described as a skin slough accompanied by complete loss of foot skin, often resulting from complications of neuropathic and/or vascular issues in patients with diabetes type 1 or 2. It has been found that occurs in (2 to 6%) of diabetic patients and influences as much as (34%) of them throughout their lifetime[34]. In the United States, foot ulcers are considered a critical well-being issue influencing 5 to 6 million people who suffer from type 2 diabetes. Furthermore, in 2010, approximately (73000) lower limbs were amputated due to wounds that are in some way or other related to diabetes. According to the research in[35], more than eighty-five percent of foot amputations are related to diabetic foot ulcers. Due to the spread of information and communication technology, many methods that are cost-effective for the remote detection and prevention of DFU have been introduced. One of these methods includes the utilization of automatic intelligent telemedicine frameworks, which besides the accessible healthcare-services, can give a high quality and low-cost DFU treatment [36][37].

Diabetic retinopathy (DR) is a common and potentially sight-threatening complication of diabetes mellitus. It affects the blood vessels in the retina and the light-sensitive tissue at the back of the eye. Early detection and accurate diagnosis of DR are crucial for timely treatment and prevention of vision loss. In recent years, machine learning and deep learning techniques have emerged as powerful tools for the automated detection and classification of DR, aiding in improving diagnostic accuracy, efficiency, and accessibility[24].

2.3 Electrocardiogram (ECG)

The electrocardiogram (ECG) is a physiological recording that represents the electrical activity generated by the heart [38]. The electrical impulse induces contraction and relaxation of the cardiac muscles, resulting in the expulsion of blood from the heart throughout a typical cardiac cycle. It is signal exhibits quasi-periodic behavior, with its frequency varying over time. This characteristic makes it an important instrument for various non-invasive biomedical applications, including heart rate estimation, monitoring heart rhythm, emotion recognition, biometric identification, and diagnosis of diabetes mellitus or cardiac abnormalities[39].

ECG signal pattern of individual participants may exhibit variations over time, body posture, and different physical contexts. The waveform has been visually shown in Figure 2.1 which composes of several components, including the P wave, PQ-segment, QRS complex with a distinct R-peak, ST-segment, and T wave. The given data is interpreted in terms of voltage amplitudes measured in millivolts (mV) and time intervals measured in milliseconds (msec). The process of atrial depolarization leads to the formation of a P wave, which is then followed by an isoelectric line that represents the AV delay, also known as the PQ-segment. The QRS complex is generated by ventricular depolarization, which can be hindered by delays or blockages in electrical impulses, such as bundle branch block. The ST-segment is an isoelectric line that aligns with the PQ-segment, and finally, the T wave is generated by ventricular repolarization. The heart rate is observed to be higher during standing and sitting positions in comparison to the state of sleeping[40], [41].

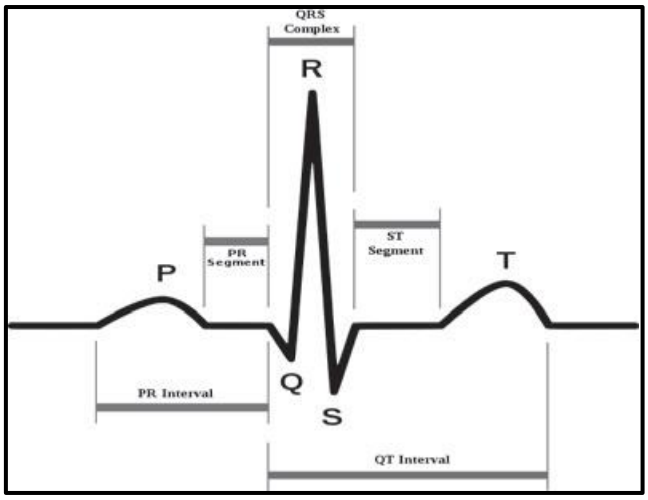


Figure 2.1 A schematic representation of a single heart cycle [40].

Research indicates that in around one-third of individuals with DM, there exists an initial phase of autonomic neuropathy that can be identified by an examination of heart rate variability (HRV) before the start of DM symptoms. The level of autonomic dysfunction linked to DM is contingent upon the severity and duration of the condition. The decrease in HRV characteristics appears to have both an adverse prognostic significance and occurs prior to the manifestation of cardiovascular autonomic neuropathy in clinical settings[42].

2.3.1 Heart Rate Variability (HRV)

The temporal interval between consecutive heartbeats is not uniform but rather exhibits variability from one heartbeat to the next. The term used to describe this occurrence is HRV. It is associated with the mean of heart rate (HR), indicating that it represents the consecutive changes in heartbeats, specifically examining the differences in HR from beat to beat [42]. In an alternative aspect, it is the variation of the time interval between prominent R-peaks. Additional terminologies that are employed in this context encompass "variability in cycle length," "variability in R-R intervals" (where R denotes the peak of the QRS complex in the ECG waveform, and RR represents the duration between consecutive Rs), and "variability in heart period."[40]. Figure 2.2 shows the RR interval.

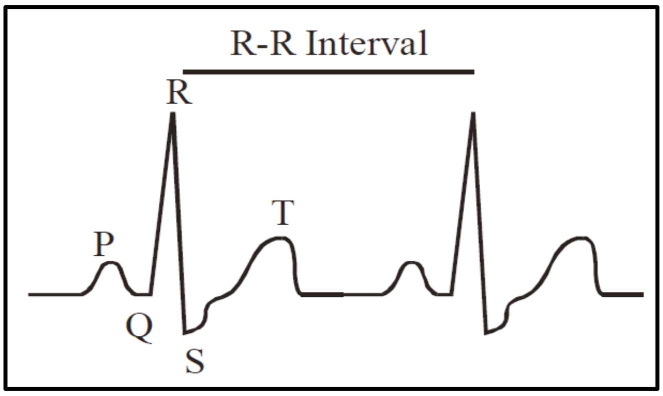


Figure 2.2 Describe the R-R Interval [41].

The examination of HRV in both the time domain and frequency domain is considered to be more straightforward and accessible. Significantly, the utilization of time domain and frequency domain analysis of RR interval enables the timely identification of autonomic dysfunction, as it has the capability to detect alterations prior to the manifestation of clinical symptoms, hence facilitating prompt intervention. This dissertation will examine time domain metrics including statistical and geometrical methods as follows:[43], [44]

A- Statistical Methods:

- **Mean RR Interval (MeanNN):** It is the mean of interval in seconds. Equation 2.1 explains the MeanNN formula.

$$\text{MeanNN} = \frac{\sum_{i=1}^{N} NN_i}{N} \qquad 2.1$$

Where NN_i is the duration of each RR interval, and N is the total number of RR intervals.

- **Standard Deviation of NN intervals (SDNN)**: The standard deviation of all the RR intervals, measuring the overall HRV. Equation 2.2 explains the SDNN formula.

$$\text{SDNN} = \sqrt{\frac{\sum_{i=1}^{N}(NN_i - \text{MeanNN})^2}{N-1}} \qquad 2.2$$

- **Root Mean Square of Successive Differences (RMSSD):** Particularly sensitive to parasympathetic (vagal) activity, it reflects the square root of the mean of the squared differences between adjacent NN intervals. Equation 2.3 explains the RMSSD formula.

$$\text{RMSSD}= \sqrt{\frac{\sum_{i=1}^{N}(NN_i - NN_{i+1})^2}{N-1}} \qquad 2.3$$

- **Percentage of successive RR intervals differing by more than 50 ms (pNN50):** A measure of short-term HRV, it represents the percentage of successive RR intervals that differ by more than 50 milliseconds. Equation 2.4 explains the SDNN formula.

$$\text{pNN50}= \frac{count\ (NN_i - NN_{i+1}) > 50\ ms}{N \times 100} \qquad 2.4$$

- **Count of successive RR intervals differing by more than 50 ms (NN50_count):** It represents the Number of adjacent NN intervals differing more than 50 milliseconds. Equation 2.5 explains the SDNN formula.

$$\text{NN50\_count}= count\ (NN_i - NN_{i+1}) > 50\ ms \qquad 2.5$$

- **Mean of Heart Rate (MeanHR**): It computes the mean of Heart Rate (HR). HR varies across time in a healthy heart. Equation 2.6 explains the HR formula. Equation 2.7 explains the MeanHR formula.

$$\text{HR}= \frac{60 \times fs}{\text{MeanNN}} \qquad 2.6$$

$$\text{MeanHR} = mean\ (HR) \tag{2.7}$$

- **Standard Deviation of Heart Rate (SDHR):** It computes the standard deviation of HR. Equation 2.8 explains the SDHR formula.

$$\text{SDHR} = SD(HR) \tag{2.8}$$

B- Geometrical Methods:

- **HRV Trianglar Index (HRVI):** It is determined from a histogram of RR intervals, where it is derived from the mode, denoted as X, which represents the most often occurring value with its absolute frequency Y. Figure 2.3 shows the measurement of HRVI. Equation 2.9 explains the HRVI formula.

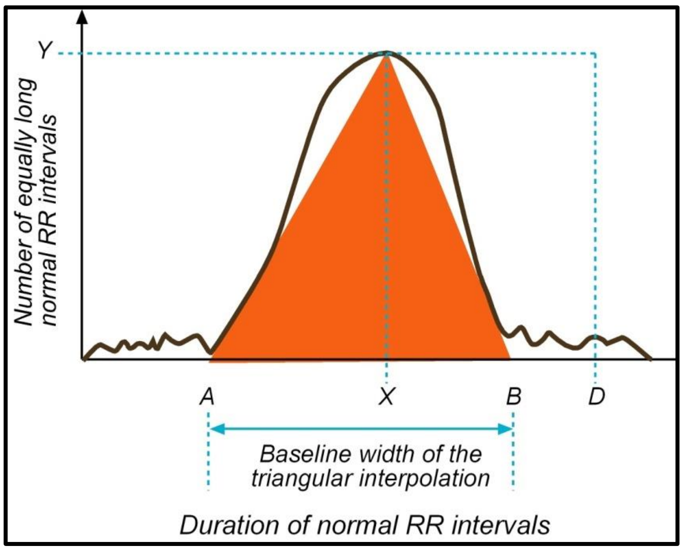

Y
Number of equally long normal RR intervals
A
X
B
D
Baseline width of the triangular interpolation
Duration of normal RR intervals

Figure 2.3 HRV triangular index (HRVI) measurement [43]

$$\text{HRVI} = \text{D/Y} \tag{2.9}$$

Where D is the sample density distribution which equals to the total number of all NN intervals. Y is the maximum sample density distribution (D).

- **Triangular Interpolation of NN Interval (TINN):** The geometric pattern is inferred through the process of interpolation using a mathematically determined shape. The process of approximating the distribution histogram by the use of a triangular shape. The

TINN (triangular interpolation of the discrete distribution of NN intervals) method employs triangular interpolation to estimate the distribution of RR intervals based on histogram counts. Equation 2.10 explains the TINN formula.

$$\text{TINN} = \text{B} - \text{A} \qquad 2.10$$

Where The values A and B are determined along a time axis. In addition, a multilinear function is also present. The function q is defined in such a way that q(t) equals zero for t less than or equal to A and t greater than or equal to B, and q(X) equals Y.

2.4 Data Preprocessing Techniques

It is important for improving the quality of data. Data preprocessing approaches can be classified into two categories: The first concerns transforming or changing data. The second deals with cleaning up redundant data and outliers.

2.4.1 Features Scaling

Often the datasets include features with vary units and ranges. To accurately and efficiently model the data, most ML techniques must have the data standardized or normalized to a consistent scale. Min-max (normalization) and Z-score (standardization) are two of the most often utilized techniques for scaling in the preprocessing stage. These methods are utilized to enhance the efficiency of ML techniques, specifically in scenarios where features exhibit varying units and ranges[45].

Min-max technique converts the data in a particular range between 0 and 1. Equation 2.11 explains the formula of normalization.

$$\text{Norm}(x) = \frac{\text{v-}\min(x)}{\max(x) - \min(x)} \qquad 2.11$$

Where x is the original value, the distribution's minimum and maximum values are min (x) and max (x), respectively.

On the other hand, with Z-score, the data are altered to approximate a standard distribution. Equation 2.12 explains the formula of standardization and Equations 2.13 and 2.14 explain the mean ($\bar{x}$)and Standard Deviation (SD) formulas:

$$\text{Z-score } (x_i) = \frac{x_i - \bar{x}}{\text{SD}}, i = 1,2 \ldots, n \qquad 2.12$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{\text{n}} \qquad 2.13$$

$$\text{SD} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{\text{n}}} \qquad 2.14$$

Where x_i is the original value and $\bar{x}$ is the mean.

2.4.2 Outlier Detection and Removal

In ML, outliers can be defined as data objects that possess characteristics that deviate from the majority of other data objects within a given dataset[46]. In numerous data analysis operations, a significant number of variables are captured or sampled. The initial stage in achieving a cohesive analysis involves identifying anomalous observations. While outliers are commonly regarded as errors or noise, they might potentially contain significant information. Hence, it is imperative to determine their identification before the beginning of building and analysis of the model[47].

In this section, the method that is used to detect the outliers is called the Z-Score or Standard Deviation Method (SDM). It is a statistical method to determine the outliers. In this method, the process entails the computation of the Z-score for every individual data point. The Z-score is a statistical measure that measures the number of standard deviations by which a given data point deviates from the mean. Outliers are defined as data points that have the Z-score exceeding a specified threshold, often Z-score > ± 3[47].

2.4.3 Data Augmentation

Data augmentation (DA) is a widely employed technique in the fields of ML, DL, and computer vision (CV), which aims to artificially enhance the diversity and magnitude of a dataset by implementing a range of modifications (e.g., transformations) on the original data. The investigation of DA has also been examined in the context of oversampling applications. Oversampling is a method employed to address the issue of uneven class distributions by re-sampling the data in a manner that reduces the potential bias of the model towards identifying cases belonging to the majority class[48]. Increased variability of data that allows procedure aids in the generalization of ML/DL models and reduces overfitting, allowing them to operate better on new or complicated data[49].

2.4.3.1 Image Data Augmentation

Lecun et al. [50] proposed LeNet-5 which contains a data warping technique for image data augmentation. One of the first applications of CNNs for classifying handwritten digits was this one. To achieve optimal performance, a CNN requires a significant amount of labeled training data. Because the CNN parameters aren't adjusted enough, a small set of training will produce overfitting [48]. Additionally, it is quite expensive to get a significant number of medical-images, so this method will be utilized to address the issue[51].

To create a new form of the images while maintaining the underlying meaning behind it, image augmentation requires performing a variety of transformations to the incoming images. Typical enhancement methods include[48], [49]:

- **Rotation** is the process of altering the orientation of images by applying different angles, hence introducing differences in the positioning of objects.
- **Flipped**: The process of flipping images involves creating mirror images by either horizontally or vertically reversing the original image, thereby imitating alternate viewpoints.
- **Shearing** is a commonly employed technique aimed at improving image quality, enabling computers to accurately perceive images from different angles.
- **Shifting:** The process of shifting an image involves shifting it in various directions, such as left or right, bottom or top.
- **Cropping**: Extraction of image portions to isolate particular features.
- **Zooming:** It is used to resize the image to a small size (e.g., reduce the dimension of the image). By decreasing the dimensions of the image, the model exhibits decreased sensitivity towards complex details and instead exhibits higher-level characteristics.

Figure 2.4 shows some data augmentation methods, that depend on transformation techniques.

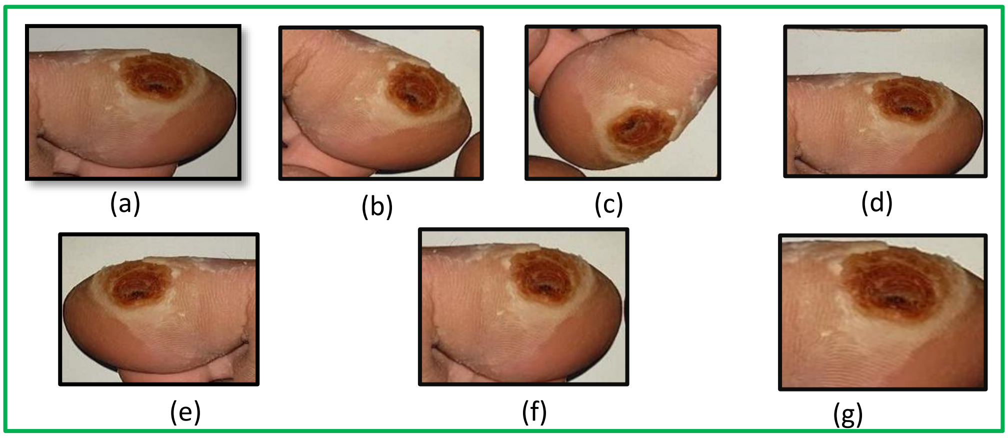


Figure 2.4 Data augmentation operations of DFU sample [18] (a) Original image (b) Rotate with 45 degree (c) Rotate with 135 degree (d) Shift Range with value equal 0.2 (e) Flipping Horizontal (f) Zooming shift (g) Cropping

2.4.4 Spectrogram Analysis Method

A spectrogram is a visual representation of the frequency characteristics of a signal as it changes over time. The 2D representation displays time on the x-axis, frequency on the y-axis, and the intensity or magnitude of the frequencies is represented by color. A spectrogram can be applied in several fields including signal analysis, speed recognition, and audio analysis[52]. Creating the spectrogram is required several steps. Firstly, windowing process the signal is partitioned into small overlapping pieces by employing a window function, such as the Hamming or Hanning window [53]. Secondly, the Fast Fourier Transform (FFT) is used to convert each segment into the frequency domain, allowing for analysis of the frequency components present within that specific time window. Thirdly, the Power Spectral Density (PSD) is calculated by squaring the magnitude of the Fast Fourier Transform (FFT) at each frequency bin. Finally, the spectrogram is constructed by organizing the Power Spectral Density (PSD) values for each time frame into a matrix. In this matrix, the x-axis represents time, the y-axis represents frequency, and the color indicates the magnitude of the frequencies. Typically, brighter colors are indicative of greater amplitude or energy in the respective frequency ranges[54].

2.5 Dimensionality reduction

The issues that come with dealing with high-dimensional datasets are known as the "curse of dimensionality," which is addressed by the core ML and data analysis technique known as "dimensionality reduction". In general, applications in the real-world use datasets with a lot of features, which can make computations more complex, reduce the performance of models, and make visualization and interpretation more challenging. The goal of dimensionality reduction techniques is to decrease feature numbers while maintaining as much useful data as feasible [46], [47]. In this dissertation, it has been used Singular Value Decomposition (SVD) approach for reducing the dimensionality.

2.5.1 Singular Value Decomposition (SVD)

The techniques to minimize the number of input variables within the training data are indicated as dimensionality reductions, such as the SVD technique. Poor performance might be caused by a huge number of input features. SVD allows for an accurate form of any matrix (M), as well as the removal of less important components. Equation 2.15 shows the SVD formula. It is applied by employing the following formula:[55].

$$\mathrm{M} = U \bullet \Sigma \bullet V\text{^}T \qquad 2.15$$

Where U refers to a column-orthonormal matrix, Σ refers to a diagonal matrix, V refers to a column-orthonormal matrix and V^T refer transposed form of V.

The reduction process can be achieved by applying the SVD method as follow:[56]

1- Let be considered a matrix M with dimensions m x n, where (m) represents the number of rows and (n) represents the number of columns.

2- In certain instances, it may be necessary to perform a matrix transposition (M^T) on matrix M in order to achieve it's of the desired form.

3- Matrix multiplication involves multiplying matrix M by its transpose, denoted as M^T, or vice versa. This operation yields the product matrix M^T•M. The matrix obtained from this process exhibits symmetry.

4- Perform eigenvalue decomposition on the symmetric matrix acquired in the previous step. This involves finding the eigenvalues and eigenvectors associated with the matrix.

5- Construct matrices U, Σ, and V. The matrices U and V represent the eigenvectors, whereas Σ denotes a diagonal matrix containing the square root of the eigenvalues.

6- Reduction of rank: In the case of a matrix with a high rank, it is possible to decrease its rank by selecting the highest k singular values in the diagonal matrix Σ, as well as the corresponding columns in U and V.

2.6 Image Enhancement

Image enhancement includes capturing an image and enhancing it visually, frequently by utilizing responses from the human visual system. The techniques of image enhancement are utilized for presentation and analysis, by emphasizing image features, making it sharper, and /or smoother. The practice of applying these techniques to images to improve them makes it easier to develop a solution to a computer vision issue. Enhancing an image's contrast is among the easiest enhancement methods[57]. In the next subsection, explain histogram Equalization and its evolution.

2.6.1 Contrast limited adaptive histogram Equalization (CLAHE)

Histogram equalization is a common method for enhancing the visibility and quality of an image[57]. This is accomplished by extending the dynamic-range of an image's histogram. Briefly, the Histogram equalization converts the input image's gray levels into the produced image's uniform gray levels (e.g., the gray level distribution of an output image is uniformed). It improves the contrast of the image by stretching the input image's histogram's dynamic range. Figure 2.5 (a) shows the default histogram and Figure 2.5 (b) shows the flattened histogram. A deeper understanding of Histogram equalization can be attained by examining its mathematical framework as follows[58]:

- The computation of the image's histogram is performed, the frequency at which each pixel intensity level occurs as explained in Equation 2.16. Let N refer to the number of pixels, and G represents the number of gray levels in an image in the range between (0 and 255).

$$P(g)=\frac{n_g}{N} \quad \text{for } g=0,1,\ldots,G-1 \qquad 2.16$$

Let $\mathbf{n_g}$ represent the total number of pixels associated with the gray-level **g** in an image.

- The calculation of the cumulative distribution function (CDF) is derived from the probabilities obtained from the histogram. It is the sum of the $P(n_g)$. Equation 2.17 explains the CDF formula.

$$CDF(g)=\sum_{g=0}^{G-1} P(g) \quad \text{for } g=0,1,\ldots,G-1 \qquad 2.17$$

- Transform intensity level (normalization): To cover the complete intensity range (0 to G-1), the CDF is normalized. Equation 2.18 explains normalization of CDF formula.

$$\bar{C}(g)=\frac{C(g)-\min\big(C(g)\big)}{n-1}\times(G-1) \qquad 2.18$$

Where min(C(g)) is the smallest value of C(g), but it is non-zero.

- Replace the original with a new value produced from the previous

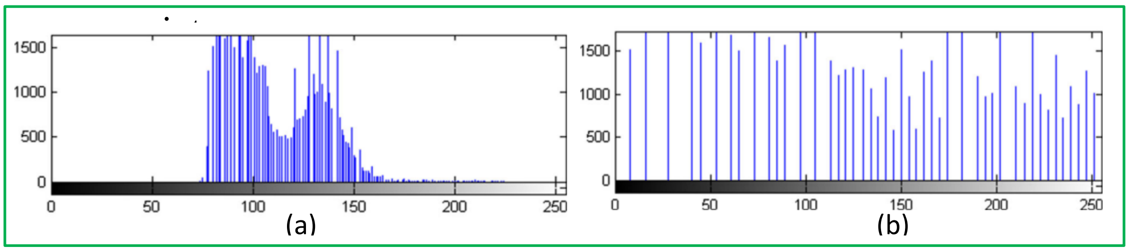


Figure 2.5 histogram equalization (a) histogram of original gray image (b) histogram of output image after applies histogram equalization method [59].

The results of histogram equalization may be inflated by noise and distortions. So, in this section, more efficient approach is used, called Contrast-Limited Adaptive Histogram Equalization (CLAHE). The drawbacks of conventional histogram equalization are addressed by CLAHE, an enhanced variation of histogram equalization and prevent artifacts and noise amplification, particularly when working with images with variable local contrast.

CLAHE method can be explained in several steps. First, separates an original image into non-overlapping blocks known as tiles with specified size (e.g., 8×8 pixels). Second, applies histogram equalization on each tile independently by adjust the pixel intensities in each tile based on the CDF. Third, clipping pixel intensities with specified clip limit (L with value 3 or 4). By determining the value of contrast limit (L) that depends on the contrast-enhanced level, the values that produces from the histogram equalization of each tile must be less of equal than L value. Finally, applies bilinear interpolation to avoid artificial boundaries between tiles and interpolates the pixel values of neighboring tiles. Figure 2.6 shows the steps of CLAHE method.

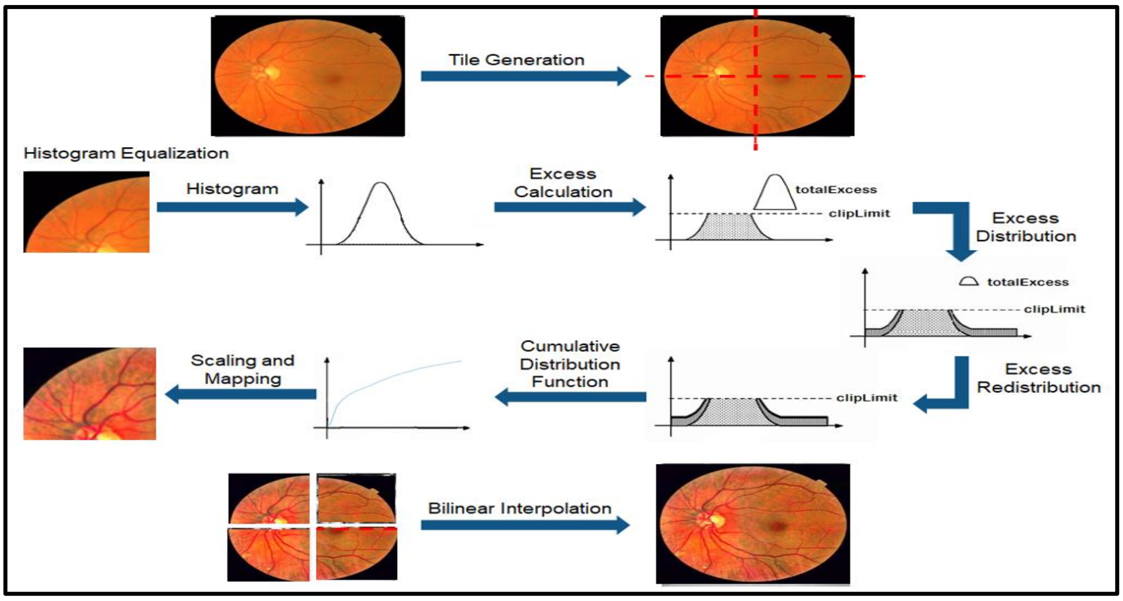


Figure 2.6 The main steps of CLAHE method including tile generation, histogram equalization, and bilinear interpolation [60]

2.7 Texture and Textural Features Analysis

The texture is a simple idea that describes characteristics like a region's coarseness, smoothness, and regularity. It is a crucial component of human vision and gives hints about the depth and orientation of a scene. Another fundamental quality of objects is texture. It offers crucial details on how surfaces are arranged structurally. Depending on the spatial resolution of the sensor and the size of the homogenous region being classified, it may be assumed that using texture in addition to spectral information will increase accuracy to some extent [61].

Two important concepts need to be clarified before getting into texture analysis techniques. First, first-order statistics calculate the probability of viewing a grey value in a randomly selected area of the image. It can be calculated using the image's histogram of pixel intensities. These are independent of neighboring pixel interactions or co-occurrences and only rely on the individual pixel values. Second, second-order statistics are described as the probability of observing a pair of grey values appearing at the ends of a dipole (or needle) that has been randomly positioned and oriented inside the image. These are characteristics of pixel value pairings. Based on second-order statistics the texture features are extracted. The statistical distribution of observed combinations of intensities at specific points relative to one another in the image is used to construct texture characteristics in statistical texture analysis [62], [63].

- **Gray-Level Co-Occurrence Matrix Technique (GLCM)**

A commonly adopted texture analysis approach is the gray level co-occurrence matrix, which was initially introduced by Haralick Image features related to second-order statistics are extracted (e.g., a feature of GLCM utilizes the particular relation between two pixels in an image that are separated by a specific distance) This occurs in micro-texture regions whenever the primitive element size is large (e.g., rapidly changing value), or in macro texture regions in case the primitive element size is small (e.g., slowly changing value) [64]. A GLCM is a matrix (Gl), whereby the number of its rows and columns are similar to the number of gray-level ranging from (0, 255). The number of occurrences of the pair of intensity levels i and j, at a distance apart and in the direction in the gray image, is equivalent to each entry (i, j)th in Gl. Adjacency can be defined as a relationship that exists in each of the four directions (horizontal, vertical, left, and right diagonal) and at any distance. Figure 2.7 shows four directions of adjacency. Figure 2.8 (a) represents a (5×5) image with three grey levels, and Figure 2.3 (b) shows the corresponding GLCM's general form. For instance, the value in cells (2, 3) shows how frequently gray levels 2 and 3 occur with a particular direction (θ)and distance (d). Figure 2.3 (c) to (f) refer to the outcome for the four above-mentioned directions for d = 1 with different angles including (00, 900, 450, 1350) respectively.

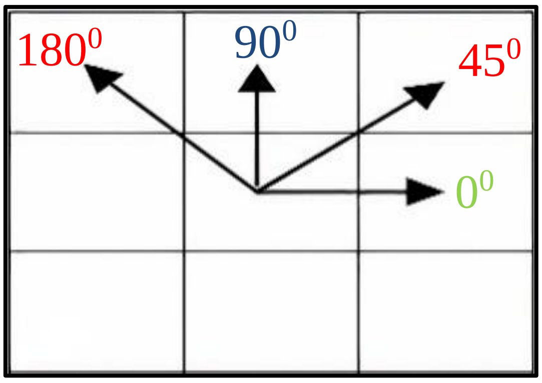
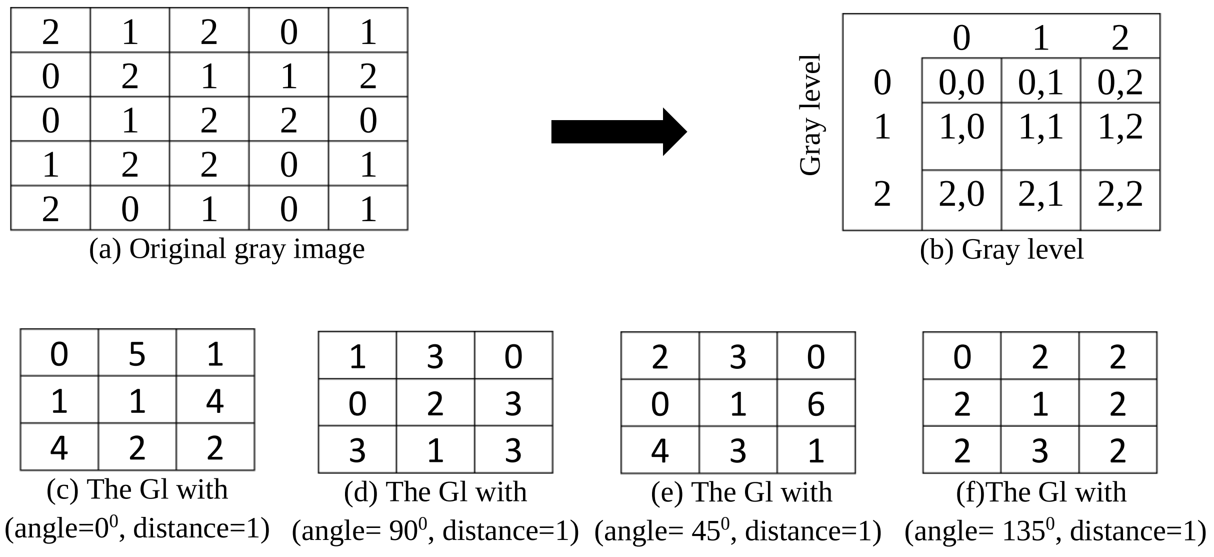


Figure 2.7 GLCM adjacency direction[51]



2	1	2	0	1
0	2	1	1	2
0	1	2	2	0
1	2	2	0	1
2	0	1	0	1

(a) Original gray image

Gray level	0	1	2
0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

(b) Gray level

0	5	1
1	1	4
4	2	2

(c) The Gl with (angle=0^0, distance=1)

1	3	0
0	2	3
3	1	3

(d) The Gl with (angle= 90^0, distance=1)

2	3	0
0	1	6
4	3	1

(e) The Gl with (angle= 45^0, distance=1)

0	2	2
2	1	2
2	3	2

(f)The Gl with (angle= 135^0, distance=1)

Figure 2.8 Example of GLCM method with max-gray level= 2, distance = 1, and different angles include (00, 900, 450, and 1350)

After that the matrix (Gl) is completed, in the next step, a set of statistical measures is calculated, as follows: [65][66].

- **Entropy**: It indicates unrest in any system, and with texture analysis as shown in Equation 2.19, it measures the spatial unrest. It could be calculated as:

$$\text{Entropty}= -\sum_{p=0}^{N-1}\sum_{k=0}^{N-1} \text{Gl}(p,k) \log \text{Gl}(p,k) \qquad 2.19$$

- **Contrast**: It refers to the gray-level diversity as explained in Equation 2.20. It can be calculated as:

$$\text{Contrast}=\sum_{p=0}^{N-1}\sum_{k=0}^{N-1} (p-k)^2\, \text{Gl}(p,k) \qquad 2.20$$

- **Energy**: Because energy measures local homogeneity, it is the polar opposite of Entropy as explained in Equation 2.21. This attribute indicates how consistent the texture is. It can be calculated as:

$$\text{Energy}=\sum_{p=0}^{N-1}\sum_{k=0}^{N-1} \text{Gl}(p,k)^2 \qquad 2.21$$

- **Dissimilarity**: It describes how different grey-level pairs in an original image vary as explained in Equation 2.22. It can be calculated as:

$$\text{Dissimilarity}=\sum_{p=0}^{N-1}\sum_{k=0}^{N-1}|p-k|\,Gl(p,k) \qquad 2.22$$

- **Homogeneity**: It refers to the consistency of the GLCM's non-zero entries as explained in Equation 2.23. It can be calculated as:

$$\text{Homogeneity}=\sum_{p=0}^{N-1}\sum_{k=0}^{N-1}\frac{1}{1-(p-k)^2}Gl(p,k) \qquad 2.23$$

- **Correlation**: It denotes the image's texture similarity in two different directions, particularly, in both the x and y direction directions as explained in Equation 2.24. It can be calculated as:

$$\text{Correlation}=\frac{\sum_{p=0}^{N-1}\sum_{k=0}^{N-1}(p-\mu_p)(k-\mu_k)Gl(p,k)}{\sigma_p\sigma_k} \qquad 2.24$$

Where N-1 represents maximum gray level. p represents row of Gl matrix and k represents column of Gl matrix.

2.8 Artificial Neural Network (ANN)

Artificial neural network systems are considered simplified mathematical models like human brain systems. It is different from traditional computers, which are manually programmed to implement particular tasks, while ANN must be trained (auto-programmed). The advantage of ANN applications, they can make models easy to use and the accuracy becomes high rather than a complex natural system. It is resistant to noise, missing data, and ambiguity. It can also learn from and adapt to its surroundings[67], [68].

ANN has become a more common and useful model for many fields such as classifications, regression, and clustering. Learning in ANNs is divided into two main categories including SL and USL. Concerning SL, the outputs in the neural network are pre-defined. After that, the weights are modified in a way that enables them to make the desired outputs somewhat similar to the estimated network outputs, for example, Forward Artificial Neural Network (FANN). While USL relies on the clustering of the data inputs. That means the knowledge of an input's membership in a specific class is not available. The network will help in establishing classes and potential borders between them by gradually discovering properties and a history of training. For example, Kohonen’s networks [69], [70].

There are several structures of ANN such as Single Layer Perceptron (SLP), Multiple Layer Perceptron (MLP), and Recurrent Neural Network (RNN). SLP is similar to a linear-regression[69]. It is useful for pattern classification into two or more classes as shown in Figure 2.9.

MLP is considered one of the most common supervised Artificial Neural Networks (e.g., using objects of data with known outputs) and is a strong modeling tool that efficiently learns complex systems. The architecture of MLP is changing, which means that consists of many layers of neurons connected in a feed-forward way as shown in Figure 2.10.

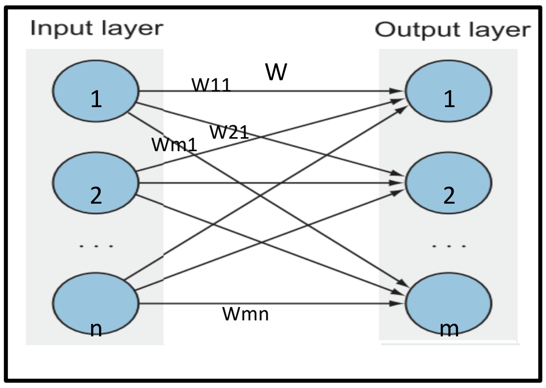


Figure 2.9 The SLP architecture contains an input layer and an output layer.[72]

This form is used to generate a non-linear function model that can the classification of data output that gives from input data[68] [71].

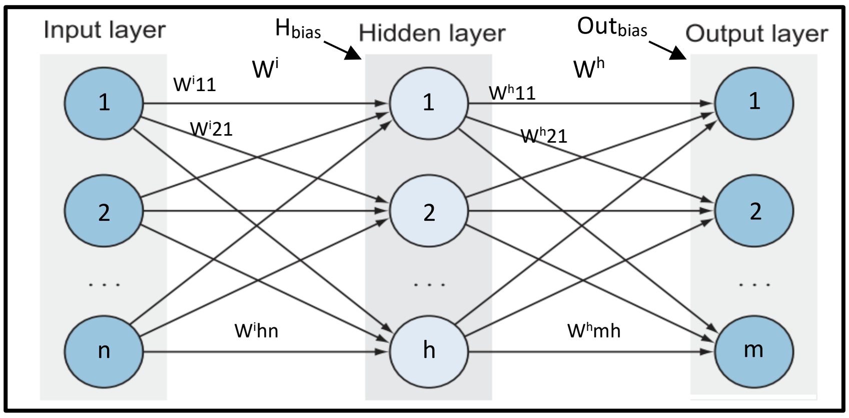


Figure 2.10 The MLP architecture contains an input layer, hidden layer and an output layer [72]

The architecture of MLP in Figure 2.10 consists of three main layers. The first, the input layer (1, 2, ..., n) units, receives the input vector of features to feed NN. The second, the hidden layer (1, 2, …, h) units, it is known as hidden, so the inputs cannot be controlled or seen. finally, output layer (1, 2, …, m) units. In addition, the NN has two vectors of biases H_{bias} and Out_{bias}. These vectors are connected to the hidden and output layer respectively. The W^i is the weight that is assigned between the connection of the input layer and the hidden layer units. W^h is the weight that is assigned between the connection of hidden layer units and output layer units. These weights reflect the importance and influence of units on the final classification in the output layer.

Training of MLP based on a supervised learning approach is called the BP algorithm. The error function is decreased and controlled by tunning parameters of the network (weights, biases). It once propagated error derivatives concerning each weight from the output layer back to the input layer to update weights[73].

2.9 Activation Functions

An activation function within the context of a neural network refers to a mathematical function that influences the output of a unit or neuron, taking into account its weighted inputs. The incorporation of non-linearity into the network facilitates its ability to acquire and reflect complicated connections within the data. In a neural network, every individual neuron performs the application of an activation function to the weighted sum of its inputs, subsequently transmitting the outcome to the subsequent layer of neurons [69]. In this section, the activation functions as follows:

- **Linear Activation function (Linear):** It is one of the most basic activation functions used in ANN, known as the identity activation function the inputs are the same as the outputs as shown in Figure 2.11. Equation 2.25 represents it as follows:

$$\text{Linear }(x)=\text{x} \qquad 2.25$$

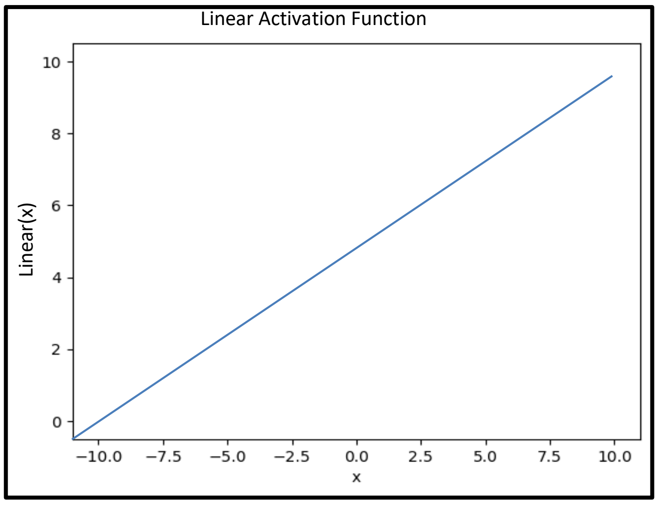


Figure 2.11 Linear activation function

- **Relu Activation Function (ReLU):** It is a nonlinear activation function that comes after each dense layer as presented in Figure 2.12. Equation 2.26. This function is performed by applying the max function, as follows: [74]

$$\text{ReLU (x) = Max (0,x)} \qquad 2.26$$

Where x is the input vector

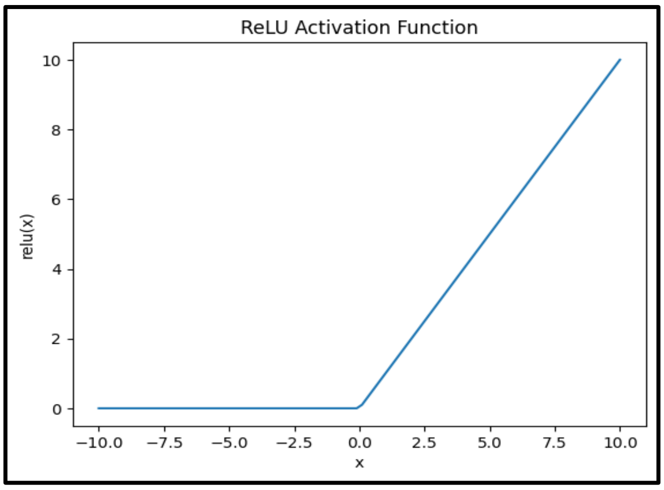


Figure 2.12 Relu function.

- **Sigmoid activation function**: It is a nonlinear-activation function that looks at the S-shape as presented in Figure 2.13. It exists between 0 and 1. It is used to predict the probability of binary classes. This function is applied as in Equation 2.27 :[75]

$$\text{Sigmoid}(x)=\frac{1}{1+e^{-x}} \qquad 2.27$$

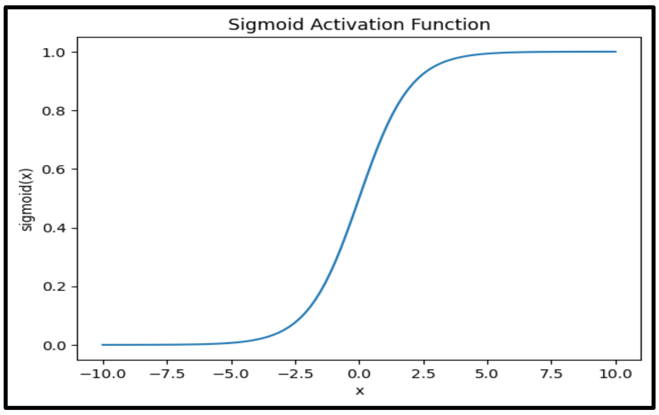


Figure 2.13 Sigmoid function

2.10 Deep Learning Techniques

Deep Learning (DL) is considered one of the most common ML techniques and can be regarded as a continuation or expansion of the neural network paradigm[6]. DL enables computer models, which are made up of several processing-layers, to learn data representations at various levels of abstraction. By using this technique, the state-of-the-art in many other fields, including image recognition, discovering drugs, the field of genomics, object identification, and speech recognition has been significantly enhanced. It uses the BP algorithm to change the parameters (weights, biases) in each layer of the neural network. It can be employed by a deep neural network (DNN)[76].

DNN consists of multiple layers, having two or more hidden layers. DL techniques currently outperform traditional ML techniques and become more scalable, and the high precision can be obtained by using a huge training dataset or increasing the size of neural networks [6].

There are several DL techniques which been suggested to deal with different problems such as Convolution Neural Network (CNN) to deal with image, video, or speech data, and Recurrent Neural Network that uses sequential data text or speech data [7]. In this section, convolution neural networks are considered.

2.10.1 Convolution Neural Network (CNN)

CNN is considered one of the foremost well-known Deep Neural Network (DNN) architecture ordinarily taught by a gradient-based optimization technique. CNN is not only a DNN that has numerous hidden layers, but it works similarly to the visual cortex of the brain that distinguishes or classifies images[6]. It differs from the traditional methods of machine learning in that the feature extraction stage is automatic instead of manual. It is designed for image recognition. CNNs function similarly to regular neural networks in that they include many layers that enable each layer to find increasingly complicated features. Some fundamental features such as edges and lines are learned by the first layer of convolutions. The following layer learns slightly more complicated features such as squares, circles, and so forth, the subsequent layer locates even more complicated features called abstract features [6],[77]. Currently, it is important to understand that the Convolutional Neural Network (CNN) architecture refers to the same structure as other neural networks, but the difference part is that the CNN uses convolution layers and pooling layers for learning features instead of fully connected layers. Neurons are arranged in hidden layers upon one another; during the training process, weights are initialized at random and are learned, and activation functions are applied and compute the error function. Finally, update the weights using the BP algorithm[6]. Figure 2.14 clarifies the architecture of CNN.

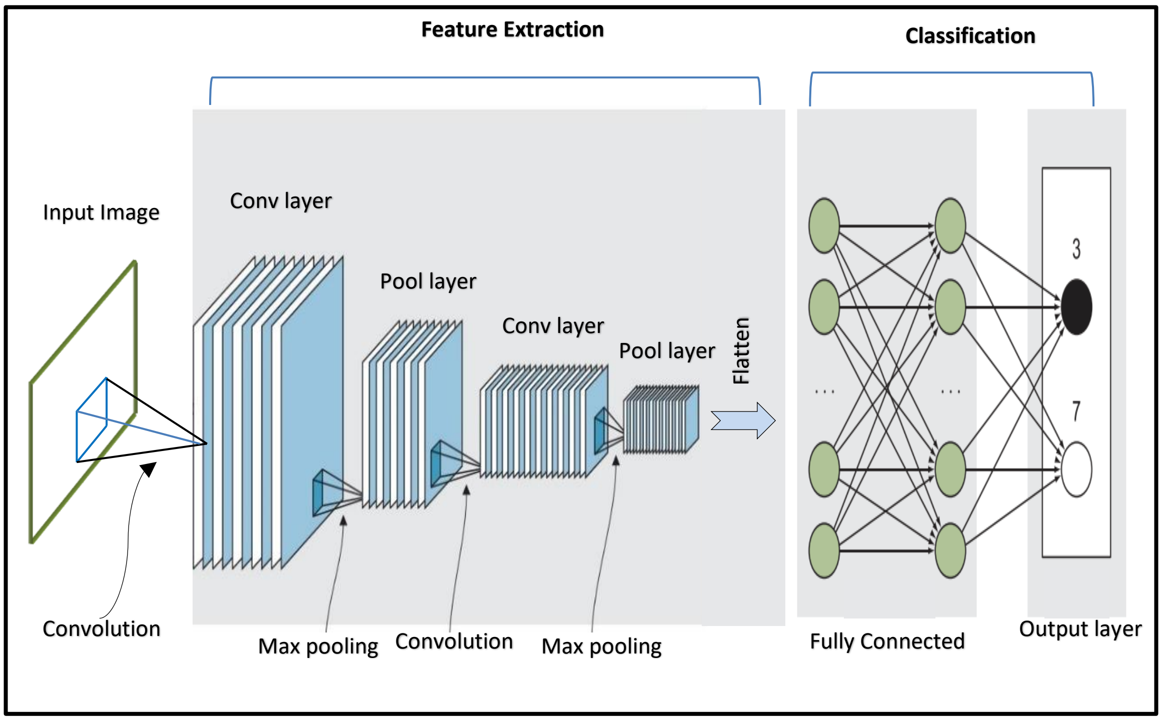


Figure 2.14 CNN architecture[72]

2.10.1.1 Basic Components of CNN

In practically every convolutional network, there are three primary types of layers; Convolution layer (Conv), Pooling layer (Pool), and Fully Connected layer (FC). As follows:

- **Convolution Layer:** It is the fundamental unit of a CNN. Conv functions as a feature seeker screen that moves pixel by pixel across the image to find useful characteristics that assist in recognizing the objects in the image. Convolution in mathematics is the process of combining two steps to get a new product called a feature map (e.g., a modified image with new pixel values). The first step is the input image, and the second step is applying the convolution filter. This filter slides over the input image to produce the feature map or activation map. Figure 2.15 explains the convolution process. The Conv layer has three important hyperparameters as follows:

- **Filter or Kernel**: It is a 2D matrix with size (3×3), (5×5), or (7×7). This matrix has weights that slide over an image for extracting features.
- **The stride** represents the amount of sliding of the filter over the image. Figure 2.16 explains the sliding filter over the image with a stride equal to (1).
- **Padding**: It is used to add zeros around an image to keep it on the border of it. Figure 3.17 shows how to add zeros to an image.

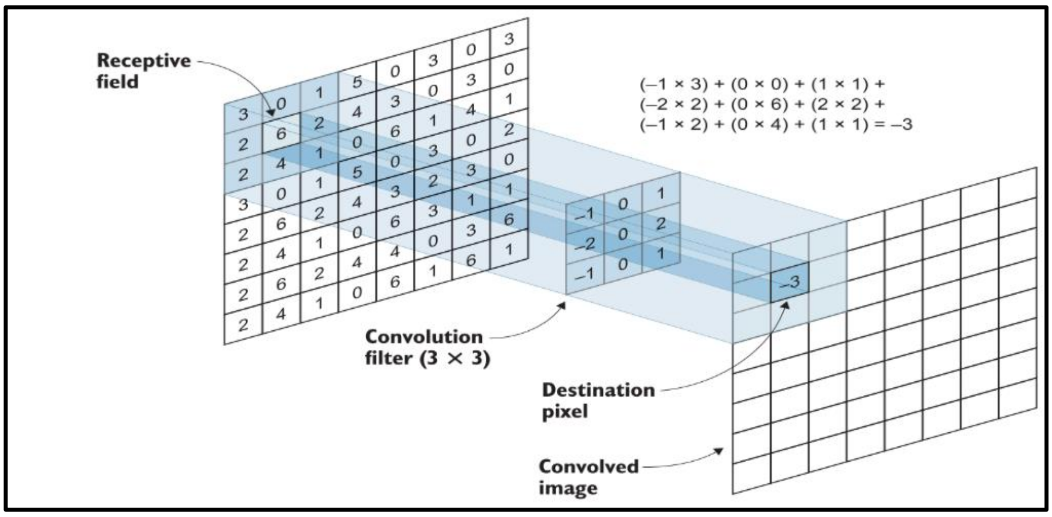


Figure 2.15 Convolution operation[72]

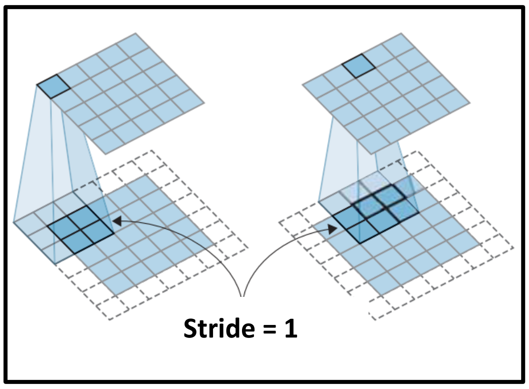
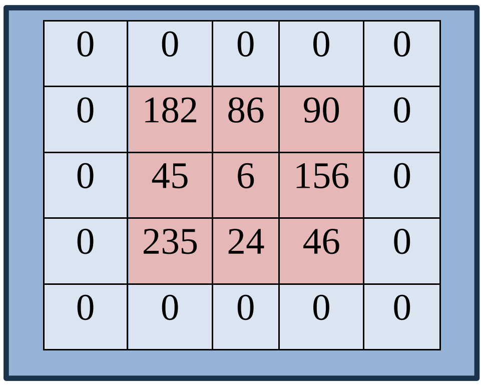


Figure 2.16 Sliding process [72]



0	0	0	0	0
0	182	86	90	0
0	45	6	156	0
0	235	24	46	0
0	0	0	0	0

Figure 2.17 Padding process[6]

- **Pooling or Subsample layer:** It is used to reduce the image size, by combining nearby pixels in a specific area of an image into a single value. The reason behind using this layer is that when a lot of convolutional layers are added to the network, this will increase the number of parameters (weights) in the network, thus increasing the time and space complexity. So, the objective of this layer is to reduce the size of the network by down-sampling the feature map, and thus the number of parameters is decreased by applying two statistical mechanisms, the Maximum and the average. Figure 2.18 clarifies the pooling process with maximum and average approaches.

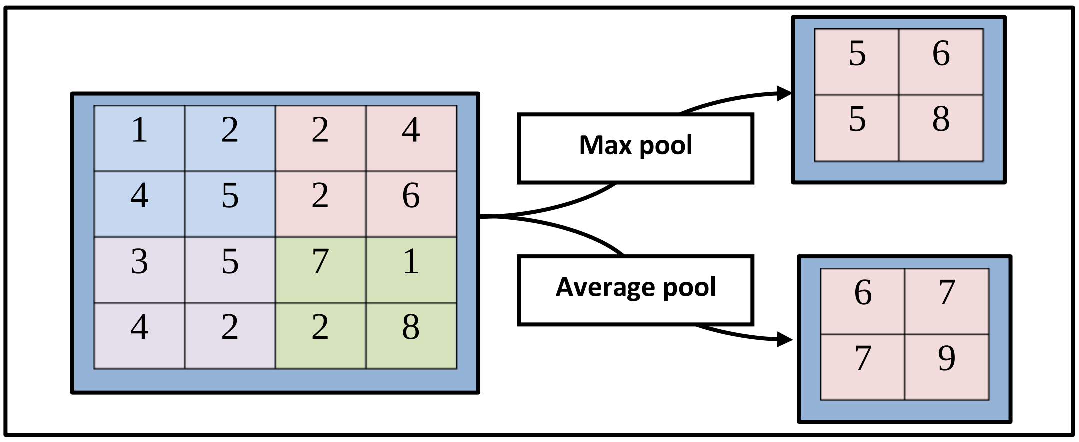


Figure 2.18 Pooling with two different approaches including max pool and average pool [6]

- **Fully Connected layer**: all the output units from a certain layer are linked to all input units of the next layer. It is used after extracting all the feature maps by running the image through Conv and Pool layers and putting them in a long tube called (flatten). After that, need to classify an image by using these features. Can be used in any classifier such as MLP of the DNN model[74].

2.11 Tunning of Hyperparameters

Hyperparameters are parameters that are assigned certain values before beginning the training process. Each problem should have its own set of hyperparameters. High classification accuracy can be attained with little to no hyperparameter adjustment. Some hyperparameters affect how much time and memory the algorithm needs to execute, while others affect how well the model predicts the future[78]. Unlike ML, the DL is full of hyperparameters. There are many challenges and ambiguities when tuning them (e.g., there are no universally applicable magic numbers). In this section, two techniques are used for tunning parameters and hyperparameters including trial-and-error and optimization method[72].

In general, three main groups can be divided the neural network hyperparameters as follows:

1) Architecture of Neural Network,
2) Optimization and Learning,
3) Regularization techniques,

And using a Genetic algorithm for tuning hyperparameters of the GLCM method.

This section provides a short but complete explanation of a standard set of hyperparameters and how they are tuned.

2.11.1 Tunning Architecture of Neural Network

The network's depth (means the number of hidden layers) and width (means the number of neurons in each layer) must be chosen, whether an MLP, DNN, CNN is created, or another type of neural network. The network's ability to learn is described by the number of hidden layers and units. A larger network may cause an overfitting problem, whereas a smaller network may cause an underfitting problem[79]. The objective is to choose a number that will allow the network to understand the features of the data. The model will need more learning ability to learn its features the more complex the dataset is[80]. Figure 2.19 shows a closer look at the three datasets. In general, adding hidden neurons is beneficial until the error of validation stops getting better. The GridSearchCV method is used to tune the number of hidden layers and their units.

GridSearchCV is a popular technique for fine-tuning hyperparameters in machine-learning models. Popular machine learning packages like scikit-learn in Python include GridSearchCV. It is a potent tool that automates the process of thoroughly scanning the whole parameter space to find the optimal set of hyperparameters for a particular model. The method of creating a grid of hyperparameter values and rigorously assessing the model's performance on each set of values contained in that grid is known as "GridSearch". Cross-validation, the "CV" in GridSearchCV, is a method for determining how well a model performs on unobserved data. The input for the GridSearchCV algorithm is a set of hyperparameters and their corresponding values. A grid of all conceivable combinations of these hyperparameters is then created. The program employs a cross-validation technique to train and assess the model for each combination. The user can choose from a variety of performance metrics, including accuracy, precision, recall, and F1 score, to assess the performance of the model.

The automation of the hyperparameter tuning process, which can be laborious and error-prone if done manually, is one of the main benefits of GridSearchCV. By methodically investigating every possible parameter combination, GridSearchCV makes sure that no set of hyperparameters is overlooked. Choosing the appropriate hyperparameters based on unbiased assessment aids in avoiding bias.

In addition, our neurons wouldn't solve any nonlinear issues if they weren't activated (e.g., don’t use activation function); instead, they would just transfer linear combinations (weighted sums) to one another. The effectiveness of a neural network is greatly influenced by the choice of an appropriate activation function. The network can learn complex correlations between inputs and outputs thanks to activation functions, which introduce non-linearities to the network[72].

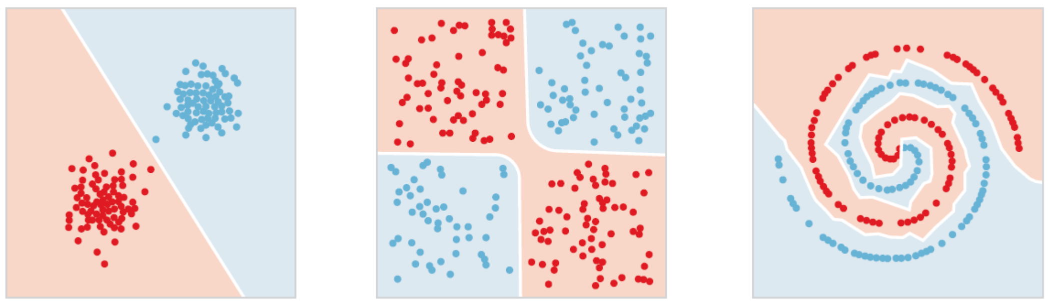


Figure 2.19 The capacity of the model to learn its features will increase with the complexity of the dataset[72].

During the training process by using gradient propagation, the neural networks change the parameters of the model (like weights and biases) through using the technique of optimization based on gradient optimization. The network's backpropagation of gradients is influenced by the activation function selected. An activation function should, in theory, enable effective gradient propagation while avoiding problems like disappearing or exploding gradients[81]. There are several types available of activation functions such as sigmoid, tanh, and ReLU with its variants (such as Leaky ReLU function) most effectively in hidden layers. The trial-and-error method was used to determine the activation function of the hidden layers.

2.11.2 Optimization and Learning

Once the network infrastructure has been constructed, it is now necessary to talk about the hyperparameters that control how the network learns and modifies its parameters to produce the least amount of error.

Optimization is a field of mathematics and computer science that researches methods and approaches created especially for discovering the "best" answer to a particular "optimization" problem. The goal of optimization is to maximize or minimize some value in a situation. The minimization of a loss function(*f*), is used to define the problem of learning. It is a formula that assesses a neural network's efficiency on a certain dataset [82].

The loss function typically consists of two terms: an error and a regularization term. The neural network's fit to a set of data is quantified by the error term. The regularization term prevents overfitting by restricting the model's complexity[83]. Optimizing the error function in neural networks requires modifying the weights and biases until identifying the optimal weights or the best weight values that produce the least amount of error. In perceptron, is considered one of the simplest forms of neural network that only have one weight (*w*) and one output. Figure 2.20 shows the 2D curve that tries to minimize errors concerning the weight. This section explains which hyperparameter should be tuned and how to tune it at each step as follows:

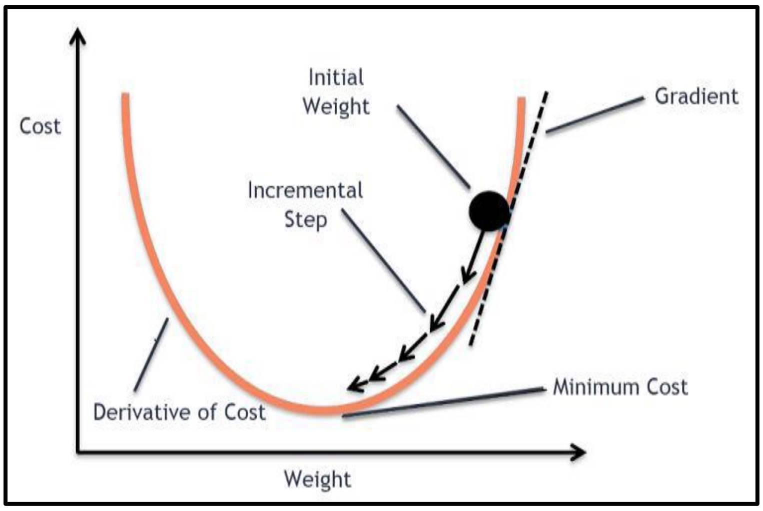


Figure 2.20 The weight-related cost function[84]

2.11.2.1 Optimization Algorithms

Gradient descent (GD) is the most straightforward method of training; it is considered a first-order method because it requires information derived from the gradient vector. During the training phase of a neural network, the parameters (weights and biases) are updated using the fundamental optimization procedure known as gradient descent. It aims to minimize a cost or loss function by repeatedly modifying the parameters in the direction of the steepest fall.

Assume error function *f(x)*, and $x \in \mathbb{R}^n$ represent weights, ∇*f(x)* is the equivalent gradient. Iteration k's step size is equal to η. Equation 2.28 explains the GD formula as follows:

$$x^{i+1} = x^{i} - \eta \nabla \mathrm{f(x)}\, , i = 0,1,2, \ldots. \qquad 2.28$$

where ∇*f(x) is* $(\frac{\partial E}{\partial x})$ that refers to the weight-dependent derivative of the *E*. The factor η represents the rate of learning. One-dimensional optimization along the training route can be used to decide this value at each step or set it to a constant value. It is often desirable to obtain an ideal value for the training rate through line minimization at each succeeding step[85]. Algorithm 2.1 shows the algorithm of GD.

Algorithm 2.1 GD Algorithm[84]

```
Algorithm GD

1. x= random vector as initial.
2. While (x is the best solution or time is run out)
3.          $x^{i+1} = x^{i} - \eta \frac{\partial E}{\partial W}$
4. End while
5. return x
```

A particularly effective approach for locating the minimal error is gradient descent. But it has two significant flaws. First, not all cost functions resemble straightforward bowls. Reaching the minimal error may be challenging due to holes, ridges, and other types of irregular terrain. That means there are local minimum problems. Second, the whole training set is utilized in gradient descent to calculate the gradients at each stage[72].

The solution that solves these two issues. First, Stochastic Gradient Decent method, Adaptive moment estimation (Adam), adaptive moment estimation (Adam), and Adaptive Gradient (AdaGrad). Second, the mini-batch technique takes a sample of the dataset with a specific size called batch size to feed the network and update the weights. In this dissertation, use the Adam optimizer most widely recommended by many authors, and the mini-batch gradient descent technique.

2.11.2.2 Adaptive moment estimation Optimizer (Adam)

Adam is a well-known optimization technique used to modify the parameters of DNN, in particular, during training. It adds a new feature for controlling the momentum and adaptability of the learning rate while combining the advantages of both the Adagrad and RMSProp optimizers. It is ideally suited for a variety of optimization issues since it is built to effectively handle sparse gradients and non-stationary targets[86], [87].

Adam optimizer depends on the momentum and GD approaches. It implements an adaptive learning rate for each parameter. This learning rate is adjusted based on the first and second moments of the gradients as explained in Equation 2.29 and 2.30 respectively. This feature enables the automatic adjustment of learning rates for various parameters, hence enhancing the convergence speed[86]. The first momentum and second momentum are acquired by the following formula:

$$m1_j = \beta_1\, m1_{j-1} + (1 - \beta_1) g_j \qquad 2.29$$

$$m2_j = \beta_2\, m2_{j-1} + (1 - \beta_2)\left(g_j\right)^2 \qquad 2.30$$

Where $m1_j$ is the first momentum, $m2_j$ is the second momentum, this step helps adapt the learning rate for each parameter, g_j represents the gradient operation, $\beta_1$and β_2 are the Adam parameters, control the exponential decay rates for the first and second moments, with values of 0.9 and 0.999 respectively[87]. Equation 2.31 and Equation 2.32 explain how to correct the bias of $m1_j$ and $m2_j$ respectively.

$$\overline{m1_J} = \frac{m1_j}{(1 - \beta_1)} \qquad 2.31$$

$$\overline{m2_J} = \frac{m2_j}{(1 - \beta_2)} \qquad 2.32$$

Where $\overline{m1_J}$, $\overline{m2_J}$ are used to correct the bias. This step prevents the initial bias of $m1_j$and $m2_j$ to become 0. Equation 2.23 explains how to update weights.

$$w^{j+1} = w^j - \eta \frac{\overline{m1_J}}{\sqrt{\overline{m2_J} + \varepsilon}} \qquad 2.33$$

Where $w^{j+1}$is the updated weight, ε epsilon equal to 10^{-8}, and η learning rate[87].

2.11.2.3 Mini-Batch Gradient Descent (MBGD)

Mini-batch Gradient Descent is widely employed as an optimization procedure for training ML models, specifically in the domain of DNN. It is a compromise between Stochastic Gradient Descent (SGD) and traditional gradient descent called Batch gradient descent (BGD). MBGD involves the partitioning of the training dataset into smaller batches of data. Instead of performing gradient computation and parameter updates on the entire dataset as observed in BGD or on a single data sample as observed in SGD. MBGD conducts gradient computation and parameter updates based on the average gradient calculated over each mini-batch[72], [87], [88].

2.11.2.4 Learning Rate and Scheduling

The learning Rate (LR) is by far the most crucial hyperparameter; hence it should always be fine-tuned. Presents an issue. This issue is as follows: when the LR is initially large, then the optimizer will subsequently exhibit oscillatory behavior nearby of the given spot for a long time. On the other hand, LR is initially small, the optimizer will take a long time to get the optimal solution. Figure 2.21 shows how the LR value affects the loss function during the training process.

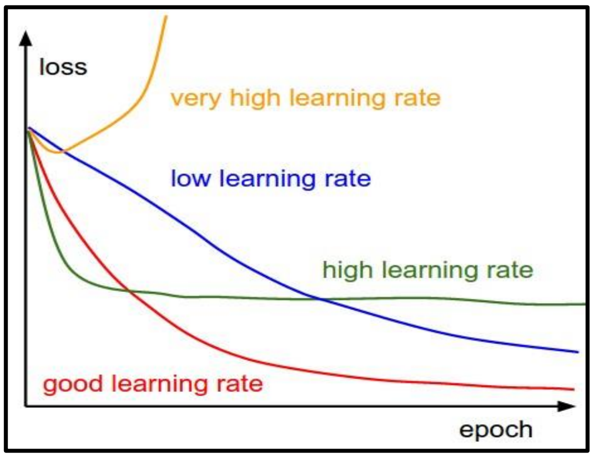


Figure 2.21 Comparison between the states of LR[72]

Learning rate scheduling (LRS) is a widely employed mechanism in the training of ML/DL models, specifically DL, to enhance their convergence and overall performance. The most common type of Scheduling function called (the decaying function) is an exponential function. Equation 2.34 explains LRS formula.

$$LR_t = LR_0 \times \; exp^{(-k \times t)} \qquad 2.34$$

Where (LR_0) refers to the initial learning rate with value equal to 0.1, (LR_t) refers to the learning rate in the (t) epoch. Finally, the (k) parameter controls the decay rate with value equal to 0.1.

2.11.2.5 Early Stopping Technique

It is common knowledge that DNN is trained using the BP method in phases, starting with the realization of relatively simple mapping functions and working their way up to more complex ones. The fact that the mean-square error reduces as the number of training iterations increases reflects this. The best generalization of the model is the goal of the training process. So, it is difficult to specify the ending learning process manually by showing the curve of learning for training by itself[89], [90]. In this case, need to use a mechanism to know the best time for ending the training process.

An algorithm known as early stopping is frequently used to choose when to stop training before overfitting takes place. Simply said, it keeps track of the validation error rate and ends training when it starts to rise[89]. How is it possible to accomplish this?

The dataset is divided into training, validation, and testing sets. The training set is used for training (computing the gradient and adjusting the parameter of NN). During the session, the error of the validation set is monitored. Normally, the error of validation will decrease during the training process. When the error of validation increases for a certain iteration number, the training process ends, and the parameters (weights) are returned on the last minimum validation error as shown in Figure 2.22.[90]

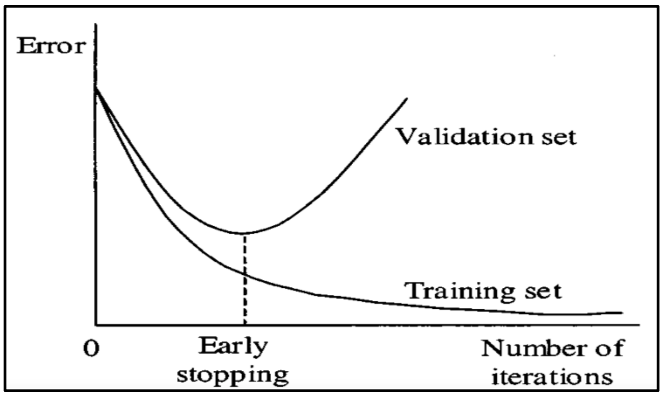


Figure 2.22 Early-Stopping[90]

2.11.2.6 Batch Normalization

Batch Normalization (BN) is a widely utilized technique in the field of deep learning, specifically in the training of deep neural networks. Its primary purpose is to enhance the stability and efficiency of the optimization process, hence facilitating convergence. The internal “covariate shift” problem (the distribution of data is changed) is effectively addressed with the implementation of input normalization during the training process[91].

This technique adds a layer to DNN, and works as follows: Firstly, it computes mean and standard deviation to mini-batch data x_i. Secondly, compute the Z-score of input mini-batch data. Finally, scaling and shifting the results. Equation 2.35 explains the shifting and scaling of the input.

$$out_i = \gamma \times x_i + \beta \qquad 2.35$$

Where out_i is the output of the BN layer, γ is the scaling parameter, and β is the shifting parameter [91].

2.11.3 Dropout Regularization

The utilization of dropout as a regularization approach in DNN is employed to reduce overfitting and enhance the generalization capabilities of the model. The process entails the random removal of a certain proportion of units throughout each iteration of the training process. If units are removed, every connection both in and out from those units must also be removed. Note, that these units are samples that only come from the input layer and hidden layers. This implies that the units that have been dropped out do not play a role in either the forward or backward pass of that particular iteration, hence generating a collection of various network structures[92], [93].

In each iteration of the training process, a subset of units, as determined by a dropout rate or called probability parameter (p), is stochastically chosen and assigned a value of zero, therefore essentially eliminating them from the computation for that particular iteration[92]. Figure 2.23 shows the process of dropout in the training process.

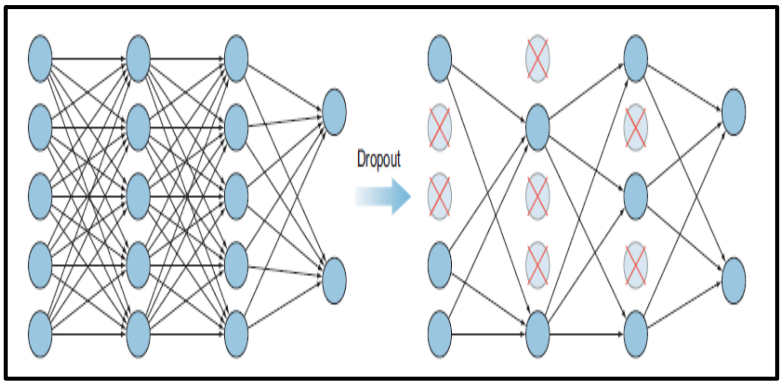


Figure 2.23 Dropout process [72]

2.11.4L1-Regularization

L1-regularization, alternatively referred to as Lasso regularization, is a regularization methodology employed in the fields of ML and DL to prevent overfitting, and feature selection, and enhance the generalization capabilities of models. This is achieved by incorporating a penalty term into the loss function during the training process, which promotes sparsity in the model's weights. Sparsity refers to the phenomenon where some weights are precisely driven to zero[82], [94]. Equation 2.36 illustrates the loss function after adding the absolute values of the model parameters to the lasso regression (L1 regularization).

$$L1\,loss = L + \lambda \sum_{j=0}^{n} |w_j| \qquad 2.36$$

Where L is the loss function can be (MSE or Cross Entropy Binary (CEB). λ is the parameter that controls the strength of regularization with value 0.1. A higher value emphasizes regularization more strongly. The **w** is the weight of the model. Equation 3.37 explains the (CEB) formula[82].

$$CEB = -y \times log(\hat{y}) - (1-y) \times \log(1-\hat{y}) \qquad 2.37$$

Where y is the actual output, $\hat{y}$ is the predicted output.

L1 regularization exhibits an ability for feature selection. Because the regularization strength is increased it tends to force the weights associated with less informative features towards zero, so efficiently choosing a subset of the most pertinent features for the given task. In alternative terms, certain input has the potential to be deleted[82].

2.11.5 Genetic Algorithm

Genetic Algorithms (GA) belong to a category of evolutionary optimization methodologies that draw inspiration from the mechanisms of natural selection and genetic diversity. It offers a robust methodology for addressing complicated optimization and search issues by emulating the mechanisms of evolution. GA is extensively employed across many disciplines such as computer science, engineering, finance, biology, and other related domains. This method demonstrates high efficiency in addressing optimization problems characterized by a complicated, nonlinear solution space that lacks a readily apparent mathematical description[95].

The fundamental principle underlying a GA involves the iterative process of developing a population of feasible solutions to a given problem.

The population consists of a set of candidate solutions called (individuals, solutions, or chromosomes). Firstly, each individual must encode in a form that enables the algorithm to manipulate and assess it. Secondly, apply GA to get the best solution. The key steps of GA as follows[96], [97]:

- **Initialization**: A population of randomly chosen solutions is created.
- **Fitness**: it measures the effectiveness of each solution to the problem.
- **Selection**: The selection of solutions from the existing population to fulfill the role of parents for the subsequent generation.
- **Crossover:** During the process of crossover, pairs of carefully chosen parents engage in the exchange of genetic information, resulting in the production of offspring.
- **Mutation**: Some solutions in the population have their genetic code randomly altered or mutated. It enables the investigation of new areas of the search space.
- **Replacement:** Depending on the fitness values of their individuals, new offspring take the position of the preceding generation's individuals. This guarantees that the population retains better solutions.
- **Termination:** The algorithm continues until a stopping requirement is satisfied, such as convergence of population fitness or a set number of iterations.

2.12 Unsupervised Learning Techniques

With unsupervised learning, the goal is to find hidden correlations, structures, relationships, or hierarchies using the system's sampled data [8]. The data can be comprehended better and the underlying processes that produced it with the help of this type of information. Similar samples can be grouped into the same cluster by using similarity or distance measurements to build the clusters. Since the similarity of two samples can be seen as the distance between the two samples in the feature space, the similarity measure can also be thought of as a distance measure [10]. In this dissertation, USL was used based on neural network techniques to achieve some of its goals, including clustering and feature engineering, and they will be explained as follows:

2.12.1 Winner-Take-All Competitive Networks (WTA-CN)

WTA-CN networks are neural networks that select the greatest number of inputs from a collection of inputs and put them in groups. This network functions in an extreme contrast enhancement mode, with all other neurons in the network being suppressed while only the maximum stimulated neuron reacts. It is called a competitive neural network. Kohonen's network is one of the most famous competitive neural networks. It uses the clusters found in the training set to classify input vectors into one of the m categories that are supplied. The collection of **m-**weight vectors is treated as variable vectors by the learning process. All weight vectors that were randomly selected must be normalized before learning [98], [99]. Figure 2.24 shows the architecture of WTA-CNN

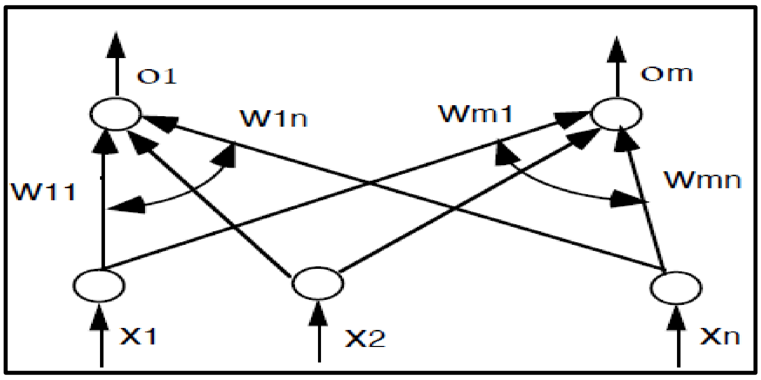


Figure 2.24 The winner-take-all neural network [99]

Where $\{x_1, x_2..., x_n\}$ is a vector of input data. $\{o_1..., o_m\}$ refer to output vector. The weights refer to $\{w_{11}, w_{1n}, ... w_{mn}\}$.

The selection of w_u in this kind of training is the weight adjustment criterion as explained in Equation 2.38.

$$\|x-w_u\| = \min_{i=1,\ldots,m} \|x-w_i\| \qquad 2.38$$

The vector $\mathbf{w_u}$**,** which is the most accurate representation (closest approximation) of the input **x**, and the index **u** indicate the selected unit number (winner unit) associated with each. It can be concluded that seeking the biggest among the **m** (scalar products) corresponds to looking for the minimum of the **m** distances as explained in Equation 2.392.39. Figure 2.25 explains select the winner weight w_1.

$$<w_u, x> = \max_{i=1,\ldots,m} <w_i, x> \qquad 2.39$$

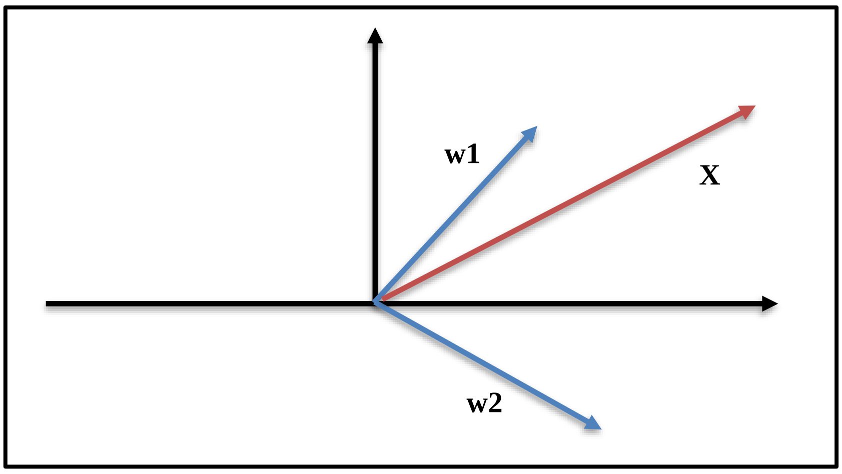


Figure 2.25 Description the winning weight (w_1).

The weight of the winning neuron must be modified after it has been detected and declared a winner to shorten the distance $\|x-w_u\|$ in the current training step. Equation 2.40 shows how the winner weight is updating.

$$w_u = w_u + \eta\,(x - w_u) \qquad 2.40$$

Where w_u is the winner weight, η is the learning rate.

2.12.2 Autoencoder technique

Autoencoders, which are ANN architectures, attempt to rebuild their input based on their output. In other words, they are intended to encode input data and then decode it to restore it to its initial state and develop effective data representations. It is generally employed in dimensionality reduction and unsupervised learning jobs. Particularly helpful applications of autoencoders include feature extraction and noise reduction. The reconstructed error of the network is used to calculate the loss function. There is no requirement for labeling because the loss function just uses the inputs and estimated outputs; as a result, the neural network can be learned without supervision [8], [100].

An encoder and a decoder are the two primary parts of an autoencoder's architecture. Here is a quick description of each element:

The encoder uses the input data to transform it into different representations in the latent space, which can have low, high, or equal dimensions. Usually, there are one or more hidden layers, followed by a linear or non-linear activation function [9].

The decoder reconstructs the initial input data using the encoded form it extracts from the latent space. It has hidden layers, just like the encoder [9]. Figure 2.26 illustrates the architecture of the Autoencoder neural network.

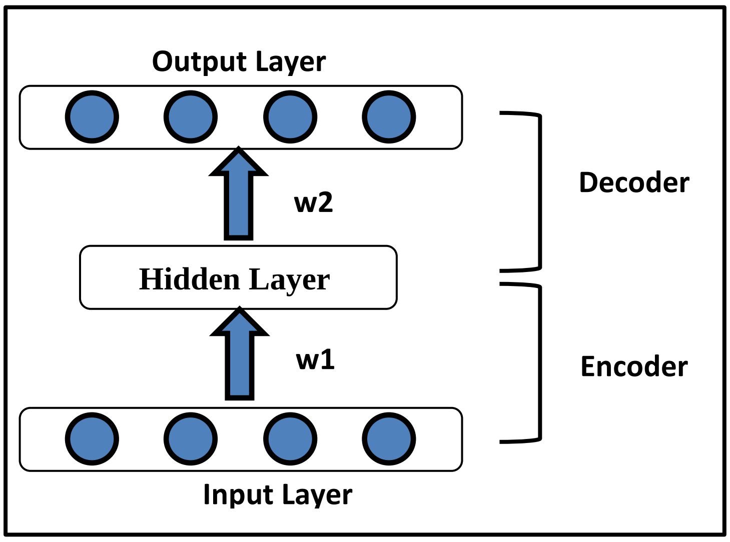


Figure 2.26 Autoencoder Architecture [9]

The network in the architecture shown in Figure 2.26 contains layers that are fully connected. Assuming that the input vector is $x \in \mathbb{R}^n$, there are **n** units in the input layer. The output layer similarly has an **n** of units. The **m** units of the hidden layer can be equal to, less than, or greater than **n**. The matrix of weights between the hidden and input layers is called **w1**. It is a matrix of $\mathbb{R}^{m \times n}$. The matrix of weights between the hidden layer and the output layer is called **w2**. It is a matrix of $\mathbb{R}^{n \times m}$ [100].

It is possible to train the structure by establishing a loss function, such as (the MSE loss function) from the difference between the input and the output, and minimizing the loss function concerning the network weights. BP algorithm is used to compute the gradient of the loss function concerning the weights, so can be used error BP to train the NN without using examples with labels [100].

2.13 Evaluation Measures

In this section, different methods of evaluating the effectiveness of deep learning models or machine learning are explored.

2.13.1 Performance Metrics for SL

The metrics have been recorded for evaluating the proposed classification model, including Recall, Specificity, Precision, F1-score, and Accuracy metrics. To assess the classification model's effectiveness, it is essential to examine the number of accurate and inaccurate classifications made by the model during testing. These counts are arranged in a table called Confusion-Matrix (CM), and all previous metrics rely entirely on it. Table 2.1 represents CM for a binary classification scenario [101].

Table 2.1 Confusion Matrix (CM)

		Predicted Class	
		positive	negative
Actual Class	positive	TP	FN
	negative	FP	TN

Where, True Positive (**TP**) occurs when an object is predicted to belong to a class, and it does indeed belong to that class. True Negative (**TN**) occurs when an object is predicted to not belong to a class and it does indeed not belong to that class. False Positive (**FP**) occurs when an object is predicted to belong to a class when in reality it does not belong to it. False Positive (**FP**) occurs when an object is predicted to belong to a class when in reality it does not belong to it. False Negative (**FN**) occurs when an object is predicted to not belong to a class when in fact it does belong to it [101]. The equations of evaluation metrics are as follows:

- **Accuracy Metric**: It is frequently used as an evaluation metric for binary and multi-class classification problems. It assesses the quality of the output by determining the proportion of accurate classifications over the total number of instances. Equation 2.41 is used to calculate this metric as follows: [46]

$$\text{Accuracy}=\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \qquad 2.41$$

- **Precision Metric (PM)**: It measures the proportion that is truly positive in the group identified by the classifier as the positive-class-based ***FP***. It is also called (positive predictive value). Equation 2.42 is used to calculate this metric [46]. However, because the precision metric only takes positive behavior into account, it is unable to identify growing behavior that has been forecast to decline. The metric is investigated as follows.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad 2.42$$

- **Recall Metric**: is used to identify the number of true positive classifications corresponding to the **FN**. Once more, it is inadequate because it cannot tell the difference between decreasing behavior that has been incorrectly predicted and rising behavior. Equation 2.43 is used to calculate this metric.

$$\text{Recall}=\frac{TP}{TP+FN} \qquad 2.43$$

- **F1-Score Metric**: In classification problems, the F1-score is a commonly used measurement that combines a model's recall and precision into a single measure of performance. It allows a fair evaluation of the ability of a model to correctly classify instances of both negative and positive labels of classes. Equation 2.44 explains the formula of this metric as follows:

$$\text{F1 score} = \frac{2\times TP}{2\times\ TP+FN+FP} \qquad 2.44$$

- **Specificity Metric**: Specificity, also known as the True Negative Rate (TNR). It assesses how well the model can recognize negative instances (i.e., examples of the negative class) among the real negative instances that are present in the dataset. Equation 2.45 explains the formula of this metric.

$$\text{Specificity}=\frac{TN}{FP+TN} \qquad 2.45$$

2.13.2Performance Metrics for USL

different methods of evaluating the effectiveness of autoencoder and clustering are explored. The metrics have been recorded for evaluating the autoencoder model and clustering, including Mean Square Error (MSE) and Silhouette Coefficient.

- **Silhouette Coefficient**: This metric is used in clustering to measure how closely the data in the cluster matches that of another cluster. The value of the silhouette coefficient ranges between (-1 and 1). If the value is close to 1, that means the clusters are different from each other. If the value is close to -1, that means the clusters are overlapping. If the value is equal to 0, means that the data does not support any interesting clusters[102]. Equation 2.46 explains silhouette coefficient formula.

$$\mathrm{Sillouette} = \frac{\mathrm{b - a}}{\max(\mathrm{a, b})} \tag{2.46}$$

Where **a** represents the average distance between a sample and all other samples in a similar cluster. **b** represents the average distance between a sample and all other samples in the nearest cluster.

- **Mean Square Error (MSE)**: It is a widely employed statistic in measuring the average squared difference between the predicted outcomes and the actual values within a given dataset[47]. Equation 2.47 explains the MSE formula.

$$\mathrm{MSE} = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.47}$$

Where y_i is the actual value and $\hat{y}_i$ is the predicted value.

2.14Evaluation Performance Model

After constructing the model, it can be used on the test set to forecast the class labels of unseen data instances. Evaluating the model's performance on the test set is valuable since it gives an impartial estimate of its generalization error. Therefore, performance measures can be used to compare learning algorithms, and this is done by knowing the class labels of the test records. The following section provides an overview of some of the frequently utilized techniques for assessing the efficiency of a classifier [46].

2.14.1Hold-out Approach

It involves dividing the initial data into two separate and non-overlapping sets known as the training data and the test data. Then, the test data is used to evaluate the model. The allocation of data for training and testing purposes is generally determined by the analysts and can vary, such as a 50-50 split or a ratio of two-thirds for training and one-third for testing. The easiest evaluation procedure has one drawback if there isn't much data, the test sets might have too few examples to statistically reflect the data at hand. In this case, using the k-fold approach is a solution to this problem[46], [103]. This dissertation split the dataset into three parts, including training, validation, and testing. The explanation for using the validation test is that to develop the model, its hyperparameters must be adjusted. Figure 2.27 shows the simple implementation of the hold-out approach.

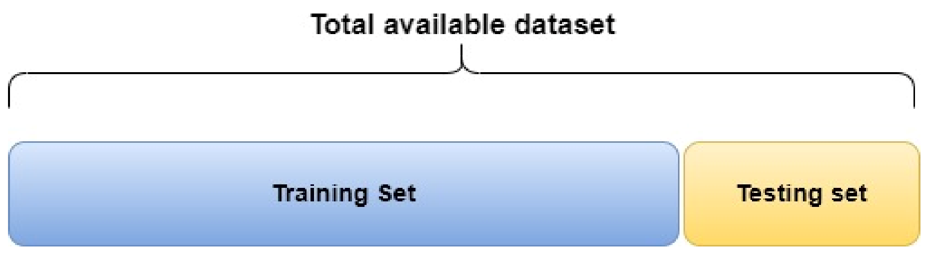


Figure 2.27 Hold-out approach

2.14.2K-fold Cross Validation Approach

It is a statistical method used to evaluate and compare learning algorithms. Data is split into k equal-sized parts. Where each iteration uses a k-1 segment for training and the remaining data for testing. That means the training and testing phases can be carried out repeatedly with different groups of data. To determine the overall accuracy of the model, all its iterations' accuracy is averaged together [46], [103]. The performance of our model has been evaluated using the 10-fold cross-validation. Often, it is favored to present unbiased results because it allows the proposed model to train on multiple training-testing splits. This will give the best indicator of the model's performance for unseen samples[104]. Figure 2.28 shows the simple implementation of the k-fold cross-validation approach.

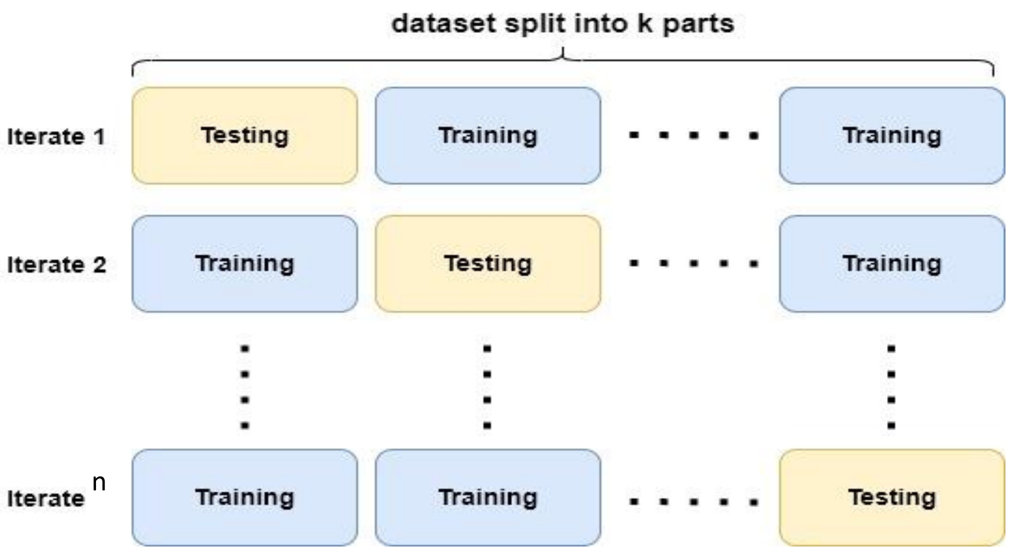


Figure 2.28 *K*-fold cross-validation approach

CHAPTER THREE:

The Proposed System

Design of the Proposed System

3.1 Introduction

This chapter presents the proposed deep learning model that can help in classification the diabetes as well as knowing its complications, which include diabetic ulcer foot and diabetic retinopathy. The proposed model depends on deep learning techniques. It deals with different data types including tabular data, ECG data, and image data. This model consists of four models, each dealing with a specific data type. The models are trained, and then the decisions resulting from these models are aggregated to classify diabetes and its complications.

This chapter presents the proposed system with its stages, shows algorithms to explain its operational stages, presents some outcomes, and provides illustrations for some of those stages. Finally, uses several methods to assess the system's performance strength in terms of accuracy.

3.2 The Architecture of the Proposed System

Figure 3.1 shows how the proposed system uses deep learning techniques to classify diabetes mellitus and its complications. It views four proposed models: CAER-DNN, ECG-CNN, GLCM-CNN, and CLAHE-CNN. First, the suggested model (CAER-DNN), which uses an autoencoder with regularization and unsupervised techniques to reconstruct and select features and DNN for the classification of the DM dealing with tabular datasets. Second, the proposed model (ECG-CNN), which uses statistical methods and CNN for extracting features and DNN for classification of the DM, uses the ECG dataset. Third, the proposed model (GLCM-CNN) uses the GLCM method with GA, the CNN-trained model for extracting important features, and DNN for classification of the DFU complications, uses the image dataset. Also, this model can be used for clustering the ulcer foot into three based on a competitive neural network with a winner-take-all technique. Fourth, the proposed model is called CLAHE -CNN, which uses CLAHE for enhancing an image and CNN for feature extraction, and then uses the DNN to predict the RD complications, uses the image dataset.

Each model in the proposed system has several stages to reach a decision, including the inductor function that is used to pass the source data type to the specific model, preprocessing, feature transformation, feature extraction, feature selection, and the classification stage, which is then aggregated with other decisions from another model to give the final decision. The system classifies the final decision into five classes: diabetic with DFU, diabetic with RD, diabetic with DFU, RD, and none.

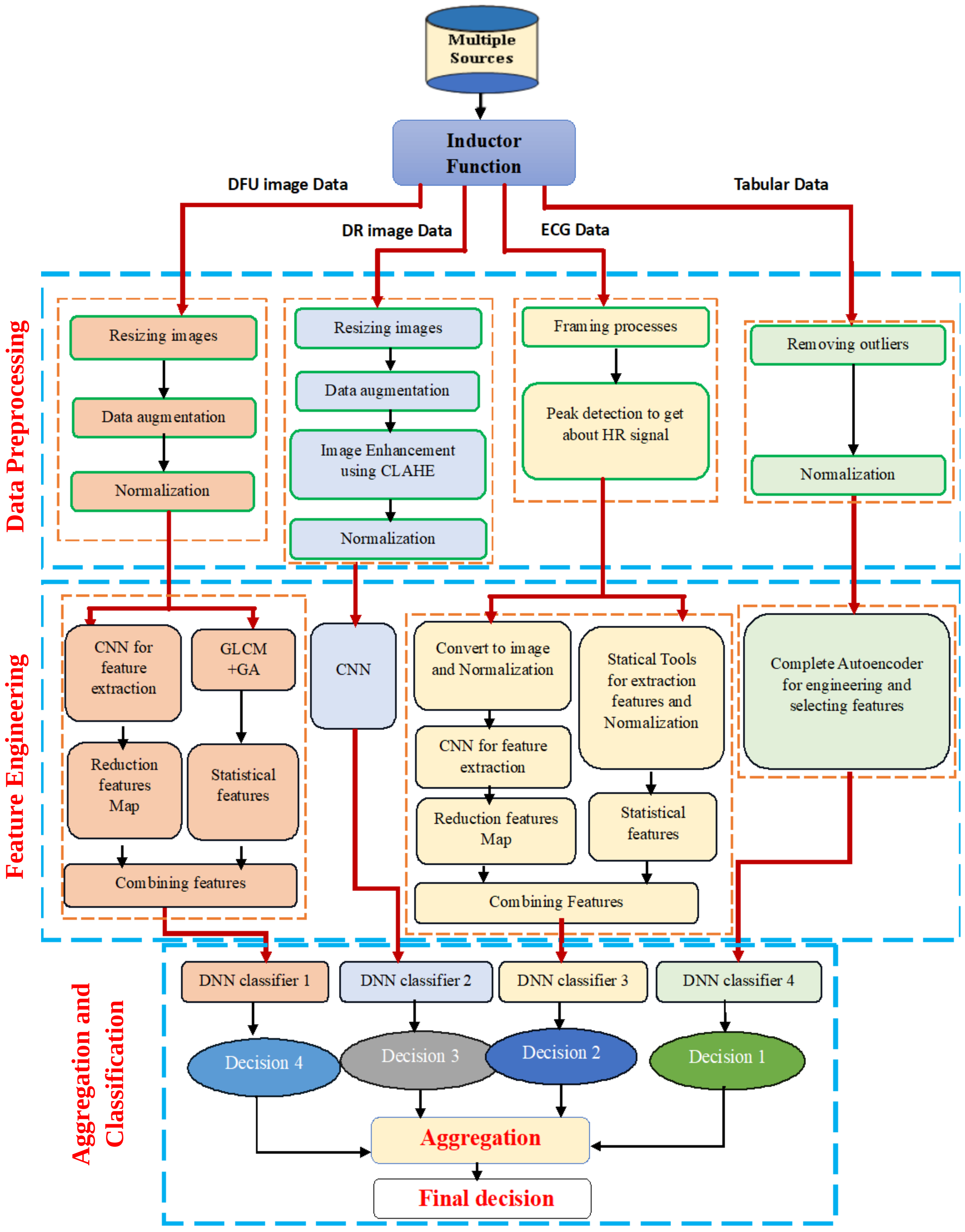


Figure 3.1 The diagram of the proposed system

3.3 The CAER-DNN model Methodology:

This model has been proposed to deal with tabular data in the proposed system. Figure 3.2 shows the structure of CAER-DNN model. This proposed model has classified the DM into binary classification (normal = 0 or abnormal = 1) based on autoencoder with regularization and DNN techniques. Algorithm 3.1 explains the CAER-DNN algorithm.

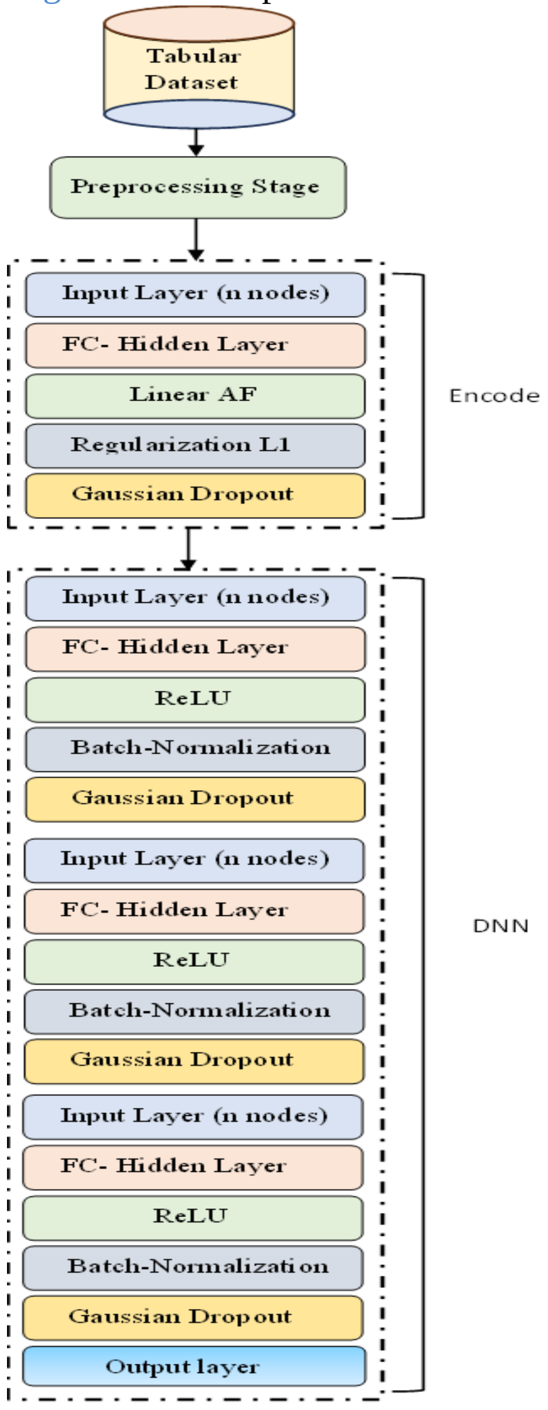


Figure 3.2 The architecture of CAER-DNN model

Algorithm 3.1 The CAER-DNN algorithm for classification of the DM based on tabular data.

```
Algorithm: CAER-DNN model
Input: Dataset (Orignal-D), number of features (no_features), epochs (E), batch size (BZ=32), learning rate α, dropout probability (p = 0.5).
Output: tab-model /*model with the best parameters*/
Begin:
Step 1: Preprocessing Stage:
        • Convert Class label to Numerical value
        • Normalization process for Original-D without a class label
        • Removing outliers
Step 2: Apply CAER:
        • D ← Apply CAER for features engineering and selection.
Step 3: Training new dataset that is output from CAER by the DNN Classifier:
        • Shuffled the D.
        • Y ← Class label of D
        • X ← Features of D
        • Standardization process for X.
Step 4: Split Dataset:
        Split X and Y into Training sets (Tr_X), labels of training set (Tr_Y), validation sets (Val_X) and labels of validation set (Val_Y), and Testing sets (Ts_X) and labels of testing set (Ts_Y).
Step 5: Initialize Parameters:
        • Weights ← Initialization randomly.
        • Biases ← Initialization with zeros value.
Step 6: Training Process:
        • Forward process:
        • for i ←1 to E do
              /*batch is array of patterns with size (BZ) that gets from (Tr_X) and label is 1D array with size (BZ) represents class labels of batch*/
        •   for batch, label in (Tr_X, Tr_Y) do
        •     output_train ← Apply DNN layers that finds in Figure 3.2 with(batch) as input
        •     Compute the loss and accuracy:
                o Loss ← Compute by Equation 2.37 between (output_train) and (label).
                o Accuracy ← Compute by Equation 2.41
        • Validation process:
            o output_val ← Apply forward process on Val_X as
              input with (Weights , Biases) parameters.
            o Accuracy ← Calculate the Accuracy between Val_Y
              and output_val.
        • Backward propagation:
            o Weights , Biases ← Update parameters using the
              Back-propagation algorithm and Adam optimizer
              with Equation 2.33.
        • End for batch
        • End for i
Step 7: Evaluation Process:
        • output_test ← Apply forward process on Ts_X as input with
          (Weights , Biases) parameters.
        • Accuracy ← Calculate the Accuracy between Ts_Y and
          output_test.
Step 8: tab-model ← Save the model with the best parameters
Step 9: return tab-model
End Algorithm
```

The CAER-DNN includes several stages that will be listed as follows:

3.3.1 Data Preprocessing Stage of the CAER-DNN Model

It is an essential step in the development of predictive models because the original format of the data is not good for classification, so it needs to be preprocessed. In this dissertation, the preprocessing methodology has been accomplished through a series of three fundamental processes, including (i) Outliers Removing and (ii) Data Normalization.

Data Normalization is used to optimize the training process and enhance the model's performance, it is necessary to modify the input data. in this step using the Max-min method.

Outliers' removal, the process entails the identification and management of data points that exhibit substantial dissimilarity from the prevailing majority of the dataset. In this dissertation, the Standard deviation method for determining the outliers and then removing them.

3.3.2 Feature Engineering and Selection Stage of the CAER-DNN Model

In this stage, an unsupervised technique called (autoencoder) is used for features engineering (generating new features). This dissertation uses a complete autoencoder (CAER) to do this stage. Figure 3.3 shows the structure of CAER. Table 3.1 shows the number of nodes and number of parameters with each layer in CAER model.

The encoder function is applied to reconstruct original features and generates newly learned features or observations. The decoder function is utilized to restore the newly acquired observations to their original format.

Additionally, regularization techniques are used to enhance the CAER performance, select more important features, and avoid overfitting. In this dissertation, two regularization techniques are used: Gaussian dropout and L1-regularization. Where Gaussian dropout is used to decrease the complexity of model and L1-regularization is used for feature selection by pushing the parameters of the model to zero for less important features. Algorithm 3.2 explains the CAER model.

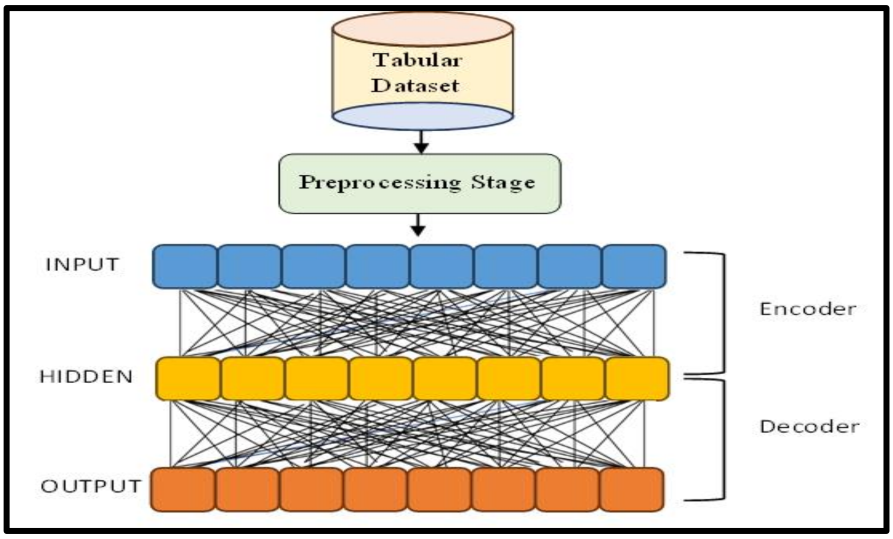


Figure 3.3 The architecture of a complete autoencoder

Algorithm 3.2 The Complete Autoencoder with regularization (CAER) algorithm

```
Algorithm: CAER model
Input: Dataset (D), epochs (E), batch size (BZ=32), learning rate α = 0.001, dropout probability (p=0.3), L1 regularization parameter(λ=0.1),
Output: learned_features /*output of encoder part*/.
Begin:
Step 1: Preprocessing Stage:
    • Shuffled the D.
    • Y←Class label of D and convert to Numerical data
    • X ←Features of D
    • Normalization process for X.
Step 2: Split X and Y into Training set (Tr_X), (Tr_Y) and validation set (Val_X) and (Val_Y).
Step 3: Initialize parameters:
    • W1, W2 ← Initialization randomly.
    • B1, B2 ← Initialization with zeros values.
Step 4: Training Process:
    • Forward process:
    •   For i ←1 to E do
          /*batch is array of patterns with size (BZ) that gets
           from (Tr_X) and label is 1D array with size (BZ)
          represents class labels of batch*/
    •     For batch, label in (Tr_X, Tr_Y) do
    •       Encoder part:
    •         Z ← (batch • W1) + B1   // (•) dot product operation
    •         OutEncoder ← linear (Z) // Equation 2.25
    •         Apply Gaussian dropping layer with probability (p)
              on OutEncoder.
    •       Decoder part:
    •         Y ← (Z • W2) + B2.
    •         OutDecoder ← linear (Y) // Equation 2.25
    •       Compute the loss function with L1 regularization:
    •         Loss (MSE) ← compute by Equation 2.47 between
                    (batch) and (outDecoder)
    • Backward propagation:
            o W1, W2 , B1, B2 ← Update parameters using the
              Back-propagation algorithm and Adam optimizer
              with **Equation 2.33**.
        • End for batch
        • End for i
Step 5: Evaluation Process:
        • output_test ← Apply forward process on Ts_X as input with
          (W1, W2, B1m B2) parameters.
        • MSE ← Calculate the MSE between Ts_Y and output_test.
Step 6: CAER_model ← save model with best parameters (W1, W2,
                                                       B1, B2)
Step 7: learnd_features ← Apply D on Encoder part in CAER_model
Step 8: return learned_features.
End Algorithm
```

Table 3.1 Description the architecture of a complete autoencoder model

Layer name	Output shape	AF	No. parameters
Input layer	(None, n)	None	0
Dense layer	(None, n)	linear	(n× n) + n
Gaussian dropout layer	(None, n)	linear	0
Output layer	(None, n)	linear	(n× n) + n

Where n represents number of input features. Equation 3.1 shows how to calculate the number of parameters of DNN model.

$$Number\ of\ parameters = (input\ neurons\ \times\ hidden\ neurons)\ +\ biases \qquad 3.1$$

3.3.3 Classification Stage of the CAER-DNN Model

In this stage, the DNN classifier is used to classify the DM. After applying the CAER in the previous stage, the results of the dataset with new features will be entered into the DNN classifier, before that, the dataset will be preprocessed, applying a data standardization process, and split into training sets, validating sets, and testing set. After that, train the training set by the DNN classifier. DNN classifier consists of the following: Input layer, three blocks, each block includes a dense layer (FC layer), BN layer, and gaussian dropout layer. Finally output layer. Table 3.2 shows the number of nodes and number of parameters with each layer in the DNN classifier.

Table 3.2 Description of the architecture of a DNN classifier of CAER-DNN model with number of neurons in each layer and number of parameters

Layer name	Output shape	No. parameters
Input layer	(None, 8)	0
Dense layer	(None, 2048)	18432
Batch-normalization layer	(None, 2048)	8192
Gaussian dropout layer	(None, 2048)	0
Dense layer	(None, 2048)	4196352
Batch-normalization layer	(None, 2048)	8192
Gaussian dropout layer	(None, 2048)	0
Dense layer	(None, 2048)	4196352
Batch-normalization layer	(None, 2048)	8192
Gaussian dropout layer	(None, 2048)	0
Output layer	(None, 1)	2049

3.4 The GLCM-CNN model Methodology

The proposed of this model aims to address and handle the image data within the proposed system. This proposed model has predicted the DFU as normal or abnormal. It consists of several steps including the preprocessing step (resizing image, image data augmentation, normalization), the feature extraction step that considers the important step, based on the GLCM method and trained CNN model with SVD for extracting the important features, and then using DNN and SVM classifier for classification the DFU normal or not.

Figure 3.4 shows the architecture of the GLCM-CNN model. Algorithm 3.3 explains the steps of the GLCM-CNN model for classifying the DFU images dataset. Algorithm 3.4 explains how to extract CLCM and CNN features.

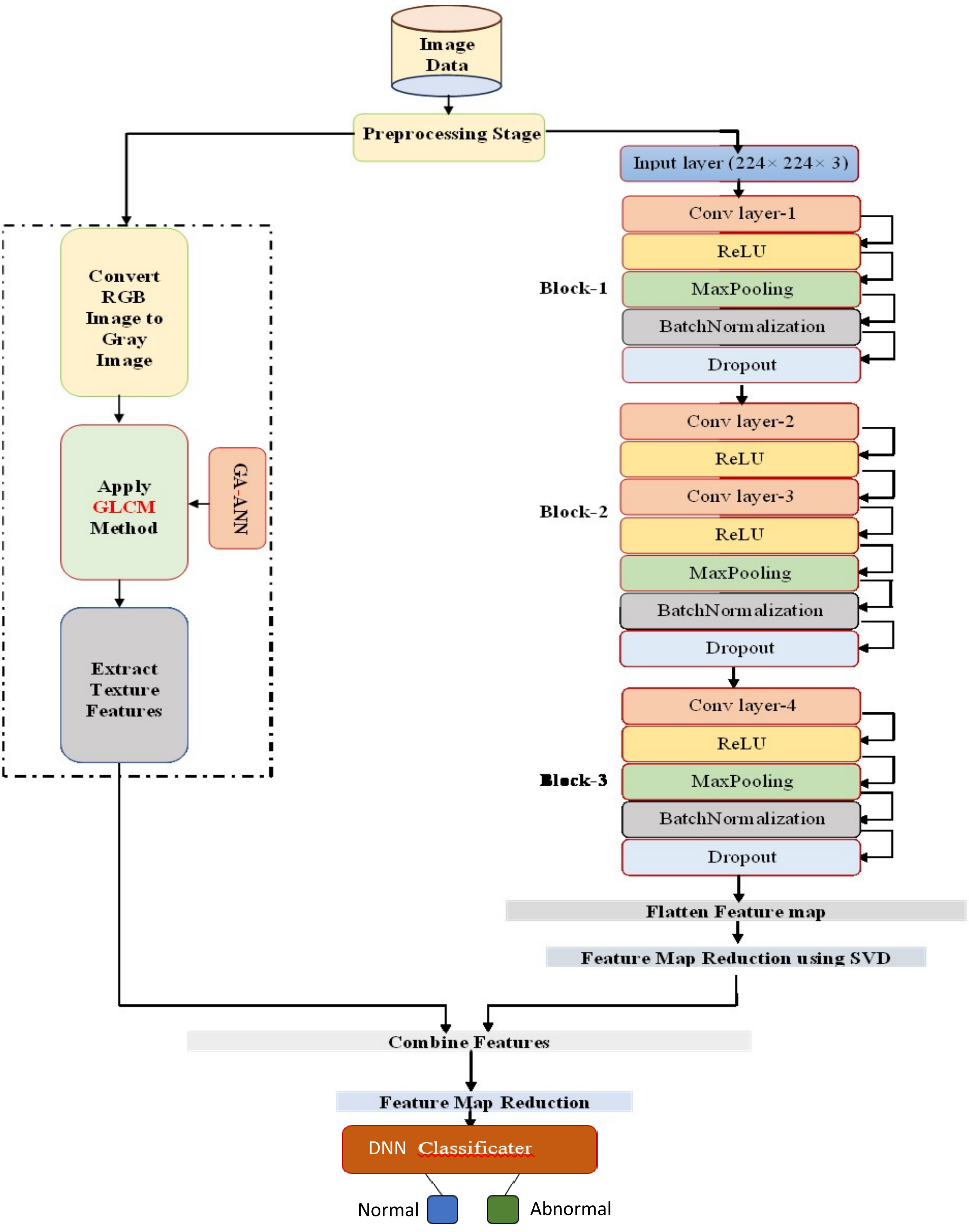


Figure 3.4 The architecture of GLCM-CNN model

Algorithm 3.3 The GLCM-CNN algorithm

```
Algorithm:  GLCM-CNN model
Input:   Ulcer dataset (UD). epochs (E), batch size (BZ), learning rate
α, dropout probability (p = 0.5, Pairs of distance and angle as array
(arr_d_a) with size (5,2).
Output:  DFU-model   /*model with best parameters*/
Begin:
Step 1: Preprocessing stage:
•  Shuffled the UD.
•  Resizing images by (224*224)
•  Training set (Tr_X), (Tr_Y), Validation set (Val_X) and (Val_Y) and
   Testing set (Ts_X) and (Ts_Y).
•  Data Augmentation for Tr_X:
         o  Rotation by (45°)
         o  Zooming
         o  Shearing
         o  Flipping
•  Normalization process divided by 255.
Step 2:  Initialize parameters:
•    Weights ← Initialization randomly.
•    Biases ← Initialization with zeros value.
Step 3: Feature Extraction:
      •  /* Apply Algorithm 3.4*/
      •  New_Tr_X  ← GLCM_CNN_FE (Tr_X, BZ, arr_d_a )
      •  New_Val_X  ← GLCM_CNN_FE (Val_X, BZ, arr_d_a )
      •  New_Ts_X  ← GLCM_CNN_FE (Ts_X, BZ, arr_d_a )
Step4: Training Process:
•     Forward process:
•     for i ←1 to E do
        for batch, label in (New_Tr_X, Tr_Y) do
•          Apply DNN layers that finds in Figur 3.2 with(batch) as
           input
      •  Compute the loss and accuracy:
            o  Loss ← Compute by Equation 2.37 between
               (output_train) and (label).
            o  Accuracy ← Compute by Equation 2.41
      •     Validation process:
            o output_val ← Apply forward process on New_Val_X
              as input with (Weights , Biases) parameters.
            o Accuracy ← Calculate the Accuracy between Val_Y
              and output_val.
        • Backward propagation:
            o Weights , Biases ← Update parameters using the
              Back-propagation algorithm and Adam optimizer
              with Equation 2.33.
•       End for batch
•     End for i
Step 5: Evaluation Process:
        • output_test ← Apply forward process on New_Ts_X as input
          with (Weights , Biases) parameters.
        • Accuracy ← Calculate the Accuracy between Ts_Y and
          output_test.
Step 6: DFU-model ← Save the model with the best parameters
Step 7: return DFU-model
End Algorithm
```

Algorithm 3.4 The algorithm for extracting features by GLCM and CNN trained model

```
Algorithm:   GLCM_CNN_FE function
Input:  input data (data), batch size (BZ), Pairs of distance and angle
as array (arr_d_a) with size (5,2)
Output: new_features  /* features that produce from combine CNN
features and GLCM features*/
Begin:
•   For batch in (data) do
        // Algorithm 3.8
•       X  ← Apply CNN_features (batch, BZ)
        // Algorithm 3.9
•       Z ←Apply GLCM_features (batch, BZ, arr_d_a)
•       new_features  ←Concatenate (X, Z)
•   returen new_features
End Algorithm
```

The GLCM-CNN model includes several stages that will be listed as follows:

3.4.1 The Image Data Preprocessing of the GLCM-CNN Model

In this stage, the following preprocessing techniques will be used to prepare the image data for the classification process: (i) image resizing, (ii) image data augmentation, and (iii) normalization.

Image resizing is considered one of the most important techniques in image preprocessing. Because it enhances the performance of the classification process by focusing on the important features in the image and neglecting the unimportant ones, it reduces the computational overhead, optimizes the memory, avoids overfitting, and promotes generalization. In this model, Bilinear Interpolation is used to resize the image. This method is similar to the down-sampling method.

Data augmentation is a crucial pre-processing approach that is effective in training deep learning models with high discriminative capabilities. In this step, data augmentation was used to enhance the results by applying a variety of image processing methods including rotation, flipping, scaling, and shearing. The rotation ensures that the model is unaffected by the object's orientation. The input images are rotated in any direction between 0 and 360 degrees randomly. When rotating an image, certain pixels will shift out the image, necessitating a fill-in using the image reflected in the model. The image is also flipped horizontally, after which the image is zoomed randomly. Shearing is typically used to enhance an image so that computers can identify how people see things from various angles. Finally, shifting the image means that an image is shifted to the left or right, bottom or up, randomly.

3.4.2 Feature Extraction Stage of the GLCM-CNN Model

Feature extraction in the process of classification plays an important role within the field of DL. At this stage, two techniques for feature extraction have been used. First, the CNN technique is used to extract the abstract features. It starts with training the proposed CNN as shown in Figure 3.5, and then uses it as a trained model for feature extraction by removing the specific part for the classification from it and using the flattened layer as output (e.g., vector of features map). After, producing the vector, the feature map reduction technique is called (SVD) for reduction. The objective of the reduction process is to focus on the important features and neglect the unimportant ones, as well as reduce the sparse. Algorithm 3.5 clarifies the steps of the trained CNN model. Algorithm 3.6 explains the steps of feature extraction by using CNN with SVD as reductional features.

Table 3.3 Summary architecture of trained CNN in GLCM-CNN model

Layer name	**Output Shape**	**No. Parameters**
Input layer	(224, 224, 3)	0
Conv layer & Relu	(224, 224, 64)	1792
Max Pooling2D	(112, 112, 64)	0
Batch-normalization	(112, 112, 64	256
Dropout	(112, 112, 64)	0
Conv layer & Relu	(112, 112, 64)	36928
Conv layer & Relu	(112, 112, 128)	73856
Max Pooling2D	(56, 56, 128)	0
Batch-normalization	(56, 56, 128)	512
Dropout	(56, 56, 128)	0
Conv layer & Relu	(56, 56, 128)	147584
Max Pooling2D	(28, 28, 128)	0
Batch-normalization	(28, 28, 128)	512
Dropout	(28, 28, 128)	0
Flatten	(100352)	0

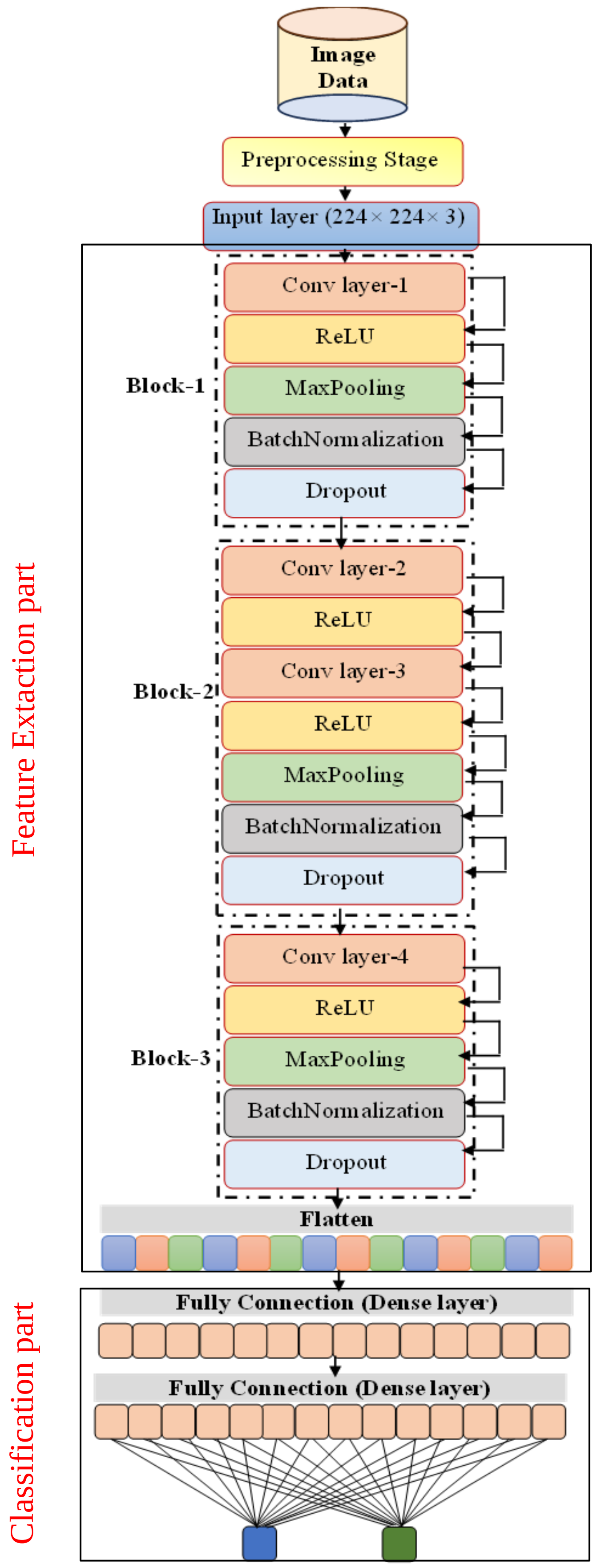


Figure 3.5 The diagram of the CNN model.

Algorithm 3.5 The trained CNN model for features extraction

```
Algorithm: traind-CNN model
Input:   Ulcer dataset (UD). epochs (E), batch size (BZ), learning rate
$\alpha = 0.001$, dropout probability ($p = 0.5$).
Output:  traind_CNN-model /* model with best parameters*/
Begin:
Step 1: Preprocessing stage:
        • Shuffled the UD.
        • Resizing images by ($224\times224$)
        • Training set (Tr_X), (Tr_Y), Validation set (Val_X) and
          (Val_Y) and Testing set (Ts_X) and (Ts_Y).
        • Data Augmentation for Tr_X:
              o Rotation by ($45^o$), Zooming
              o Shearing
              o Flipping
        • Normalization process divided by 255.
Step 2: Initialize parameters:
        • Weights ← Initialization randomly.
        • Biases ← Initialization with zeros value.
Step 3: Training Process stage:
        • Forward process:
        •   for I: = 1 to E do
        •     for batch, label in (Tr_X, Tr_Y) do
        •       output_train ← Apply CNN layers that finds in Figure
                3.5 with(batch) as input
        •       Compute the loss and accuracy
        •       Validation process.
        •       Backward propagation
        •     End for batch
        • End for i
Step 4: Evaluation Process.
        • output_test ← Apply forward process on Ts_X as input with
          (Weights , Biases) parameters.
        • Accuracy ← Calculate the Accuracy between Ts_Y and
          output_test.
Step 5: traind_CNN-model ← Save the model with the best parameters
Step 6: return traind_CNN-model
End Algorithm
```

Algorithm 3.6 Feature extraction using trained CNN model with SVD

```
Algorithm: CNN-features function
Input: Batch of color image (Batch_img) with size (BZ, N, M, 3), where BZ is Batch size, Max_pool_size = (2,2), dropout probability (p = 0.5).
Output: features_cnn /*vectors of features map*/
Begin:
Step 1: Extracting features:
        • Featurs ← Applying feature extraction part in traind_CNN algorithm 3.5 with Batch_img as input
Step 2: Features map reduction:
        • features_cnn ← Apply SVD method // Equation 2.15
Step 3: return features_cnn
End Algorithm
```

Second, using the GLCM method to extract the statistical features. It is considered one of the most common techniques used for feature extraction of image datasets. There are several steps to do that involve: converting RGB image to gray image as a first step. Then, tuning hyperparameters of GLCM that involve (distances, and angles) by using GA with ANN as shown in Algorithm 3.7. where angles are (0^o, 45^o, 90^o, 135^o) and distances are (1,2,3,4,5). At this step, select the best five pairs of (distance, and angle) based on the accuracy of the classification process. These hyperparameters will apply to the GLCM method as shown in Algorithm 3.8 that used to generate the GLCM matrix. After that, apply the best five pairs of hyperparameters to the GLCM matrix to generate thirty features as a vector. These features include Entropy, Contrast, Energy, Dissimilarity, Homogeneity, and Correlation. Algorithm 3.9 explains how these features are calculated based on the best pairs of hyperparameters (distances and angles).

Algorithm 3.7 Tunning hyperparameters of GLCM based on GA with ANN

```
Algorithm: Select best hyperparameter GLCM using built-in GA

Input: 1D array of distances (d), 1D array of angles (ang), population
size (N), DFU image dataset (D) with image size (M, N,3).

Output: best_glcm_hyper /*best pairs of distance and angle as a 2D
array with size (5,2) */

Begin:
Step 1: Prepare the D dataset.
Step 2: Represent the solution or individual:
        • Initial d_a array:
            o d_a ← {{1, 0}, {2, 0}, {3, 0}, {4, 0}, {5, 0},
                     {1, 45}, {2, 45}, {3, 45}, {4, 45}, {5, 45},
                     {1, 90}, {2, 9}, {3, 90}, {4, 90}, {5, 90},
                     {1, 135}, {2, 135}, {3, 135}, {4, 135},
              {5,135}}
        • Initial population (P):
        •   for i ←  1 to N do
        •       // P[i] represents the solution (e.g., {2,9,1,0,19})
        •       for j  ← 1 to 5 do
        •           P[i][j]← a random number between 0 and 19
                             without repetition
            end for j
        • end for
Step 3: Calculate the fitness function (represents the accuracy) based
on ANN:
        • for i ← 1 to N do
        •   for j ← 1 to 5 do
        •     hyper_pra_vector [i][j] ← d_a [P[i][j]]
              //new_data represents new dataset with glcm features
        •     new_data ← Apply the (GLCM_ feature) with
                           hyper_pra_vector, D, and size image
                           (M, N,3).
        •    Fitness_value[i] ← Apply ANN on new_data to get the
                                accuracy of ANN.
        •   end for j
        • end for i
Step 4: Apply GA:
        • The dimension of individual (5), population P with size N,
          and Fitness_value.
          // best solution (e.g., {2,6,1,7,5})
```

```
• Best_selution ← get the best solutions.
• for i ← 1 to 5 do
•     best_glcm_hyper[j] ← d_a [Best_selution[j]]
• end for i
Step 5: return best_glcm_hyper.
End Algorithm
```

Algorithm 3.8 The GLCM method for creation glcm matrix

```
Algorithm: GLCM function
Input: Color Image Data (img) with size (N, M, 3), maximum gray
level of input image (mgl), distance (d), angle (θ).
Output: glcm_matrix /* 2D array with size (mgl × mgl)*/
Begin:
Step 1: Preprocessing Stage:
    • Convert color image to gray image:
    •   Initialize an empty matrix (gray_img)
    •    for ← 0 to N-1 do
    •      for j ←0 to M-1 do
    •          gray_img[i][j] ←0.2989 × img [i][j][0] + 0.5870 ×
               img [i][j][1] + 0.1140 × img [i][j][2]

Step 2: Calculate the glcm matrix
  • glcm_matrix ← initialize by zeros with size (mgl × mgl)
  • for i ← 0 to N-1do
  •  for j ←0 to M-1do
  •   if ( θ == 0) then:
  •         r_off ← i,
  •         c_off ← j + d
  •   else if ( θ == 45) then:
  •          r_off ← j + d
  •          c_off ← i + d
  •   else if (θ == 90) then:
  •           r_off ← i + d
  •           c_off ←j
  •   else if (θ == 135) then:
  •          r_off ← i + d
  •         c_off ← j + d
  •  if (r_off >= 0 and r_off < N) and c_off >= 0 and c_off < N) then
  •   glcm_matrix [gray_img [row, col], gray_img [r_off, c_off]+ ← 1
  •  end for i
  • end for j
Step 3: return glcm_matrix.
```

End Algorithm

Algorithm 3.9 Feature extraction by using GLCM method

Algorithm: GLCM_features

Input: *Batch of color image (Batch_img) with size (BZ, N, M, 3), where BZ is Batch size. Pairs of distance and angle as array (arr_d_a) with size (5,2)*
// arr_d_a = [[4,0^o], [3, 45^o], [4, 45^o], [2, 90^o], [5, 135^o]]

Output*: vec_fea /* vector of features with size (BZ, 30) */*

```
Begin:
Step 1: Initialize vector of feature (vec_fea)
Step 2: Compute the features of glcm based on glcm_matrix
    • Counter ← 0
    • b ← 0
    • for img in Batch_img:
    •  for I ← 0 to N-1
    •     for j ← 0 to M-1
    •       glcm_matrix ← GLCM method (img, arr_d_a[j][0],
            arr_d_a[j][1])
    •       vec_fea [b, count++] ← Calculate Equation 2.19 of
      glcm_matrix
    •       vec_fea [b, count++] ← Calculate Equation 2.20 of
      glcm_matrix
    •       vec_fea [b, count++] ← Calculate Equation 2.21 of
      glcm_matrix
    •       vec_fea [b, count++] ← Calculate Equation 2.22 of
      glcm_matrix
    •       vec_fea [b, count++] ← Calculate Equation 2.23 of
      glcm_matrix
    •       vec_fea [b, count++] ← Calculate Equation 2.24 of
      glcm_matrix
    •    End for j
    •  End for i
```

```
        • b← b + 1
        • End for img
Step 3: return vec_fea
End Algorithm
```

Finally, combine the abstract features that are produced from CNN with SVD and statistical features that are produced from GLCM to do two different tasks. The first task is to predict the DFU as normal or not by using two classifiers DNN and SVM. The second task is to cluster the DFU abnormality images into three clusters that represent the state of the ulcer foot and then take to each cluster some advice to maintain the integrity of his foot and not worsen the condition.

3.4.3 The Classification Stage of the GLCM-CNN Model

There are two classifiers used, including a DNN and an SVM classifier. The objective of using DNN and SVM for comparison between them. The DNN has proved its ability for classification tasks, and also the selection of SVM among all methods due to its ability to classification of binary tasks and can be able as a neural network.

The features from the CNN model and GLCM approach are combined and entered as input to the DNN or SVM. The DNN consists of two input layers (GLCM features and CCN features with SVD), a concatenate layer that combines previous layers, and three dense layers after which comes the Relu activation function immediately. This is followed by three dropouts with different probabilities (0.3, 0.5) after each dense layer aims to prevent overfitting. An output layer with a sigmoid activation function is utilized to give the probability of a normal or abnormal class. Another hyper-parameter used in this classifier is the learning rate (0.001), The Adam optimization method is used with the back-propagation approach, whereby the loss function is the binary cross-entropy, the number of epochs is equal to (100), and the batch size equal to (32). Table 3.4 describes the layers, the number of neurons for every layer, and the number of parameters (weights).

Table 3.4 Summary architecture of DNN classifier of the GLCM-CNN model

Layer name	Output Shape	No. Parameters
Input layer1	(None, 512)	0
Input layer2	(None, 30)	0
Concat_inputs	(None, 542)	0
Dense1	(None, 128)	69504
dropout	(None, 128)	0
Dense2	(None, 128)	16512
dropout	(None, 128)	0
Dense3	(None, 64)	8256
dropout	(None, 64)	0
Output layer	(None, 1)	65

3.4.4 Clustering Task using Winner-Take-All Competitive Neural Network

The winner-take-all competitive neural network (WTA-CN) is used to achieve the second task (e.g., clustering the states of DFU abnormal foots) as shown in Figure 3.6. There are three clusters, which represent the state of ulcer foot including (C1 = "Mild," C2= "Moderate," and C3 = "Severe". This process has several stages as follows: (i) feature extraction stage by using feature extraction part in the GLCM-CNN model, which uses two techniques for extracting features CNN trained model with SVD to extract abstract features and GLCM to extract statistical features. (ii) Combining abstract and statistical features and apply SVD for the reduction of the features (iii) Applying WTA-CN technique for the clustering process. Algorithm 3.10 clarifies the steps of the clustering process for the state of the abnormal DFU.

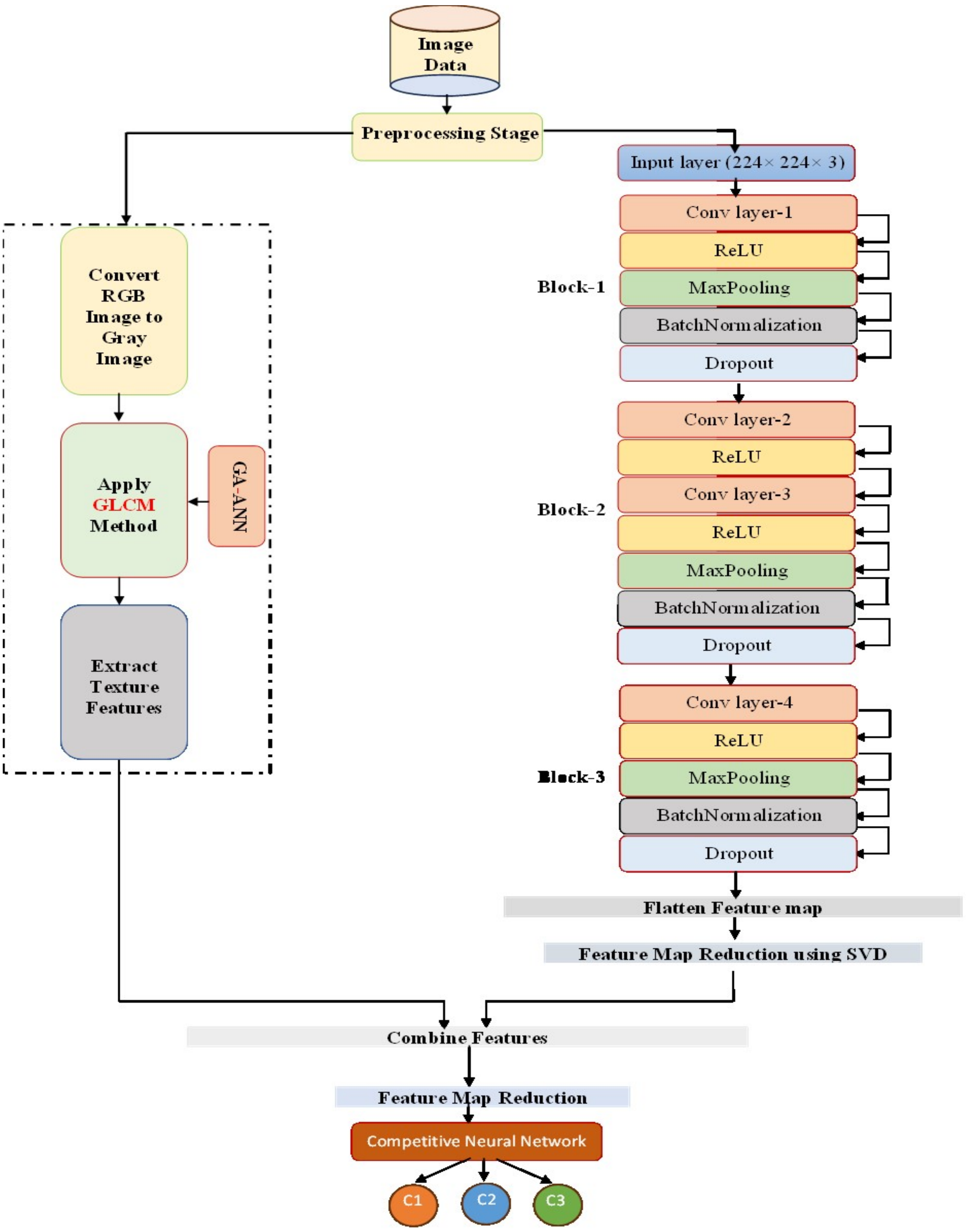


Figure 3.6 The diagram of GLCM-CNN for Clustering the DFU abnormal

Algorithm 3.10 Clustering the DFU abnormal images into three clusters by using a competitive neural network

```
Algorithm: Winner-Take-All Competitive Neural Network
Input:  Ulcer abnormal dataset (UND). epochs (E), batch size (BZ),
learning rate α= 0.0001, arr_d_a.
Output:  cluster_model /* the model with the best parameters */
Begin:
Step 1:  Preprocessing stage:
        •  Shuffled the UND.
        •  Resizing images by (224×224)
        •  Split UND into Training set (Tr_U), and Testing set (Ts_U).
        •  Data Augmentation for Tr_X:
              o  Rotation by (45°)
              o  Zooming
              o  Shearing process
              o  Flipping
        •  The normalization process divides images by 255.
Step 2: Apply the CNN_features model and GLCM_features model:
        •  Initialize features_vectors.
        •  For img_batch in Batches
        •     X←Apply CNN_features algorithm (img_batch, BZ)
        •     Z ← Apply GLCM_features algorithm (img_batch, BZ,
                                                  arr_d_a)
        •     Concat_vector ← Concatenate (X, Z)
        •     features_vectors←append (Concat_vector)
Step 3:  Apply SVD method for reduction:
        •  r_fea_vec ←Apply SVD for reduction of the features_vectors.
Step 4: Initial parameters:
        •  input_dim←no. of columns of r_fea_vec
        •  no_classes ← 3
        •  weights ←Initial weights as random with size (no_classes,
           input_dim)
Step 5: Training Process:
        •  Forward process
        •  for i←1 to E
        •   for vec in r_fea_vec:
        •     winner ← max (dot product (weights, x))
              weights[winner] +← α ×(x - weights[winner])
        • end for vec
        • end for i
Step 6: Evaluation Process:
        • Output←Apply forward process on Ts_U as input with
          (weights) parameters.
        • Using silhouette metric to evaluate the clusters
Step 7: cluster_model ← save model with best parameters
Step 8: return cluster_model
End Algorithm
```

3.5 The ECG-CNN Model Methodology

This model has been proposed to deal with ECG data in the proposed system. This proposed model has classified the DM as normal or abnormal based on DL techniques. The proposed model consists of several steps including the preprocessing step (framing process and peak detection), the feature extraction step that considers the important step, based on the statistical method and CNN trained model with SVD for extracting the important features, and then using DNN classifier for classification the DM normal or not. Figure 3.7 illustrates the architectures of the ECG-CNN model. Algorithm 3.11 clarifies the steps of ECG-CNN model. The ECG-CNN model includes several stages that will be listed as follows:

Algorithm 3.11 The ECG-CNN algorithm

```
Algorithm: ECG-CNN model
Input: ECG dataset (D). epochs (E), batch size (BZ), learning rate α,
dropout probability (p = 0.5).
Output: ECG_model /* model with best parameters */
Begin:
Step 1: Preprocessing stage:
        • Framing ECG signal:
              o F_data ←signal_framing(D). // algorithm 3.12
        • Split Dataset (D):
              o Training set (Tr_X), (Tr_Y), Validation set (Val_X)
                and (Val_Y), and Testing set (Ts_X) and (Ts_Y).
        • Data Augmentation for Tr_X.
        • Peak detection:
            o HR_Sig ←Apply Pan and Tompkinson method on
              Tr_X, Val_X, Ts_X.
        • Normalization
Step 2: Training Process:
        • For i←1 to E do
        •  For batch_hr in Tr_X
        •    X ←ECG_Statistical_features (batch) // algorithm 3.13
        •    X_image ← Convert signal to image by Spectrogram(X)
        •    Z: = Apply CNN_features (X_image, B) algorithm
             Classifier part:
        •    Output ← Apply the DNN layers in table 3.5
        •    Validation process
        •   Backward propagation
            o Update the parameters (weights, biases.) using the
              Back-propagation algorithm with Adam optimizer.
        • end for batch
        • end for i
Step 3: Evaluation Process stage:
Step 4: ECG_model ← save the model with the best parameters.
Step 5: returen ECG_model
End Algorithm
```

3.5.1 The ECG Data Preprocessing of the ECG-CNN Model

In this stage, the subsequent preprocessing techniques will be employed to preprocess the electrocardiogram (ECG) data to ready it for the classification procedure: (i) Framing process, (ii) Peak detection for generating HR signal.

Framing process, this process is used to take samples from the ECG signal with 1 minute start from the minute in fifteen. There are two causes to do that. First, the size of ECG data is very large with 60 minutes and a frequency sample equal to (250 fs), which means that the size of the ECG signal (60m×60s× 250 fs = 900000 samples in one ECG signal). Second, the first period of recording the ECG signal is usually unstable, so the sample after A sample is taken after some time as explained in Algorithm 3.12.

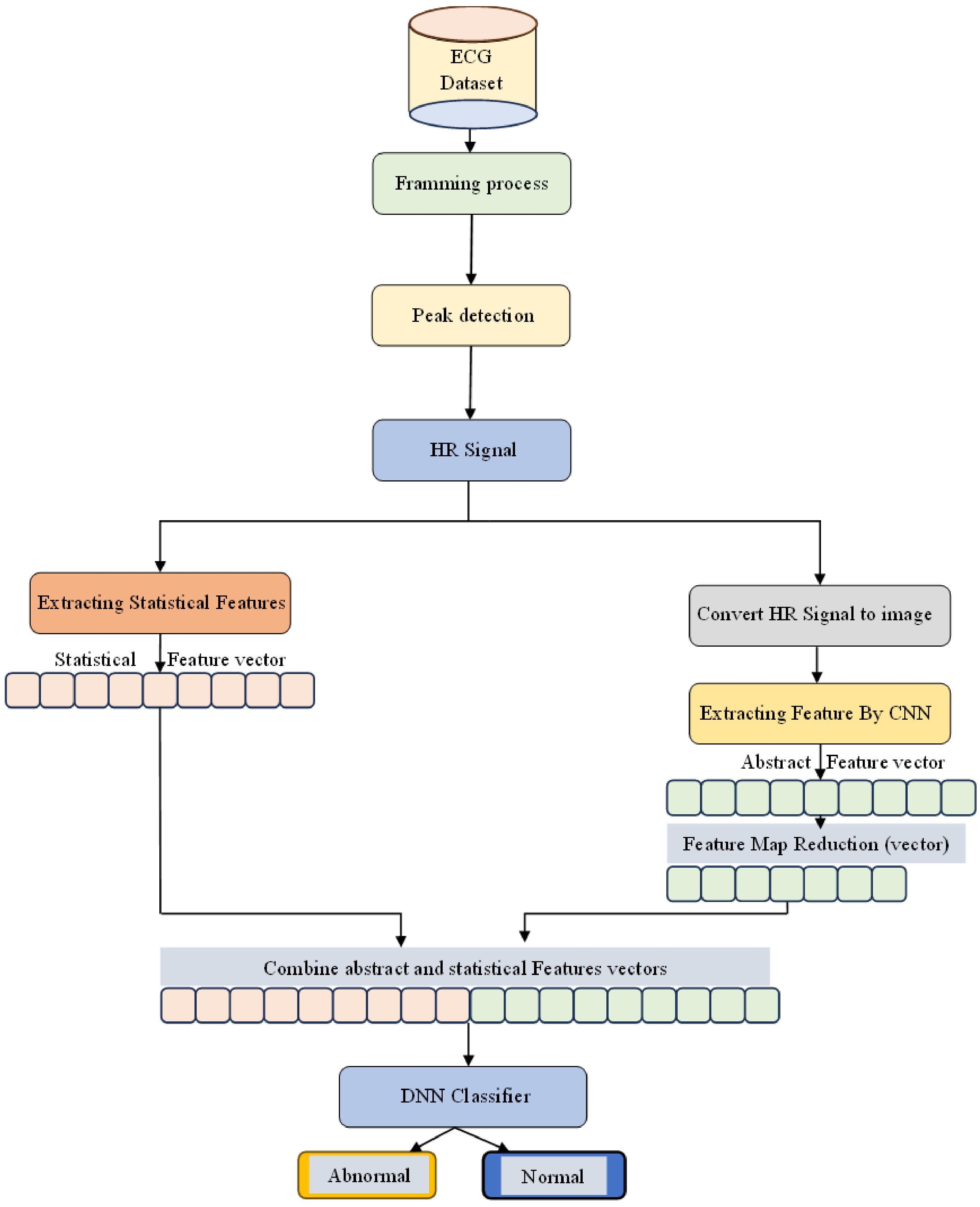

ECG Dataset
Framming process
Peak detection
HR Signal
Extracting Statistical Features
Statistical Feature vector
Convert HR Signal to image
Extracting Feature By CNN
Abstract Feature vector
Feature Map Reduction (vector)
Combine abstract and statistical Features vectors
DNN Classifier
Abnormal
Normal

Figure 3.7 The diagram of the ECG-CNN model

Algorithm 3.12 The Framing Process algorithm

```
Algorithm: signal_framing
Input: ECG signal dataset folders (ecg_data_folder), frequency-sampling (fs=250), duration (d=3 minutes), start_point (sp=15 minutes).
Output: frame_signal (frame with length 1 minute)
Begin:
Step 1: Framing process:
    • m ← 15
    • s ← 60
    • // sp mean sample
    • sp ← fs × s × m.
    • len_in_minutes ← fs × s.
    • samples_array ← initial 2D array with zeroes values
    • frame_signal ← initial 1D array with zeroes values
    • for ecg_file in (ecg_data_folder) do
    •     ecg_signal ← read ECG data signal from ecg_file
    •     ecg_signal ← removing the first column, which
                       represents the time column.
    •     // Copy the portion of the ECG signal to frame_signal
    •     startIndex ← sp × len_in_minutes
    •     endIndex ← (sp + d) × len_in_minutes
    •     for i ← startIndex to endIndex do
    •         frame_signal[i] ← ecg_signal[i]
    •     end for i
    • end for ecg_file
Step 2: return frame_signal
End Algorithm
```

Peak detection uses the Pan–Tompkins's method to detect the peak of the ECG signal, which is used to generate the heart rate signal. Figure 3.8 shows an example of the detection peaks of the ECG signal. This method consists of several steps as follows:

- Filter the ECG signal to remove noise (e.g., using bandpass or adaptive filtering).

- Compute the derivative of the filtered signal to enhance QRS slopes.
- Square the derivative signal to emphasize QRS complex peaks.
- Adaptive Thresholding: Calculate a threshold value based on the signal's energy. Use a moving average or other techniques to adjust the threshold
- Find Peaks: Identify local maxima in the squared derivative signal that exceeds the adaptive threshold. These maxima correspond to potential QRS complex peaks.
- Refinement: Search for the exact R-wave peak by analyzing the vicinity of each detected peak. Find the peak within a specified range that exhibits the highest amplitude. This reduces the likelihood of false positives due to noise.
- Measure the time intervals (RR intervals) between successive QRS complexes.
- Heart rate estimation based on RR intervals.

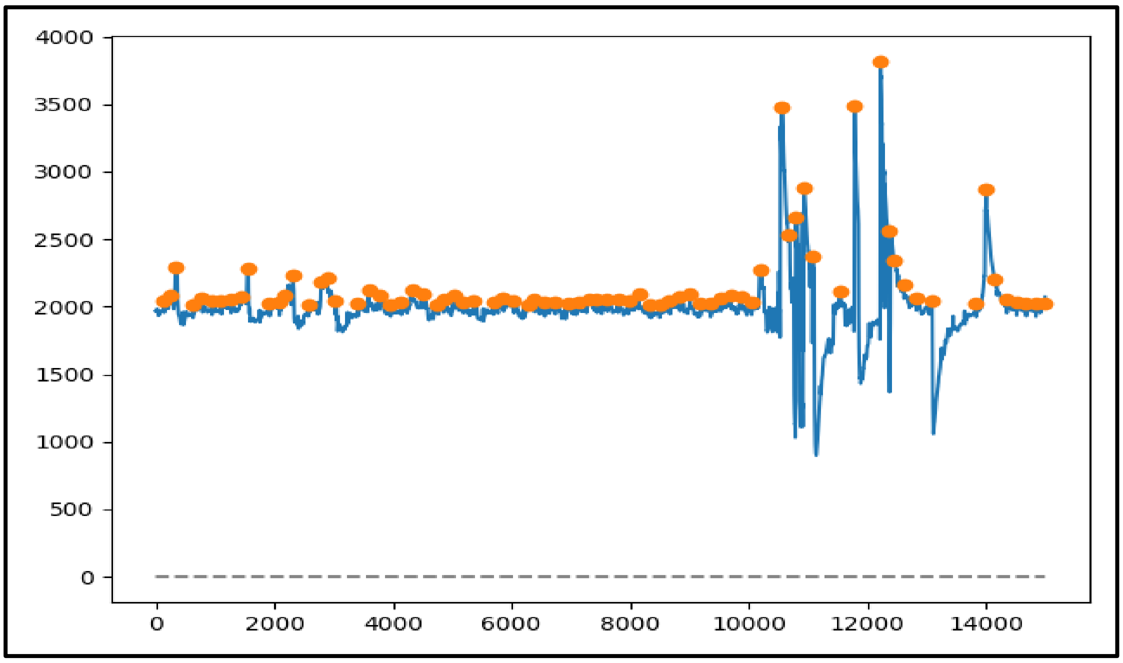


Figure 3.8 Peak detection of ECG signal

3.5.2 Feature Extraction Stage of the ECG-CNN Model

At this stage, there are two techniques for feature extraction which have been used. First, the statistical methods are used to extract the statistical features, including SDNN, RMSSD, NN50_count, pNN50, HRVI, TINN, meanNN, meanHR, and SDHR, which were mentioned in Chapter 2. These features are related to HRV that connects to DM disease. In addition, other statistical features are used include Harmonic mean (hm), mode (mod), median(med), mean (m) as clarified in Algorithm 3.13.

Algorithm 3.13 Extract statistical features of HRV

```
Algorithm:  ECG_Statistical_features
Input:  HR signal dataset (Hr_dataset).
Output:  features_vec // statistical features vector
Begin:
Step 1: Extraction features process:
    • hr ← initial 1D array with zeros values
    • for hr in Hr_dataset:
    •     MeanNN ← calculate by Equation 2.1 for hr
    •     SDNN ← calculate by Equation 2.2 for hr
    •     RMSSD ← calculate by Equation 2.3 for hr
    •     PNN50 ← calculate by Equation 2.4 for hr
    •     NN50_count ← calculate by Equation 2.5 for hr
    •     meanHR ← calculate by Equation 2.7 for hr
    •     SDHR ← calculate by Equation 2.8 for hr
    •     HRVI ← calculate by Equation 2.9 for hr
    •     TINN ← calculate by Equation 2.10 for hr
    • Compute other statistical features:
    •     $hm \leftarrow \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$, (n represents length of hr)
    •     $med \leftarrow \begin{cases} hr[\frac{n+1}{2}] \\ \frac{hr[\frac{n}{2}] + hr\,[\frac{n}{2}+1]}{2} \end{cases}$ ( n represents length of hr)
    •     mod ←Compute the mode of hr
    •     m← calculate by Equation 2.13 for hr
    • end for hr
    • features_vec ← combines all previous statistical features in a
      vector.
Step 2: return features_vec
End Algorithm
```

Second, the CNN technique is used; before using it, the signal must be converted from the HR signal to the image by using the spectrogram method. The utilization of this technique facilitates the conversion of the electrocardiogram (ECG) waveform in the time domain into a two-dimensional (2D) visual representation, enabling novel viewpoints and an enhanced understanding of cardiac function. Figure 3.9 shows several images samples that produce from spectrogram method. Convert the HR signal to an image by using spectrogram method and then, the CNN trained model used for extraction of the abstract features. The SVD method used to reduce the feature maps that are produced from the flattened layer in the CNN trained model. Finally, combine the features that are produced by statistical methods and the CNN trained model. And then use them as inputs to the classifier. The following stage explains the type of classifier that is used with this model.

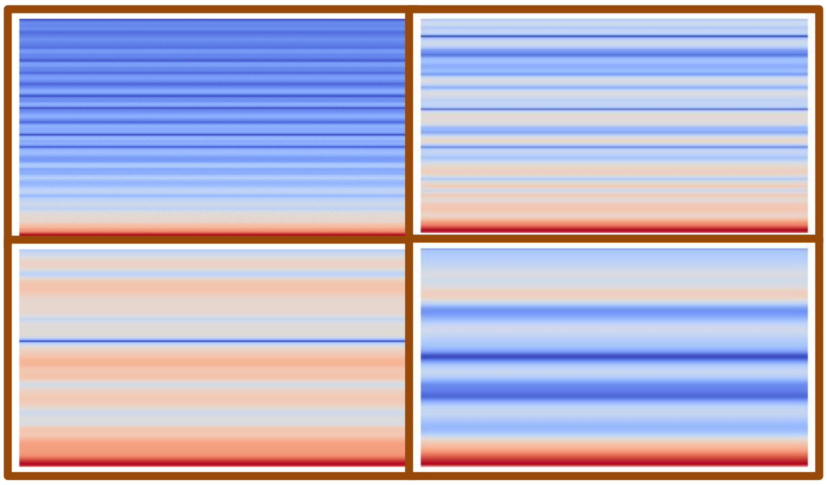


Figure 3.9 Samples of spectrogram images that result from HR signal

3.5.3 The Classification Stage of the ECG-CNN Model

In this stage, uses the DNN classifier for the classification of the DM. This classifier consists of two inputs features that come from statistical methods and a trained CNN model, and four blocks, each block has the following layers: (i) Fully Connection (FC) layer with Relu activation function, (ii) batch-normalization layer, (iii) and Gaussian dropout layer. Finally, there is one output dense layer. Figure 3.10 shows the number of nodes in each layer and the number of parameters in the DNN classifier. Table 3.5 describes the structure of the DNN classifier with the ECG-CNN model.

Table 3.5 Summary architecture of DNN for the ECG-CNN model

Layer name	Output Shape	No. Parameters
input_layer1	(None,13)	0
input_layer2	(None,512)	0
Concate_layer	(None,525)	0
first_layer (Dense)	(None,32)	16,832
batch_normalization	(None, 32)	128
gaussian_dropout	(None, 32)	0
second_layer (Dense)	(None, 32)	1056
batch_normalization_1	(None, 32)	128
gaussian_dropout_1	(None,32)	0
Third_layer (Dense)	(None,32)	1056
batch_normalization_2	(None,32)	128
gaussian_dropout_2	(None,32)	0
Forth_layer (Dense)	(None,32)	1056
batch_normalization_3	(None,32)	128
gaussian_dropout_3	(None,32)	0
output_layer (Dense)	(None, 1)	33

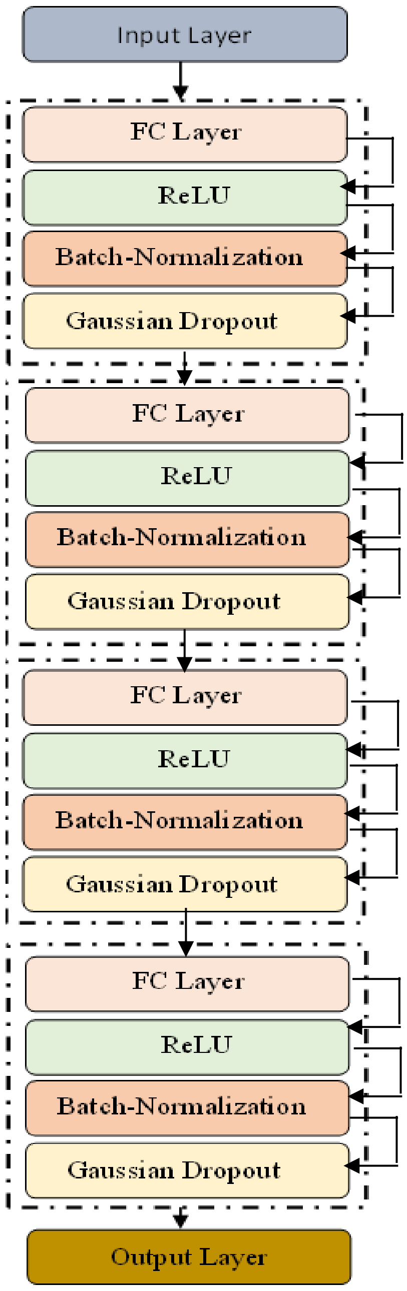


Figure 3.10 The architecture of the DNN classifier part of ECG-CNN model

3.6 The CLAHE-CNN Model Methodology

This model has been proposed to deal with image data in the proposed system. The proposed model has predicted the RD as normal or abnormal based on DL techniques. It consists of several steps including a preprocessing step (image resize, image data augmentation, image enhancement, normalization), a feature extraction step that considers the important step, based on the CNN model with DNN classifier for classification of the DR normal or abnormal as shown in Figure 3.11.

Algorithm 3.14 explains the steps of the CLAHE-CNN model for classifying the DR.

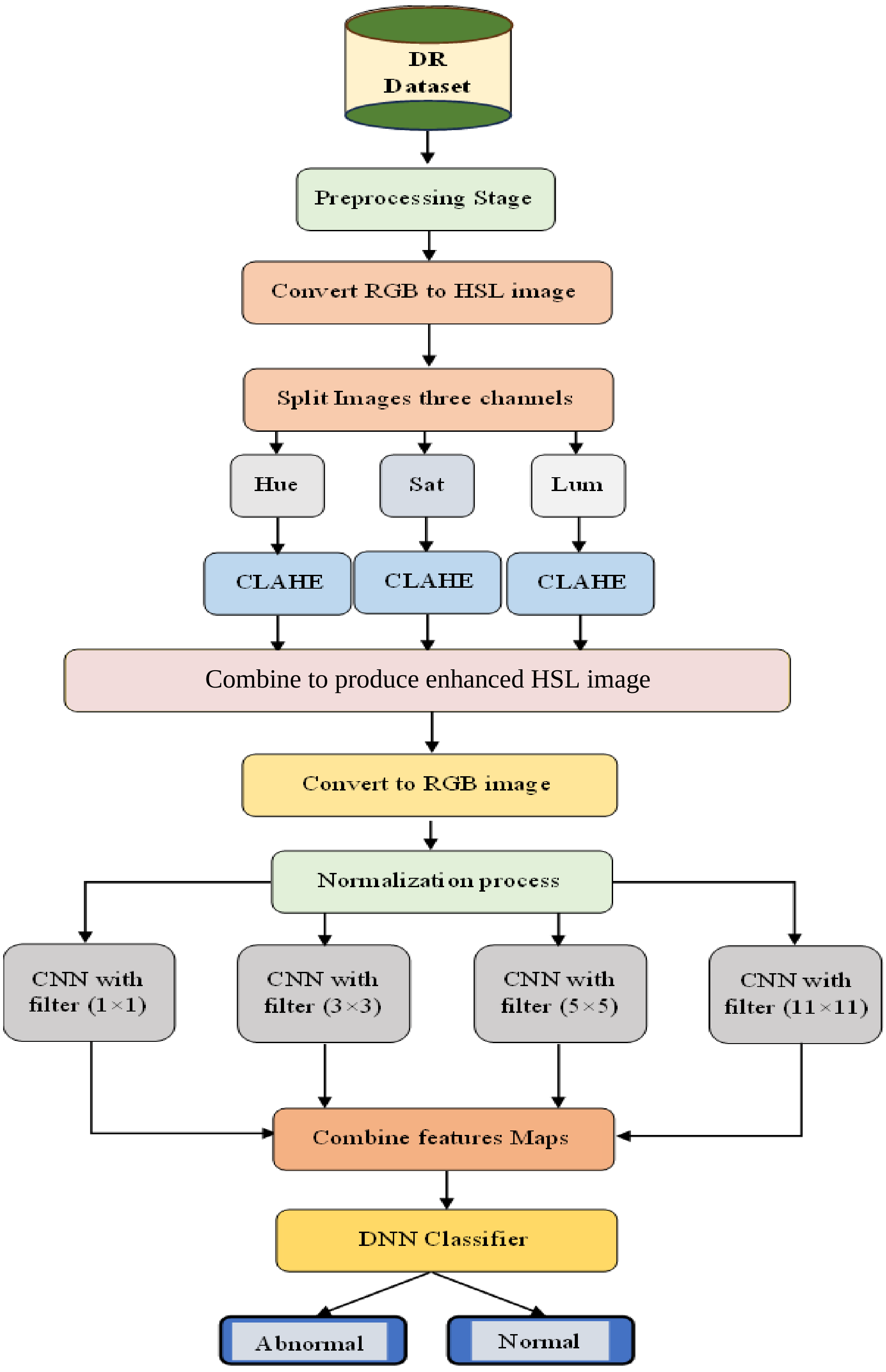


Figure 3.11 The diagram of the CLAHE-CNN Model

Algorithm 3.14 The CLAHE-CNN algorithm

```
Algorithm: CLAHE-CNN model
Input: Retina dataset (RD). epochs (E), batch size (BZ), learning rate
$\alpha = 0.001$, dropout probability ($p = 0.5$).
Output: CLAHE-CNN /* model with best parameters*/
Begin:
Step 1: Preprocessing stage:
    • Shuffled the RD.
    • Resizing images
    • Training set (Tr_X), (Tr_Y), Validation set (Val_X) and
      (Val_Y) and Testing set (Ts_X) and (Ts_Y).
Step 2: Image enchantment:
    • Apply CLAHE algorithm 3.15 on (Tr_X, Val_X, and Ts_X)
    • Data Augmentation for Tr_X:
        o Rotation by ($45^o$)
        o Zooming
        o Shearing
        o Flipping
    • Normalization process divided by 255.
Step 3: Training Process:
    • for i←1 to E do
    •  for Batch_img in Tr_X do
    •     Forward process:
    •       Feature Selection part
    •          Apply CNN part in figure 3.13
    •       Classifier part:
    •          Apply DNN classifier layers in Table 3.6
    •       Compute the loss and accuracy
    •       Validation process
    •     Backward propagation
    •          Update parameters (weights, biases)
    •  end for batch
    • end for i
Step 4: Evaluation Process
Step 5: CLAHE_CNN ← save the model with the best parameters.
Step 6: return CLAHE-CNN
End Algorithm
```

The CLAHE-CNN model includes several stages that will be listed as follows:

3.6.1 The Image Data Preprocessing of the CLAHE-CNN Model

In this stage, the following preprocessing techniques will be used to prepare the image data for the classification process: (i) image resizing, (ii) Image enhancement, and (iii) image data augmentation.

Image resizing and data augmentation are important techniques for increasing the performance of the DL model. These techniques have been mentioned previously.

After, apply preprocessing stage the enhancement stage is started. In this stage, has used the CLAHE method to enhance the contrast on an image. Firstly, convert the RGB original image into image with HSL color system (H= “hue”, S = “saturation”, and L= “lightness”). Secondly, take the H, S, and L channels and apply the CLAHE method to them. Finally, combine the output from the CLAHE method and convert it to an RGB image. Algorithm 3.15 explains the CLAHE method.

Algorithm 3.15 CLAHE algorithm

```
Algorithm: CLAHE method
Input:   input image(image_gray) with size (N, M), clip_limit=3.
Output:  enhanced_img // image after enchanced.
Begin:
Step 1: Initial 2D array enhanced_img
Step 2: Enhancement process:
  • for i ← 0 to (M-8) do step 8
  •     for j ←0 to (N-8) do step 8
  •         for k ← i to i+8 do
  •             for r ← j to j+8 do
  •                 sub_image ← image [k, j]
  •             End for r
  •         End for k
  •         sub_histogram←calculate the histogram of (sub_imge)
  •     clipped_histogram ← clip_histogram (sub_histogram,
  •                                          clip_limit)
  •     cdf ←compute CDF of clipped_histogram // Equation 2.17
  •     sub_result← interpolation(sub_image, cdf with range 255)
  •     for k ← i to i+8 do
  •         for r ← j to j+8 do
  •             enhanced_img ← sub_result [k,j]
  •         End for r
  •     End for k
Step 2: return enhanced_img
End Algorithm
```

3.6.2 Feature Extraction Stage of the CLAHE-CNN Model

This stage is considered an important stage, using parallel four CNN blocks with different kernel sizes. Each block contains of the convolution layer with (16) units, stride (1), padding, and kernel size = (1×1, 3×3, 5×5, or 11×11). After the convolution layer comes the batch-normalization layer and Gaussian dropping layer with (0.5 probability). After, applying the CNNs with different kernels, the features map that results the CNNs combines to generate a new features map, that uses it as input to the DNN classifier.

3.6.3 The Classification Stage of the CLAHE-CNN Model

In this stage, the DNN classifier is used to predict the RD. After the image dataset passes over the preprocessing stage and enhancement stage, turn comes to proposed CNN model, contains two parts feature extraction classification part. After extracting the feature maps, will be entered into the DNN classifier. DNN classifier contains an input layer and three blocks of hidden layer, each block involves an FC layer with Relu AF, and batch normalization layer, and a dropout layer. Finally, the output layer with sigmoid AF. Table 3.6 describes the structure of the DNN classifier with the CLAHE-CNN model.

Table 3.6 Summary of DNN architecture of CLAHE-CNN model

Layer name	Output Shape	No. Parameters
Input layer	(None, 258944)	0
Dense1	(None, 128)	33144960
batch_normalization	(None, 128)	512
dropout	(None, 128)	0
Dense2	(None, 128)	16512
batch_normalization	(None, 128)	512
dropout	(None, 128)	0
Dense3	(None, 128)	16512
batch_normalization	(None, 128)	512
dropout	(None, 128)	0
Output layer	(None, 1)	129

3.7 The Proposed System

In this section, the proposed system can be divided into two phases. First, the data type source has entered and employed the inductor function to determine which model the data source will pass to it. Second, aggregation process, the decisions that produce from the models are aggregated to determine the final decision of the system. Algorithm 3.16 explains the proposed system.

Algorithm 3.16 The proposed system algorithm

```
Algorithm: Proposed System algorithm
Input: four arrguments (Tabular data: tab_data, ECG data:
ECG_data, Diabetes Foot Ulcer image data: DFU_image data and
Diabetes Retina image data: DR_image data)
Output: final_decision. /* It can be any class ( none  = 0, diabetic=1,
diabetic with DFU=2, diabetic with RD=3, diabetic with DFU and
RD=4).
Begin:
Step 1: Initialize 1D array of the decisions with none value:
        • D ← [none, none, none, none]
Step 2: Apply the inductor function:
        • If (tab_data! = none) then
```

```
    • D [0]← apply a Block of CAER-DNN (tab_data).
    • If (ECG_data! = none) then
    •     D [1]← apply a Block of ECG-DNN (ECG_data).
    • If (DFU_image! = none) then
    •     D [3]← apply a Block of GLCM-CNN (DFU_image).
    • If (DR_image! = none) then
    •     D [2]← apply a Block of CLAHE-CNN (DR_image).
Step 3: final_decision ← Aggregation (D) // apply algorithm 3.17
Step 4: return final_decision
End Algorithm
```

The aggregation stage is used to produce the final decision of the proposed system, which represents five classes (diabetic with DFU, diabetic with RD, diabetic with DFU, RD, and none). The input of this stage is a decision vector with four values, each of which denotes the decision of a specific model. This vector can have any value (0, 1, or "none"), where the values (0, 1) refer to normal cases and abnormal cases of disease or its complications, respectively. The "none" value means the data type source is unavailable, so the model of this data does not give the decision value. Finally, the proposed system aims to aggregate multiple decisions into a final decision label based on specific conditions. It considers the presence or absence of decisions at different positions in the input array as clarified in Algorithm 3.17.

Algorithm 3.17 The Decisions Aggregation algorithm

```
Algorithm: Aggregation function
Input: An array of decisions with a size of (4×1)
/* Examples:
decisions = [tab_data, none, none, none]
decisions = [tab_data, ecg_data, none, none] */
Output: label /* Represents the final decision */
Begin:
Step 1: Decision Cases:
    o The algorithm checks different cases based on the presence of
      decisions in the input array.
```

- *Each case corresponds to a specific combination of none decisions.*

***Step 2:** Decision Logic:*

- *For each case, the algorithm assigns a label based on the conditions specified.*
- *If a certain condition is met, the label is set accordingly (e.g., label ← 1, label ← 2, label ← 3, or label ← 4).*
- *If no condition is met, the default label is set to 0.*

***Step 3:** return label /* the final label is returned as the output of the algorithm */.*

End Algorithm

CHAPTER FOUR:

Results and Discussion

Results and Discussion

4.1 Introduction

This chapter evaluates the proposed system's efficiency, explained in the previous chapter, by using various parameter values. It presents and discusses the outcomes obtained. The proposed system's behavior has been evaluated through the application of a comprehensive dataset that encompasses tabular, image, and ECG datasets. This dataset has been utilized as a case study to gain insights into the system's performance. Moreover, this chapter presents a detailed description and presentation of the experimental results concerning the stages of the proposed system. The proposed system has been programmed by using Python programming language with PyCharm as an IDE. It is executed on Windows 10 and 11 operation system Processor Intel i7-6700 4.0 GHz, GPU is Nvidia 3070TI 8 GB, RAM size 16 GB, Storage is SSD 1 TB.

4.2 Datasets

The utilization of datasets encompassing various data types is most important for gaining a holistic understanding of the problem research. This section will explain the data set used in the proposed system, which includes tabular, ECG, and image datasets as follows:

4.2.1 Tabular Datasets

Two tabular datasets, Pima Indian Diabetes (PID) and Mendeley Diabetes (MDD), are used in this section and can be explained as follows:

4.2.1.1 Pima Indian diabetes dataset (PID)

PID is used from the Machine learning Repository dataset (UCI)[105]. It is considered one of the most important datasets for digamous the diabetic. It has a sample of 768 females with ages not less than 21 years old, where 500 samples belong to females who do not suffer from the disease and 268 samples suffer from diabetes. Each item of the dataset has 8 features as input and one class label as output. Table 4.1 briefly shows the details of the features.

Table 4.1 Describes the features of PID dataset

Features	Description	Type
Pregnancies	Pregnancies Number with range (0-17)	Numeric
Skin Thickness	thickness of triceps skin-fold in (mm) with range (0-99)	Numeric
Glucose	The two-hour plasma glucose level in an oral glucose tolerance test with range (0–199)	Numeric
Diastolic Blood pressure	Diastolic blood pressure gauges how much pressure is present in the arteries between heartbeats when the heart is rested. with range (0-122)	Numeric
BMI	Index of mass of the body (weight in kg/ power (height in m,2)) with range (0-67.1)	Numeric
Serum Insulin	two-hours serum insulin in (mu U/ml) with a range (0-846)	Numeric
Diabetes pedigree Function	An engaging feature that helps diagnose the diabetic with a range (0.078–2.42)	Numeric
Age	Patient age with range (21-81)	Numeric

According the Table 4.1 there are various features or attributes related to a PID dataset. Each feature is associated with a brief description and its data type. These features appear to be associated with a dataset used for diabetes-related classification. They include a variety of numeric measurements related to factors that could influence or be indicative of diabetes.

4.2.1.2 Mendeley diabetes dataset (MDD)

MDD is collected from the society of Iraq. It contains 1000 samples with three classes covered including diabetes patients, healthy, and predicted diabetes patients). The Medical City Hospital laboratory and the

Specializes Center for Endocrinology and Diabetes-Al-Kindy Teaching Hospital provided the data (MCHL). It has laboratory testing and medical data. The dataset features include Blood Sugar Levels (BSL), Ratio of Creatinine (CR), Age, sex, Body Mass Index (BMI), cholesterol are all factors (Chol), and lipid status during fasting, which includes VLDL, LDL, HDL, Triglycerides (TG), and a HBA1C. Table 4.2 shows briefly the details of attributes of the dataset.

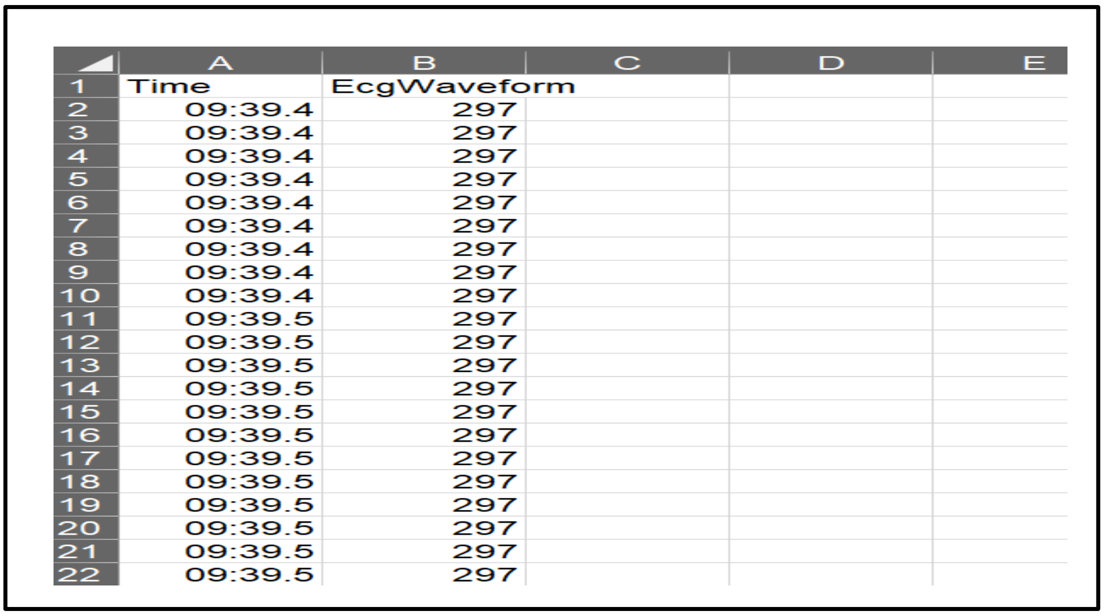
Table 4.2 Describes the features of MDD

Features	Description	Type
Urea	Urea in mg/dl, with range (0.5- 38.9)	Numeric
Gender	Male or Female	Categorical
Age	Patient age with range (20- 79)	Numeric
CR	In mol/l with range (48- 80)	Numeric
BMI	Index of mass of body (weight in kg/ power (Hight in m,2)) with range (19-47)	Numeric
LDL	In mmol/l with range (0.3-9.9)	Numeric
VLDL	In mmol/l with range (0.1- 35)	Numeric
HDL	In mmol/l with range (0.2- 9.9)	Numeric
Chol	In mmol/l with range (0.0- 10.3)	Numeric
TG	In mmol/l with range (0.3-13.8)	Numeric
HBA1C	In mmol/l with range (0.9- 16)	Numeric

The Table 4.2 offers descriptions of various features or attributes that appear to be associated with a diabetic dataset, likely used for analyzing and diagnosing health conditions. Each feature is described along with its data type and the range of values it can take. These features appear to be relevant for assessing the health status of patients, particularly in the context of conditions related to diabetes, given the presence of features like BMI, cholesterol levels, and HBA1C. The data types and ranges are essential for understanding the variables' characteristics and the potential insights that can be derived from their analysis.

4.2.2 ECG Signals Dataset (D1NAMO)

The D1NAMO dataset comprises two discrete groups: the first group consists of data obtained from 84 samples who are in good health, while the second subset consists of data obtained from 46 samples diagnosed with diabetes. The website of the dataset is "The open D1NAMO dataset: A multi-modal dataset for research on non-invasive type 1 diabetes management | Zenodo". Both groups consist of the following data: Electrocardiogram (ECG) signals, The topic of interest pertains to the phenomenon of breathing signals. The outputs of the accelerometer, the measurement of glucose levels varies according to the specific subpopulation being studied, and Photographs of food accompanied by explanatory notes provided by a registered dietitian. In this dissertation, using Electrocardiogram (ECG) signals for classification of the DM. Figure 4.1 shows the sample of the ECG data in D1NAMO dataset.



	A	B	C	D	E
1	Time	EcgWaveform			
2	09:39.4	297			
3	09:39.4	297			
4	09:39.4	297			
5	09:39.4	297			
6	09:39.4	297			
7	09:39.4	297			
8	09:39.4	297			
9	09:39.4	297			
10	09:39.4	297			
11	09:39.5	297			
12	09:39.5	297			
13	09:39.5	297			
14	09:39.5	297			
15	09:39.5	297			
16	09:39.5	297			
17	09:39.5	297			
18	09:39.5	297			
19	09:39.5	297			
20	09:39.5	297			
21	09:39.5	297			
22	09:39.5	297			

Figure 4.1 Sample of ECG in D1NAMO dataset

According to Figure 4.1, typically, the ECG sample from the D1NAMO dataset consists of two columns: time and ECG waveform. The time column denotes the temporal aspect or duration of each data point, often measured in millisecond. The column labeled ECG waveform provides the precise recorded values of the electrocardiogram (ECG) signal at the appropriate time points. The ECG waveform column contains the amplitude or voltage values of the ECG signal at various time points. This column enables the visualization of the heart's electrical activity and the detection of patterns or abnormalities in the ECG data.

4.2.3 Diabetes Ulcer Foot Dataset (DFU)

This section includes two steps: (i) collecting the dataset, and (ii) labeling the dataset. The DFU datasets are provided by[18]. First, a colored-images dataset of diabetic foot ulceration has been collected from various patients. It consists of 756 images of infected feet suffering from diabetic foot ulcer disease and healthy skin without disease, which were collected from the diabetic center of Nasiriyah’s Hospital located in southern Iraq. It has moral endorsement and composed assent from all pertinent people and patients. The images were taken using a Galaxy Note 8 and an iPad with varying brightness and viewpoints, whereby the identity has been de-identified for all images that were collected and will be overseen (taking after other related approaches).

Second, the areas or regions of interest are initially cut to a size of 224 x 224 pixels. This region could be a salient region surrounding the ulcer that involves vital tissues of both skin types, including normal and abnormal. Next, a specialist doctor labeled the regions that were cut into two ground-truth labels, namely normal and abnormal skin. Finally, a totalof 1609 skin patches were collected, with 234 normal and 1067 abnormal (e.g., DFU). Figure 4.2 shows some normal and abnormal samples.

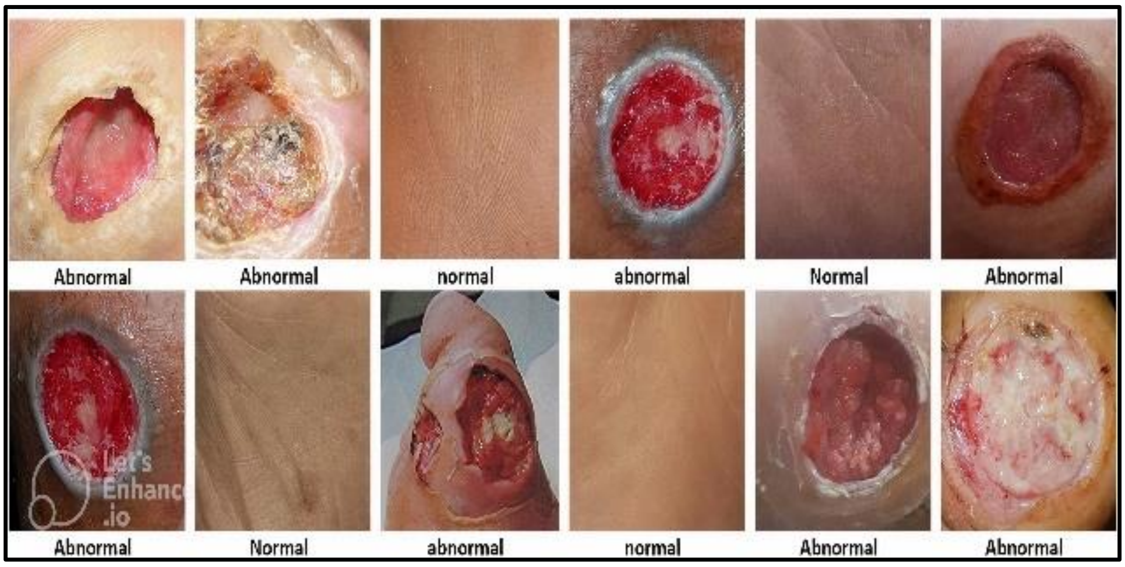


Figure 4.2 Samples of normal and abnormal images in DFU dataset

4.2.4 Diabetic Retinopathy Dataset

The Messidor datasets are provided by [106]. First, a colored-images dataset of diabetic Retinopathy has been collected from various patients. It consists of (1017) images of infected eyes suffering from diabetic Retinopathy disease and (728) healthy images without disease. Initially, DR images are preprocessed by deleting the black background to become more proper. These datasets have binary and multiple classes, in this dissertation, has used binary classification. Figure 4.3 shows samples of DR images.

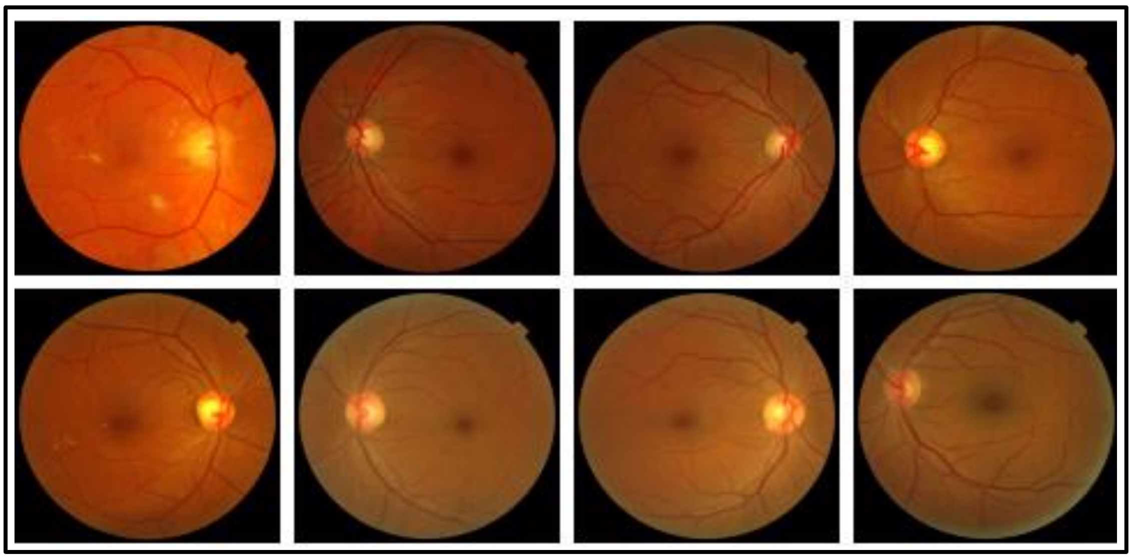


Figure 4.3 Samples of the DR dataset

4.3 Evaluate of CAER-DNN model

In this section, the classification as a problem related to diabetes was posed for patients who provided some medical information. Two datasets are used in this section for evaluating the CAER-DNN model.

4.3.1 Testing the Performance of the Autoencoder Model

Calculation of the reconstruction error for the autoencoder is required. This metric quantifies the ability of the autoencoder to accurately reproduce the initial input data based on the encoded representation, also known as the latent space. A lower reconstruction error typically signifies superior feature learning. Metrics such as Mean Squared Error (MSE) can be employed to measure the reconstruction error. A decrease in error is indicative of an improved feature representation. The performance of this model will be tested depending on hyperparameters: Number of epochs (200) and Batch size (32).

Table 4.3 Testing the performance of the autoencoder based on different hyperparameters

Cases	AF	Optimizer	Learning Rate	MSE
Case 1	**Linear**	**Adam**	**0.001**	**0.002157**
Case 2	ReLU	Adam	0.001	0.006775
Case 3	Linear	SGD	0.001	0.07168
Case 4	ReLU	SGD	0.001	0.03386
Case 5	Linear	Adam	schedule	0.008764
Case 6	Linear	SGD	schedule	0.00937

Table 4.4 Testing the performance of the autoencoder based on different Learning Rate values

Cases	Learning Rate	Epochs	MSE
Case 1	0.1	200	0.004412
Case 2	0.01	200	0.00228
Case 3	**0.001**	**200**	**0.002157**
Case 4	0.0001	200	0.005
Case 5	0.00001	200	0.048

Table 4.3 presents different experimental cases with varying combinations of activation functions, optimization algorithms, and the presence or absence of learning rate schedules. The training losses in each case indicate how well the neural network models were able to fit the training data. Choosing the right combination of these factors is essential for achieving optimal model performance in neural network applications. The result refers to case 1, which is the best MSE value achieved when using a constant Learning rate of 0.001, linear AF, and the Adam optimizer. Using a constant value, the Learning Rate achieves a better result than using a schedule.

Table 4.4 illustrates the impact of different learning rates on model training. The learning rate is a critical hyperparameter in machine learning, as it determines the step size during optimization. The choice of an appropriate learning rate depends on the specific problem and the behavior of the optimization algorithm. A learning rate that is too high can lead to overshooting, while one that is too low can result in slow convergence or getting stuck in suboptimal solutions. Experimenting with different learning rates is a common practice in hyperparameter tuning to find the best balance between fast convergence and perfect performance. The results of the performance of the model when using different constant learning rate, linear activation function, and Adam optimizer. The best value of the Learning Rate is 0.001, which produces a good MSE equal to 0.000215.

4.3.2 Testing the Performance of the CAER-DNN Model

In this dissertation, results were carried out by utilizing the CAER-DNN model. The performance of our model with two datasets (MDD and PID) datasets.

Table 4.5 Evaluating the CAER-DNN and DNN model with different datasets including PID and MDD datasets

Model with dataset	Accuracy	Precision	Recall	F1-Score
CAER-DNN with MDD	**98.48%**	98.21%	91.74%	94.51%
CAER-DNN with PID	**98.90%**	98.14%	98.66%	98.39%
DNN with PID	96.61%	97.10%	95.77%	96.40%

In Table 4.5 the CAER-DNN model with PID appears to outperform the CAER-DNN model with MDD in terms of accuracy, recall, and f1-score while the model with MDD outperforms in precision metric. The use of autoencoder technology also plays a major role in increasing the accuracy of the model due to its role in reconstructing the features and selecting the most important ones. Therefore, the model CAER-DNN outperforms the model DNN.

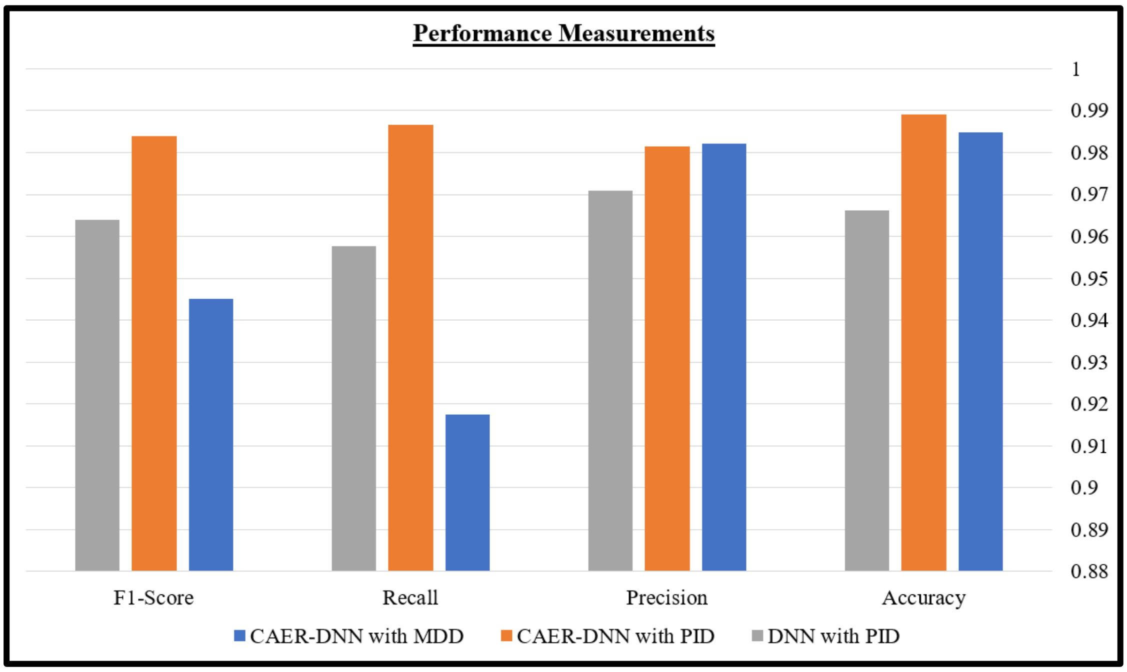


Figure 4.4 Comparison between CAER-DNN and DNN model based on two Datasets including PID and MDD datasets

Figure 4.4 shows how the CAER-DNN model has outperformed the DNN model in all measurements (f1-score, recall, precision, and accuracy). This is due to using important techniques called autoencoders to reconstruct the original features and select the best by regularization technique. Still, it takes a long time in the training process compared with using the DNN model alone.

Table 4.6 Comparison of the CAER-DNN model with state-of-the-art models for classification DM

Models	**Accuracy**	**Specificity**	**Precision**	**Recall**	**F1-Score**
CART-GA [11]	93.42%	88.89%	94.00%	95.92%	94.95%
ANN-GA[11]	81.82%	69.70%	80.00%	90.91%	85.11%
ANN [12]	90.00%	-	-	-	-
Ensemble model [14]	83.08%	-	-	-	-
RF with all features [15]	79.57%	75.00%	89.40%	81.33%	85.17%
J48 decision tree[15]	74.78%	59.63%	70.86%	88.43%	78.68%
NB[15]	78.67%	63.29%	81.88%	86.75%	84.24%
NB with feature selection three factors [15]	79.13%	62.03%	81.60%	88.08%	84.71%
NB with feature selection five factors [15]	77.83%	62.03%	81.25%	86.09%	83.60%
QML[13]	86.00%	86.00%	74.00%	85.00%	79.00%
DL[13]	95.00%	95.00%	90.00%	95.00%	93.00%
DL[107]	98.07%	-	95.22%	98.46%	96.81%
ANFIS[16]	92.00%	84.00%	-	97.00%	-
Our model CAER-DNN	**98.90%**	**99.00%**	**98.14%**	**98.66%**	**98.39%**

Table 4.6 shows how the CAER-DNN model with the PID dataset outperforms all existing state-of-the-arts that mentions in above table in terms of performance measures, which has achieved the highest scores in accuracy, specificity, precision, recall, and f1-score with (98.90, 99.00, 98.14, 98.66, and 98.39) respectively. The authors in [107] used the DL

technique but achieved a lower f1-score, recall, precision, and accuracy (96.81, 98.46, 95.22, 98.07) compared with the CAER-DNN model due to the CAER-DNN model not using only a DNN that has multiple layers but utilizing an autoencoder with regularization techniques to regenerate original features and selecting the best, which enhances the performance of the DNN by adding the encoder part as an additional block layer to DNN. It has been noted that DL technologies have outperformed ML techniques due to several causes. First, DL has multiple layers that automatically learn hierarchical representations of data. This allows the model to extract sophisticated and meaningful features from original input data by employing numerous layers of abstraction, compared with traditional ML that relies on handcrafted features. Second, DNN possess a significant number of parameters that can be tuned throughout the training process. The adaptability of deep learning models enables them to adjust to complex patterns in the data, compared with traditional ML that based on may have fewer parameters, making them less flexible in capturing complex relationships.

In Figure 4.5 shows the outperformance of the proposed model (CAER-DNN) compared with thirteen state-of-the-arts in term of accuracy, specificity, precision, recall, and f1-score. It has been noted that the closest model in terms of performance is DL model [13], [107] , and here the role of deep learning is magnified compared to traditional methods of machine learning. Furthermore, the J48 decision tree model in [15] presented the lowest results in terms of performance compared to the rest of the models.

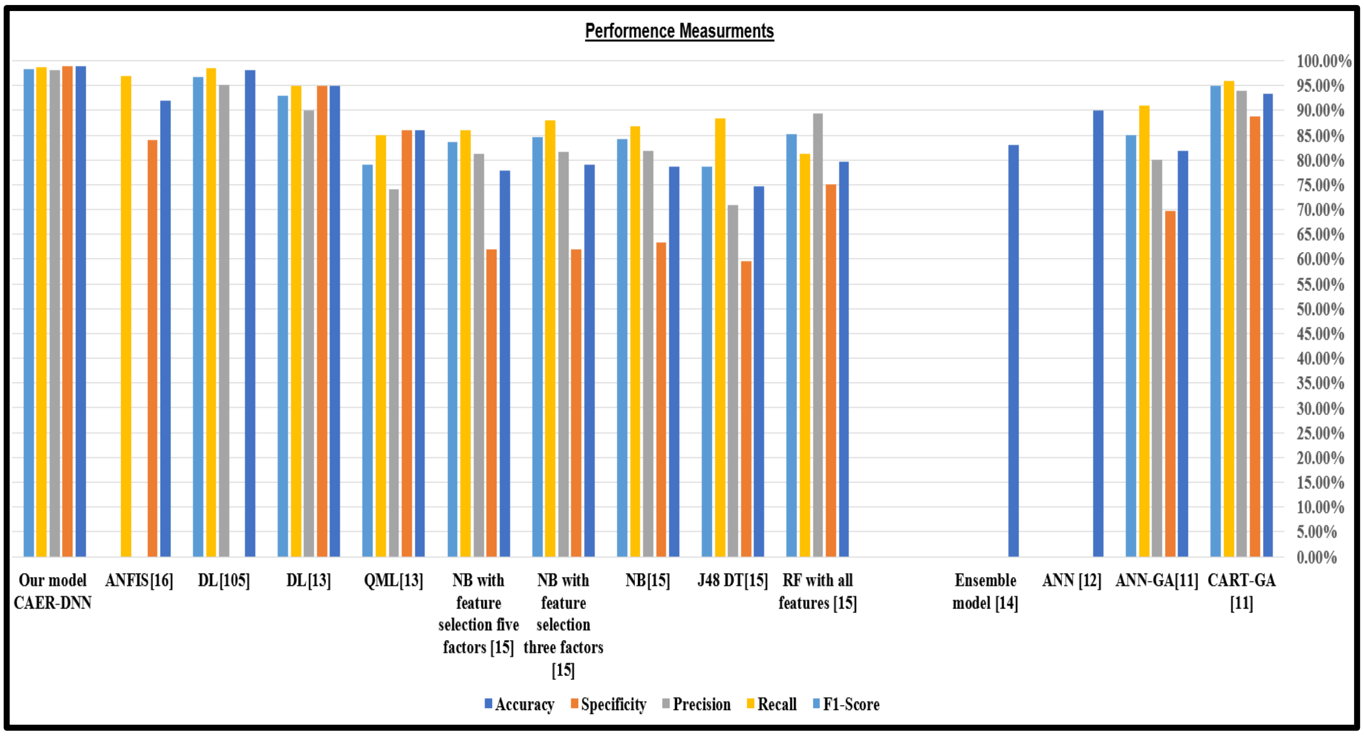


Figure 4.5 Comparison of our results with thirteen state-of-the-arts through measuring of the performance of them in term of accuracy, specificity, precision, recall, f1-score based on PID dataset

Table 4.7 Evaluating the CAER-DNN model with ML models based on MDD dataset

Model with dataset	Accuracy	Accuracy Balance[108]	Precision	Recall	F1-core
MLG[108]	86.70%	78.10%	70.00%	70.00%	70.00%
DT [108]	95.07%	82.58%	98.12%	74.62%	89.67%
RF[108]	90.64%	84.45%	75.00%	78.00%	76.40%
SGB[108]	97.04%	88.00%	**98.85%**	81.00%	89.00%
NB[108]	93.10%	80.40%	89.00%	71.86%	79.50%
Our model CAER-DNN	**98.48%**	**98.21%**	**98.21%**	**91.74%**	**94.51%**

In Table 4.7 The CAER-DNN model has outperformed all models, which included Multinomial Logistic Regression (MLG), DT, RF, Stochastic, Gradient Boosting (SGB), and NB as shown in Figure 4.6. The performance of the proposed model outperformed ML models in all evaluation measures including accuracy, accuracy Balance, recall, and f1-

score, except precision measure with the SGB model that achieved a high score compared with CAER-DNN model with score equal to (98.85%). Figure 4.6 shows how the CAER-DNN model outperformed ML models with MDD dataset.

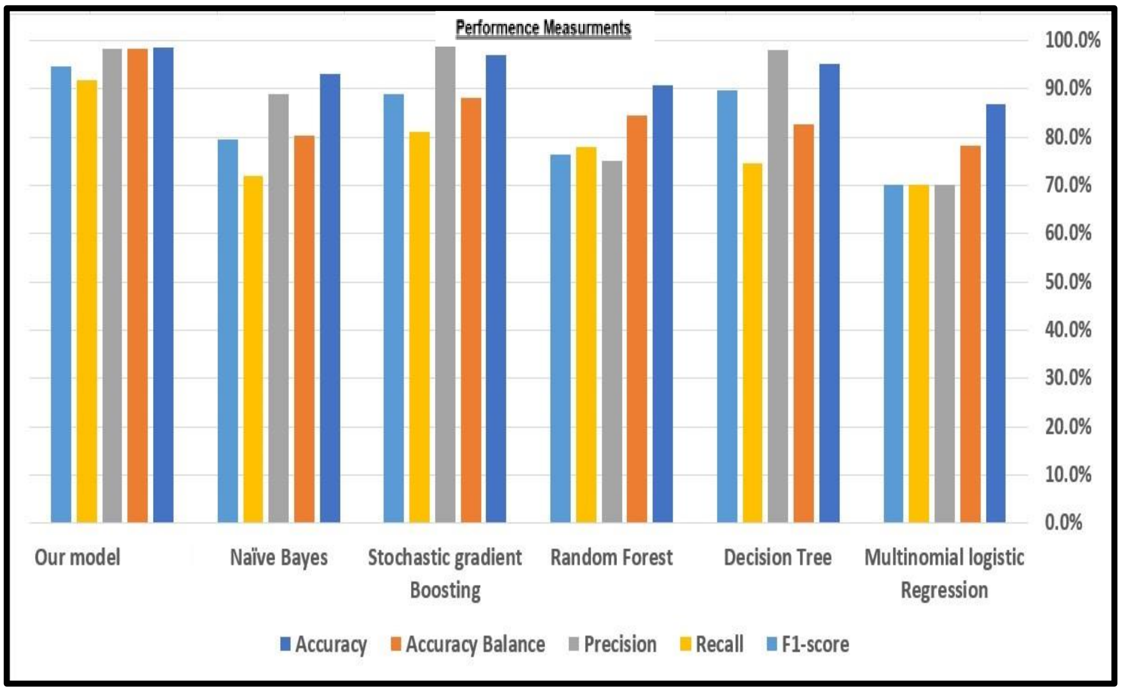


Figure 4.6 Comparison of our results with five ML techniques through measuring of the performance of them in term of accuracy, accuracy balance, precision, recall, f1-score based on MDD dataset

4.3.3 Estimated Executing Time of CAER-DNN Model

Testing the execution time of model is considered the most important feature, which refers to the quality model. In this section, the CAER-DNN model has tested with PID dataset. The execution time is presented in Table 4.8, based on the testing dataset.

Table 4.8 Execution time of the CAER-DNN model

CAER time (sec)	DNN time (sec)	Total time (sec)
0.0960	0.0670	0.163

In Table 4.8 shows the execution time of CAER-DNN model with its stages. The CAER stage takes 0.0960 s, the DNN stage takes 0.0670 s and the total time of the CAER-DNN takes 0.1630 s. So that, the model achieves good execution time.

4.4 Evaluate of ECG-CNN model

In this section, apply the ECG-CNN model on testing set of ECG dataset, and then evaluate it using the performance metrics of SL and calculate the execution time.

4.4.1 Testing the ECG-CNN Model

Initially, apply the proposed (ECG-CNN) model to ECG signals after converting them to HR signals by the Pan-Tompkins method, which extracts the peaks of the ECG signal that represent the HR signal. Next, extract two types of features, including statistical and abstract features, by using statistical tools and a CNN-trained model. Finally, apply the DNN classifier to classify the ECG into healthy and unhealthy classes. The performance of the ECG-CNN model can be shown in Table 4.9.

Table 4.9 Evaluating the ECG-CNN model and four different models with different datasets

Model	Accuracy	Specificity	Precision	Recall	F1-Score
CNN-LSTM with SVM [25]	95.70%	-	-	-	-
SVM [27]	90.50%	-	-	-	-
ANN [27]	86.30%	-	-	-	-
FFNN [26]	93.08%	96.92%	-	-	-
ECG-CNN	**99.88%**	**-**	**100%**	**99.98%**	**99.9%**

According to Table 4.9 the ECG-CNN model has compared with four models including (CNN-LSTM with SVM, SVM, ANN, FFNN). The proposed model has achieved higher accuracy with score (99.88) compare with four models. This is due to using two techniques to extract important features, including statistical methods and a CNN trained model to produce abstract features by converting the HR signal to an image using the spectrogram method, which was mentioned previously in Chapter 2. Next, combine these features and use the DNN classifier to classify healthy and unhealthy classes of DM. The strength of the model lies in relying on the integration of two technologies to obtain important features. This is due to using two techniques to extract important features, including statistical methods and a CNN-trained model to produce abstract features by converting the HR signal to an image using the spectrogram method, which was mentioned previously in Chapter 2. Next, combine these features and use the DNN classifier to classify healthy and unhealthy classes of DM. The strength of the model lies in relying on the integration of two technologies to obtain important features.

Table 4.10 Evaluating the ECG-CNN model and five ML techniques based on ECG signals in D1NAMO dataset

Model	Accuracy	Precision	Recall	F1-Score
KNN	92.3%	100%	75.0%	91.9%
DT	61.5%	42.8%	75.0%	62.9%
RF	76.9%	60.0%	75.5%	77.5%
SVM	92.3%	100%	75.0%	91.9%
LR	84.6%	75.0%	75.0%	84.6%
ECG-CNN	99.88%	100%	99.98%	99.9%

This is due to using two techniques to extract important features, including statistical methods and a CNN trained model to produce abstract features by converting the HR signal to an image using the spectrogram method, which was mentioned previously in Chapter 2. Next, combine these features and use the DNN classifier to classify healthy and unhealthy classes of DM. The strength of the model lies in relying on the integration of two technologies to obtain important features. This is due to using two techniques to extract important features, including statistical methods and a CNN-trained model to produce abstract features by converting the HR signal to an image using the spectrogram method, which was mentioned previously in Chapter 2. Next, combine these features and use the DNN classifier to classify healthy and unhealthy classes of DM. The strength of the model lies in relying on the integration of two technologies to obtain important features. This is due to using two techniques to extract important features, including statistical methods and a CNN trained model to produce abstract features by converting the HR signal to an image using the spectrogram method, which was mentioned previously in Chapter 2. Next, combine these features and use the DNN classifier to classify healthy and unhealthy classes of DM. The strength of the model lies in relying on the integration of two technologies to obtain important features. This is due to using two techniques to extract important features, including statistical methods and a CNN-trained model to produce abstract features by converting the HR signal to an image using the spectrogram method, which was mentioned previously in Chapter 2. Next, combine these features and use the DNN classifier to classify healthy and unhealthy classes of DM. The strength of the model lies in relying on the integration of two technologies to obtain important features.

Table 4.10Table 4.10 shows, the ECG-CNN model has outperformed machine learning techniques including KNN, DT, RF, SVM, and LR in all metrics. Among the traditional machine learning models, SVM and KNN also show strong performance in terms of accuracy and precision but our model has achieved high score in all measures. Figure 4.7 shows the performance of the ECG-CNN model compared with other machine learning models and previous studies.

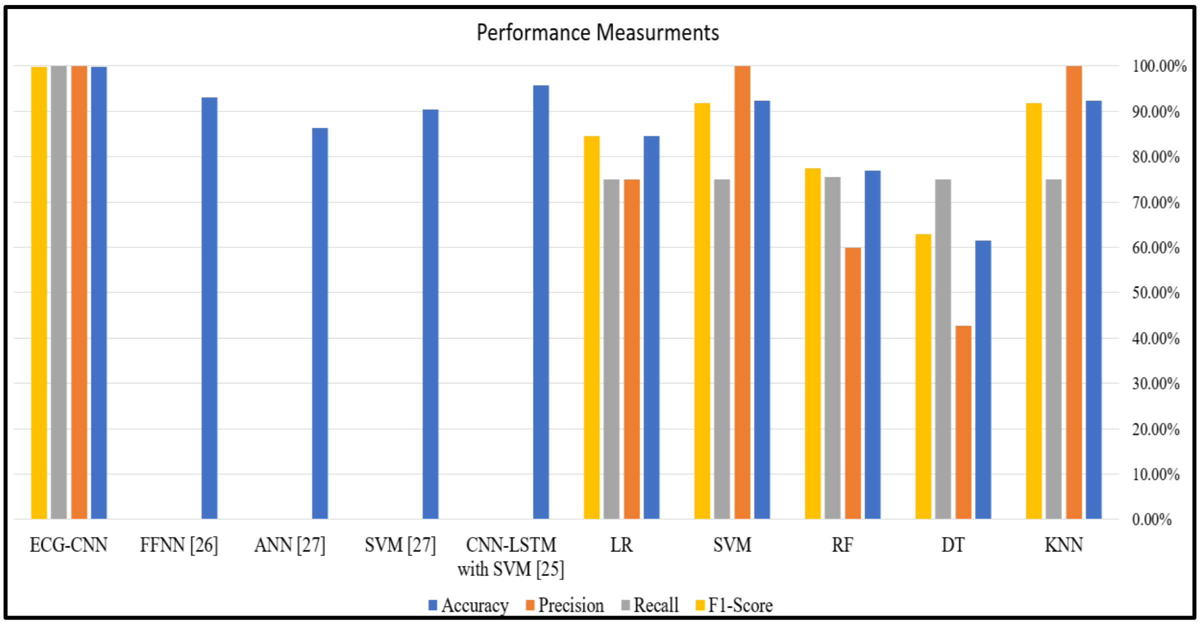


Figure 4.7 Comparison of our results with nine DL and ML techniques through measuring of the performance of them in term of accuracy, accuracy balance, precision, recall, f1-score based on D1NAMO dataset

4.4.2 Estimated Executing Time of ECG-CNN Model

Table 4.11 shows the execution time of the ECG-CNN model and each stage of it, based on the testing dataset.

Table 4.11 Execution time of the ECG-CNN model

Framing time (sec)	Convert to image time (sec)	DNN time (sec)	Total time (sec)
0.293	0.130	0.1290	0.552

In Table 4.11 shows the execution time of ECG-CNN model with its stages. The Framing stage takes 0.293s, the Convert to image stage takes 0.130s and DNN takes 0.1290s. Finally, the total time of the ECG-CNN takes 0.552s. So that, the model achieves less execution time.

4.5 Evaluate of GLCM-CNN Model

In this section, explain the DFU dataset that was applied to the GLCM-CNN model and then evaluate it using the performance metrics of SL and calculate the execution time. Then, evaluate the model for clustering the state of an abnormal ulcer foot by the USL metric called the average silhouette score.

4.5.1 Testing the Performance of the GLCM-CNN Model

The performance of the model using different classifiers, DNN and SVM, is presented in Table 4.12.

Table 4.12 The GLCM-CNN model with DNN and SVM classifier

The Model	Accuracy	Specificity	Precision	Recall	F1-Score
CNN_GLCMNet+DNN	**97.4%**	97.5%	97.5%	97.2%	97.3%
CNN_GLCMNet+SVM	96.9%	96.9%	96.7%	96.9%	96.8%

In the Table 4.12 have noted the model with a DNN classifier obtained results that are higher for all measurements, as compared to the model with an SVM classifier. The "CNN_GLCMNet+DNN" model had slightly better performance in terms of accuracy, specificity, precision, recall, and F1-Score. However, the model with SVM has fewer training parameters than the DNN classifier.

Table 4.13 Evaluating the CLCM-CNN model and ten DL techniques

Model	Accuracy	Specificity	Precision	Recall	F1-Score
Hybrid DCNN with four branches [17]	-	-	97.3%	94.5%	95.8%
Hybrid DCNN with five branches [17]	-	-	96.5%	94.2%	95.3%
Hybrid DCNN with three branches[17]	-	-	94.7%	92.9%	93.7%
Hybrid DCNN with two branches [17]	-	-	93.6%	90.7%	92.1%
DFU_QUTNet [18]	-	-	94.2%	92.6%	93.4%
DFU_QUTNet+KNN [18]	-	-	93.8%	92.7%	93.2%
DFU_QUTNet+SVM [18]	-	-	95.4%	93.6%	94.5%
DFU_SPNet [21]	96.4%	95.1%	92.6%	98.4%	95.4%
SPCD+ Ensemble CNN[22]	90.3%	92.1%	91.8%	-	90.2%
DFUNet [37]	90.7%	90.3%	86.7%	-	86.7%
VGG16 [109]	-	-	92.3%	89.7%	90.9%
AlexNet [110]	-	-	91.1%	87.2%	89.1%
GoogleNet [111]	-	-	95.6%	90.5%	92.9%
CNN_GLCMNet+DNN	**97.4%**	**97.5%**	**97.5%**	97.2%	**97.3%**
CNN_GLCMNet+SVM	96.9%	96.9%	96.7%	96.9%	**96.8%**

Table 4.13 shows the performance of various models in the classification task. CNN_GLCMNet+DNN and "CNN_GLCMNet+SVM" stand out with high accuracy and balanced precision, recall, and F1 scores. The Hybrid DCNN with four branches model also exhibits strong performance. Models with higher accuracy may not always be the best choice, depending on the specific use case and the importance of minimizing false positives or false negatives. The model's efficiency is derived from its utilization of the combination of two technologies, namely GLCM and CNN, for the extraction of significant features. Furthermore, the deep neural network (DNN) has been employed as a classifier. Figure 4.8 shows the Comparison results between the proposed models and other models.

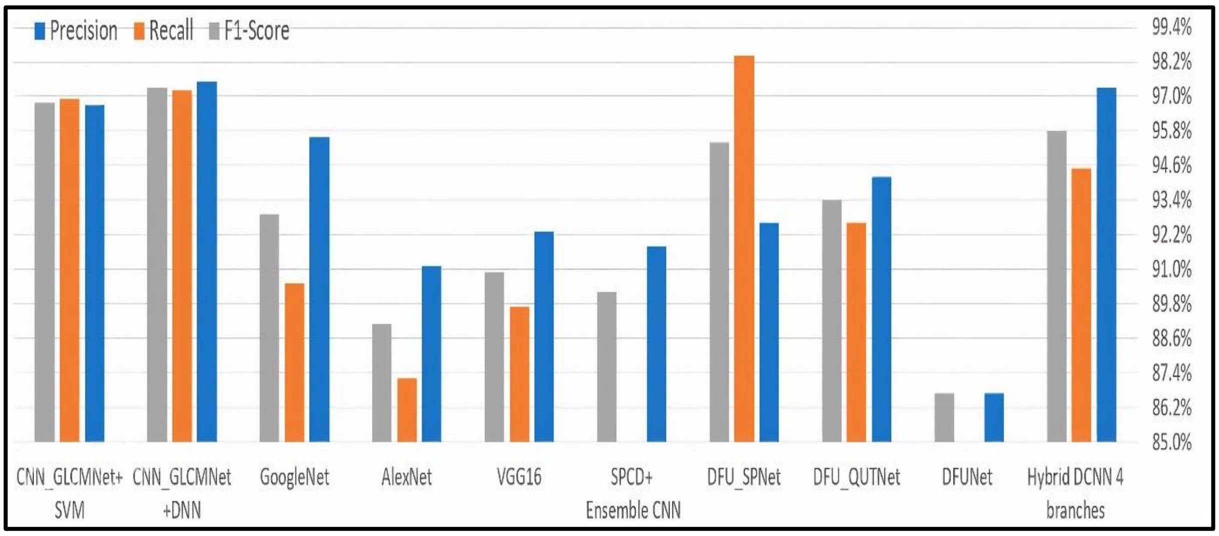


Figure 4.8 Comparison of our results of CLCM-CNN model with ten DL techniques through measuring of the performance of them in term of accuracy, accuracy balance, precision, recall, f1-score based on DFU and private DFU dataset

4.5.2 Estimated Executing Time of GLCM-CNN Model

Table 4.14 shows the execution time of the GLCM-CNN model and each stage of it, based on the testing dataset.

Table 4.14 Execution time of the GLCM-CNN model

Feature extraction by GLCM and CNN for each image time (sec)	DNN time (sec)	Total time (sec)
0.085	0.072	0.157

Table 4.14 explains the execution time of GLCM-CNN model with its stages. The feature extraction with each image takes 0.085s, and DNN takes 0.1290s. Finally, the total time of the GLCM-CNN takes 0.157s. So that, the model achieves less execution time.

4.5.3 Testing the Performance of WTA-CN

The performance of our clustering method can be evaluated by using an average silhouette score with a range of [-1, +1]. When this score is nearing +1, that means the clustering method is perfect; when the value of the score is nearing -1, the clustering method is worse. Table 4.15 shows the result of the silhouette score of WTA-CN with the number of epochs (300) and different Learning Rates. Figure 4.9 shows the results of the silhouette score for WTA-CN based on different learning rate values.

Table 4.15 The result of Winner-take-all competitive network with different hyperparameters

Cases	Learning Rate	Epochs	Silhouette Score
Case 1	0.01	300	**0.101**
Case 2	0.001	300	**0.119**
Case 3	0.0001	300	**0.429**
Case 4	0.00001	300	**0.306**

Based on the findings presented in Table 4.15, it is evident that the Learning Rate significantly influences the silhouette score, hence determining its quality. It is observed that when the learning rate is set to 0.01, the outcome is worse. However, as the learning rate decreases, there is an improvement in the silhouette score until it reaches a specified value of 0.0001. When the value exceeds a specified threshold of learning rate, the silhouette score will exhibit a drop.

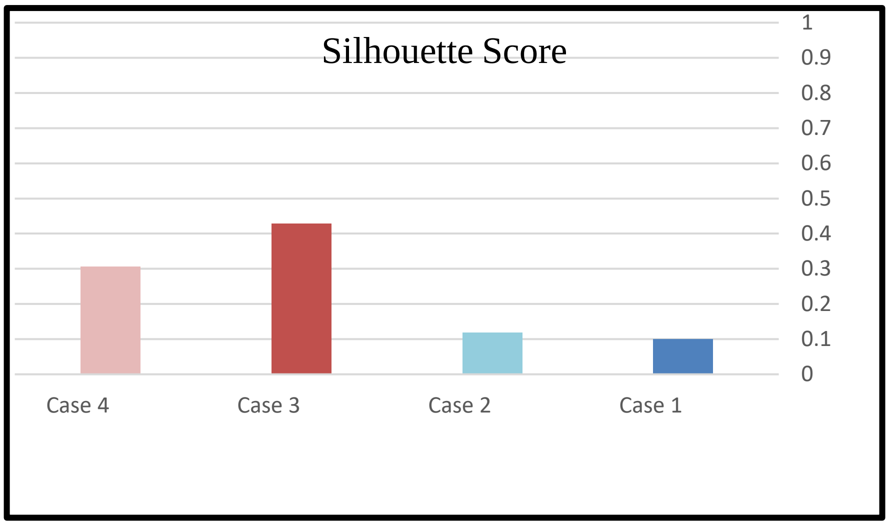


Figure 4.9 The results of silhouette score with different learning rates

4.6 Evaluate of CLAHE-CNN Model

In this section, explain the DR dataset that applies to the CLAHE-CNN model and then evaluate it using the performance metrics of SL and compute the execution time.

4.6.1 Testing CLAHE-CNN Model

Initially, apply the proposed CLAHE-CNN model to DR images. Applying the CLAHE method stages to enhance the images. Next apply CNN to extract important features and then input them to the DNN classifier. The performance of this model can be shown in Table 4.16.

Table 4.16 Evaluating the CLAHE-CNN model and four ML techniques

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
MLUDA-NN [23]	92.0%	70.0%	87.5%	77.7%
SVM [24]	83.90%	-	-	-
LM-NN [112]	80.0%	42.8%	75.0%	54.5%
DA-NN [112]	82.0%	46.1%	75.0%	57.1%
CLAHE-CNN	94.0%	91.0%	90.4%	90.6%

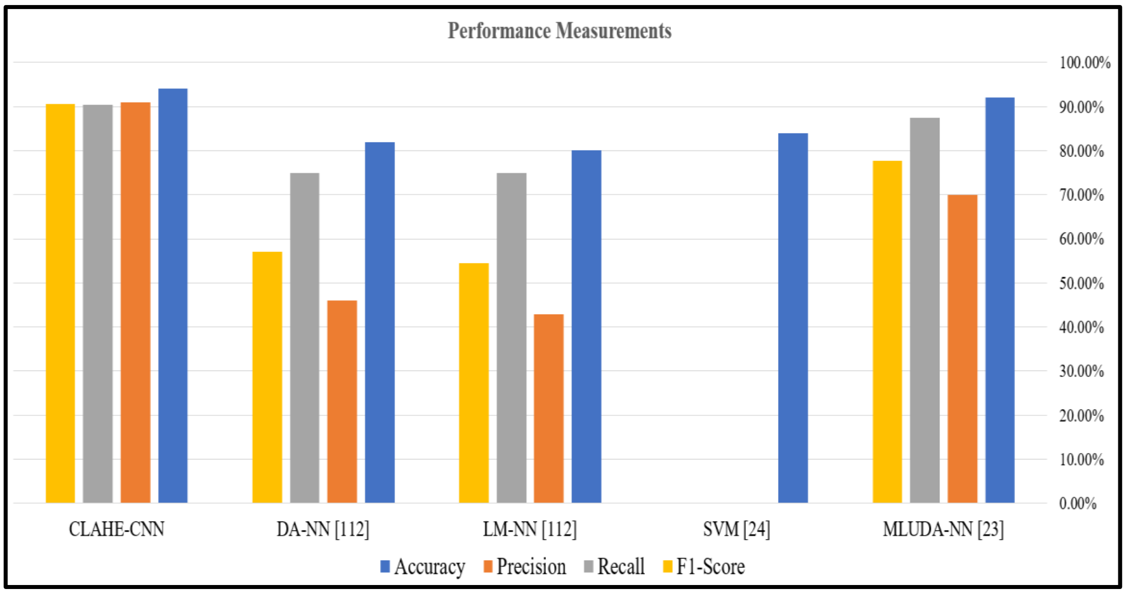


Figure 4.10 Comparison of the result of CLAHE-CNN model performance with other

According to Table 4.16, the proposed model stands out with the highest accuracy at 94.0% and strong precision, recall, and values. in addition, F1-Score was 90.6%, demonstrating a strong balance between precision and recall, indicating excellent performance in correctly classifying instances. The strength of the model lies in relying on two stages: the first enhancing the image by the CLAHE method. second obtaining important features by CNN. In addition, the DNN has been used as a classifier. Figure 4.10 shows the results of the CLAHE-CNN model compared with other approaches.

4.6.2 Estimated Executing Time of GLCM-CNN Model

Table 4.17 shows the execution time of the CLAHE-CNN model and each stage of it, based on the testing dataset.

Table 4.17 Execution time of the CLAHE-CNN model

Image enhancement time(sec)	CNN+DNN time (sec)	Total time (sec)
0.085	0.331	0.416

According to Table 4.17 the execution time of CLAHE-CNN model with its stages. The image enhancement stage takes 0.085s and CNN+DNN for feature extraction and classification takes 0.331s. Finally, the total time of the CLAHE-CNN takes 0.416s. So that, the model achieves less execution time.

4.7 Evaluate of Proposed System

In this section, evaluate the proposed system with different data types using the performance metrics of SL. The requirements for testing this system rely on real data that contains all or any data type sources. Due to the lack of this dataset, samples were taken from available data sources to evaluate the system.

4.7.1 Testing the Proposed System

Initially, the proposed system is applied to different data types such as tabular data, ECG data, DFU image data, and DR image data. After that apply each data type to own trained models. Then aggregate the output from these models to produce the final decision. shows the accuracy results of proposed model with different cases. Each case represents testing data for specific input data type.

Table 4.18 The results of proposed system with different cases

Cases	Details	Accuracy
Case 1	In this case, the testing set of the inputs data including tabular data	98.9%
Case 2	**In this case, the testing set of the inputs data including ecg data**	**99.8%**
Case 3	In this case, the testing set of the inputs data including DFU data	97.4%
Case 4	**In this case, the testing set of the inputs data including DR data**	**94.0%**
Case 5	In this case, the samples are picked from test set of the inputs data including tabular, ecg, and image data (take all inputs). The size of sample is 50.	98.0%

In Table 4.18 case 1 represents the tabular data that achieves the accuracy 98.9%, case 2 represents the ECG data that achieves the accuracy 99.8%, case 3 represents the DFU data that achieves the accuracy 97.4%, case 4 represents the DR data that achieves the accuracy 94.0%. case 5 represents tabular and ECG data type will input to model that achieves the accuracy 98.0%.

In summary, the proposed system appears to be robust and versatile, achieving high accuracy across different types of medical data (tabular, ECG, and image) data types. It performs exceptionally well on ECG data and tabular data, indicating its suitability for tasks related to diabetes mellitus. Although accuracy is slightly lowered for DR data, it remains reasonable, indicating that the approach can be broadly applied but may face difficulties in certain medical fields.

CHAPTER FIVE:

Conclusions and Future Works

Conclusions and Future Works

5.1 Conclusions

The primary objective of the proposed system is to build a system for classifying DM and its complications that deal with multiple data sources, including (tabular, ECG, and image data) and enhance the accuracy. It, therefore, extends previous literature on the classification of DM and its complications with multiple data sources based on DL techniques. The proposed topology is based on four models, each dealing with the source of the data type. First, CEAR-DNN deals with tabular data to classify the DM based on the autoencoder to regenerate and select features and DNN for classification. Second, the ECG-CNN model uses ECG data for classifying the DM based on CNN and DNN techniques with statistical tools. Third, GLCM-CNN model use DFU images data as inputs to classify into normal and abnormal foot skin based on GLCM and CNN techniques. Fourth, CLAHE-CNN model use DR image data as input and based on CLAHE method and CNN technique. Furthermore, the results of the empirical study showed that the performance of the proposed models are enhanced in comparison with other previously published models.

Many conclusions can be made from the outcomes of the proposed model as follows:

1. The efficiency of employing the autoencoder technique has been demonstrated in its capacity to effectively rebuild primary features, hence enhancing the accuracy of tabular data models. Furthermore, the utilization of the regularization technique enables the identification and prioritization of crucial characteristics that play an important role in the classification process.

2. The effectiveness of utilizing the CNN technique with statistical methods for the extraction of important features on ECG data has been proven.
3. Converting the HR signal that is produced from the ECG signal to an image using the spectrogram technique plays an important role in extracting new abstract features that combine with statistical features to enhance the classification process.
4. The utilization of the GLCM approach and CNN for the extraction of relevant features for DFU images has been found to have a substantial impact on enhancing the performance of the classifying model.
5. The utilization of a deep neural network (DNN) classifier employing a GLCM_CNN model yielded higher accuracy in comparison to the support vector machine (SVM) classifier. However, this improvement in accuracy comes at the cost of increased complexity, since the DNN classifier necessitates the processing of numerous layers.
6. Using the CLAHE method as a preprocessing stage to enhance the images of DR and highlight important features in the image that enable the CNN technique to extract the features that were not clear before applying the CLAHE method.

5.2 The Limitations of Proposed System

While the research findings show promise, this study has some limitations that warrant additional research. It can be summarized as follows:

1- The system is designed to deal with ECG data as time series signal instead of ECG image data.

2- The system is unable to locate the position of ulcer foot, but it can classify the ulcer foot accurately.

3- The CLAHE- CNN model for DR classification in the system is designed for binary classification instead of multiple classification that shows the state of DR image case.

5.3 Future Works:

Below are some potential future works that can be considered:

- Using another dataset that relates to DM classification, such as breath signals and glucose level signals, which is related to the disease. And addressing another DM complication, such as nephropathy or neuropathy.
- Apply a recurrent neural network such as the Long Short-Term Memory Networks (LSTM) architecture with an ECG signal for classification of the DM, due to the ability of this architecture to deal with time series data.
- Apply an attention model such as Vision Transformer (VIT) with the DFU and DR dataset to predict the normal or abnormal, due to the ability of this architecture to deal with image data.
- Applying the proposed system to other diseases, such as heart disease.

References

References

[1] S. Pandya *et al.*, “A study of the recent trends of immunology: Key challenges, domains, applications, datasets, and future directions,” *Sensors*, vol. 21, no. 23, pp. 1–40, 2021, doi: 10.3390/s21237786.

[2] S. Brogi and V. Calderone, “Artificial Intelligence in Translational Medicine,” *Int. J. Transl. Med. 2021, Vol. 1, Pages 223-285*, vol. 1, no. 3, pp. 223–285, Nov. 2021, doi: 10.3390/IJTM1030016.

[3] A. Rajkomar, J. Dean, and I. Kohane, “Machine Learning in Medicine,” *N. Engl. J. Med.*, vol. 380, no. 14, pp. 1347–1358, Apr. 2019, doi: 10.1056/NEJMRA1814259/SUPPL_FILE/NEJMRA1814259_DISCLOSURES.PDF.

[4] J. A. M. Sidey-Gibbons and C. J. Sidey-Gibbons, “Machine learning in medicine: a practical introduction,” *BMC Med. Res. Methodol.*, vol. 19, no. 1, pp. 1–18, 2019, doi: 10.1186/s12874-019-0681-4.

[5] S. Secinaro, D. Calandra, A. Secinaro, V. Muthurangu, and P. Biancone, “The role of artificial intelligence in healthcare: a structured literature review,” *BMC Med. Inform. Decis. Mak.*, vol. 21, no. 1, pp. 1–23, 2021, doi: 10.1186/s12911-021-01488-9.

[6] P. Kim, “MATLAB Deep Learning With Machine Learning, Neural Networks and Artificial Intelligence,” *MATLAB Deep Learn.*, vol. 130, no. 21, 2017, doi: 10.1007/978-1-4842-2845-6.

[7] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, “A Review on Deep Learning Methods for ECG Arrhythmia Classification,” *Expert Syst. with Appl. X*, vol. 7, p. 100033, 2020, doi: 10.1016/j.eswax.2020.100033.

[8] A. I. Károly, R. Fullér, and P. Galambos, “Unsupervised clustering for deep learning: A tutorial survey,” *Acta*

Polytech. Hungarica, vol. 15, no. 8, pp. 29–53, 2018, doi: 10.12700/APH.15.8.2018.8.2.

[9] A. A. Patel, *Hands-on unsupervised learning using Python : how to discover hidden patterns in unlabeled data*. O'Reilly Media, 2019.

[10] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005, doi: 10.1109/TNN.2005.845141.

[11] E. Pekel Özmen and T. Özcan, "Diagnosis of diabetes mellitus using artificial neural network and classification and regression tree optimized with genetic algorithm," *J. Forecast.*, vol. 39, no. 4, pp. 661–670, 2020, doi: 10.1002/for.2652.

[12] J. O. Orukwo and L. G. Kabari, "Diagnosing Diabetes Using Artificial Neural Networks," *Eur. J. Eng. Res. Sci.*, vol. 5, no. 2, pp. 221–224, 2020.

[13] H. Gupta, H. Varshney, T. K. Sharma, N. Pachauri, and O. P. Verma, "Comparative performance analysis of quantum machine learning with deep learning for diabetes prediction," *Complex Intell. Syst.*, vol. 8, no. 4, pp. 3073–3087, 2022, doi: 10.1007/s40747-021-00398-7.

[14] M. Abedini, A. Bijari, and T. Banirostam, "Classification of Pima Indian Diabetes Dataset using Ensemble of Decision Tree , Logistic Regression and Neural Network," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 9, no. 7, pp. 7–10, 2020, doi: 10.17148/IJARCCE.2020.9701.

[15] V. Chang, J. Bailey, Q. A. Xu, and Z. Sun, "Pima Indians diabetes mellitus classification based on machine learning (ML) algorithms," *Neural Comput. Appl.*, vol. 0123456789, 2022, doi: 10.1007/s00521-022-07049-z.

[16] M. T. Alasaady, T. Noranis, M. Aris, N. M. Sharef, and H. Hamdan, "A proposed approach for diabetes diagnosis using neuro-fuzzy technique," vol. 11, no. 6, pp. 3590–3597, 2022,

doi: 10.11591/eei.v11i6.4269.

[17] L. Alzubaidi, A. A. Abbood, M. A. Fadhel, O. Al-Shamma, and J. Zhang, “Comparison of hybrid convolutional neural networks models for diabetic foot ulcer classification,” *J. Eng. Sci. Technol.*, vol. 16, no. 3, pp. 2001–2017, 2021, doi: http://doi.org/10.2139/ssrn.4010975.

[18] L. Alzubaidi, M. A. Fadhel, S. R. Oleiwi, O. Al-Shamma, and J. Zhang, “DFU_QUTNet: diabetic foot ulcer classification using novel deep convolutional neural network,” *Multimed. Tools Appl.*, vol. 79, no. 21–22, pp. 15655–15677, 2020, doi: 10.1007/s11042-019-07820-w.

[19] S. S. Reddy, G. Mahesh, and N. M. Preethi, “Exploiting Machine Learning Algorithms to Diagnose Foot Ulcers in Diabetic Patients,” *EAI Endorsed Trans. Pervasive Heal. Technol.*, vol. 7, no. 29, pp. 1–14, 2021, doi: 10.4108/eai.24-8-2021.170752.

[20] J. Amin, M. Sharif, M. A. Anjum, H. U. Khan, M. S. A. Malik, and S. Kadry, “An Integrated Design for Classification and Localization of Diabetic Foot Ulcer based on CNN and YOLOv2-DFU Models,” *IEEE Access*, vol. 8, pp. 228586–228597, 2020, doi: 10.1109/ACCESS.2020.3045732.

[21] S. K. Das, P. Roy, and A. K. Mishra, “DFU_SPNet: A stacked parallel convolution layers based CNN to improve Diabetic Foot Ulcer classification,” *ICT Express*, vol. 8, no. 2, pp. 271–275, 2022, doi: 10.1016/j.icte.2021.08.022.

[22] M. Goyal, N. D. Reeves, S. Rajbhandari, N. Ahmad, C. Wang, and M. H. Yap, “Recognition of ischaemia and infection in diabetic foot ulcers: Dataset and techniques,” *Comput. Biol. Med.*, vol. 117, p. 103616, 2020, doi: 10.1016/j.compbiomed.2020.103616.

[23] A. S. Jadhav, P. B. Pati, and S. Biradar, “Computer-aided diabetic retinopathy diagnostic model using optimal thresholding merged with neural network,” *Int. J. Intell.*

Comput. Cybern., vol. 13, no. 3, pp. 283–310, 2020, doi: 10.1108/IJICC-11-2019-0119.

[24] H. Y. Tsao, P. Y. Chan, and E. C. Y. Su, “Predicting diabetic retinopathy and identifying interpretable biomedical features using machine learning algorithms,” *BMC Bioinformatics*, vol. 19, no. Suppl 9, 2018, doi: 10.1186/s12859-018-2277-0.

[25] G. Swapna, R. Vinayakumar, and K. P. Soman, “Diabetes detection using deep learning algorithms,” *ICT Express*, vol. 4, no. 4, pp. 243–246, 2018, doi: 10.1016/j.icte.2018.10.005.

[26] A. P. T. Seyd, P. K. Joseph, and J. Jacob, “Automated diagnosis of diabetes using heart rate variability signals,” *J. Med. Syst.*, vol. 36, no. 3, pp. 1935–1941, 2012, doi: 10.1007/s10916-011-9653-x.

[27] Y. Aggarwal, J. Das, P. M. Mazumder, R. Kumar, and R. K. Sinha, “Heart rate variability features from nonlinear cardiac dynamics in identification of diabetes using artificial neural network and support vector machine,” *Biocybern. Biomed. Eng.*, vol. 40, no. 3, pp. 1002–1009, 2020, doi: 10.1016/j.bbe.2020.05.001.

[28] T. R. Gadekallu, N. Khare, S. Bhattacharya, S. Singh, P. K. R. Maddikunta, and G. Srivastava, “Deep neural networks to predict diabetic retinopathy,” *J. Ambient Intell. Humaniz. Comput.*, no. 0123456789, 2020, doi: 10.1007/s12652-020-01963-7.

[29] H. A. Ismael, N. H. Al-A’araji, and B. K. Shukur, “Enhanced the prediction approach of diabetes using an autoencoder with regularization and deep neural network,” *Period. Eng. Nat. Sci.*, vol. 10, no. 6, pp. 156–167, 2022, doi: 10.21533/pen.v10i6.3394.

[30] A. Iyer, J. S, and R. Sumbaly, “Diagnosis of Diabetes Using Classification Mining Techniques,” *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 1, pp. 01–14, 2015, doi: 10.5121/ijdkp.2015.5101.

[31] J. Chaki, S. Thillai Ganesh, S. K. Cidham, and S. Ananda Theertan, “Machine learning and artificial intelligence based Diabetes Mellitus detection and self-management: A systematic review,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3204–3225, 2020, doi: 10.1016/j.jksuci.2020.06.013.

[32] M. Alehegn, “Analysis and Prediction of Diabetes Mellitus using Machine Learning Algorithm,” *Int. J. Pure Appl. Math.*, vol. 118, no. 9, pp. 871–878, 2018.

[33] “Complications of diabetes | Guide to diabetes | Diabetes UK.”https://www.diabetes.org.uk/guide-to diabetes/ complications (accessed Feb. 20, 2021).

[34] L. Wang, P. C. Pedersen, D. Strong, B. Tulu, and E. Agu, “Wound image analysis system for diabetics,” *Med. Imaging 2013 Image Process.*, vol. 8669, pp. 866–924, 2013, doi: 10.1117/12.2004762.

[35] J. Apelqvist and J. Larsson, “What is the most effective way to reduce incidence of amputation in the diabetic foot?,” *Diabetes. Metab. Res. Rev.*, vol. 16, pp. 75–83, 2000, doi: 10.1002/1520-7560(200009/10)16:1+<::aid-dmrr139>3.0.co;2-8.

[36] D. Leightley, J. S. McPhee, and M. H. Yap, “Automated Analysis and Quantification of Human Mobility Using a Depth Sensor,” *IEEE J. Biomed. Heal. informatics*, vol. 21, no. 4, pp. 939–948, Jul. 2017, doi: 10.1109/JBHI.2016.2558540.

[37] M. Goyal, N. D. Reeves, S. Rajbhandari, and M. H. Yap, “Robust Methods for Real-Time Diabetic Foot Ulcer Detection and Localization on Mobile Devices,” *IEEE J. Biomed. Heal. informatics*, vol. 23, no. 4, pp. 1730–1741, Jul. 2019, doi: 10.1109/JBHI.2018.2868656.

[38] A. Alberdi, A. Aztiria, and A. Basarab, “Towards an automatic early stress recognition system for office environments based on multimodal measurements: A

review," *J. Biomed. Inform.*, vol. 59, pp. 49–75, Feb. 2016, doi: 10.1016/J.JBI.2015.11.007.

[39] P. Kumar, A. K. Das, Prachita, and S. Halder, "Time-domain HRV Analysis of ECG Signal under Different Body Postures," *Procedia Comput. Sci.*, vol. 167, no. 2019, pp. 1705–1710, 2020, doi: 10.1016/j.procs.2020.03.435.

[40] D. Sadhukhan and M. Mitra, "R-Peak Detection Algorithm for Ecg using Double Difference And RR Interval Processing," *Procedia Technol.*, vol. 4, pp. 873–877, 2012, doi: 10.1016/j.protcy.2012.05.143.

[41] J. Irvine, S. A. Israel, M. D. Wiederhold, and B. K. Wiederhold, "A New Biometric: Human Identification from Circulatory Function Chronic pain treatment View project Positive Technology View project," *Artic. J. Am. Stat. Assoc.*, no. January, 2003, [Online]. Available: https://www.researchgate.net/publication/314181179.

[42] P. Seyd and V. Ahamed, "Time and frequency domain analysis of heart rate variability and their correlations in diabetes mellitus," *Int. J. Biol. Life Sci.*, vol. 4, no. 1, pp. 24–27, 2008.

[43] M. Yilmaz, H. Kayancicek, and Y. Cekici, "Heart rate variability: Highlights from hidden signals," *J. Integr. Cardiol.*, vol. 4, no. 5, pp. 1–8, 2018, doi: 10.15761/jic.1000258.

[44] S. Chattopadhyay, "The Importance of Time-Domain HRV Analysis in Cardiac Health Prediction," *Ser. Cardiol. Res.*, vol. 4, no. 1, pp. 19–23, 2023, doi: 10.54178/2768-5985.2022a5.

[45] S. Vieira, W. H. Lopez Pinaya, and A. Mechelli, *Main concepts in machine learning*. Elsevier Inc., 2019.

[46] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining (New International Editon)*, no. September. 2014.

[47] S. Wang and W. Shi, *Data mining and knowledge discovery*. 2012.

[48] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.

[49] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” *Adv. Neural Inf. Process. Syst.*, vol. 2020-Decem, 2020.

[50] Y. Lecun *et al.*, “Gradient-based learning applied to document recognition,” vol. 86, no. 11, pp. 2278–2324, 2023.

[51] H. A. Ismael, N. H. Al-A’araji, and B. K. Shukur, “An Enhanced Diabetic Foot Ulcer Classification Approach Using GLCM and Deep Convolution Neural Network,” *Karbala Int. J. Mod. Sci.*, vol. 8, no. 4, pp. 1–10, 2022, doi: https://doi.org/10.33640/2405-609X.3268.

[52] Y. Arpitha, G. L. Madhumathi, and N. Balaji, “Spectrogram analysis of ECG signal and classification efficiency using MFCC feature extraction technique,” *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 2, pp. 757–767, 2022, doi: 10.1007/s12652-021-02926-2.

[53] V. Misra, “Applications and Applied Mathematics : An International (SI10-068) Performance Analysis of Cosine Window Function,” vol. 17, no. 3, 2022.

[54] Z. Průša, “THIS IS A PREPRINT OF A PAPER PRESENTED AT THE AES INTERNATIONAL CONFERENCE ON SEMANTIC AUDIO The Phase Retrieval Toolbox,” no. June, pp. 1–6, 2017, [Online]. Available: http://github.com/ltfat/phaseret.

[55] S. Tanwar, T. Ramani, and S. Tyagi, “Dimensionality reduction using PCA and SVD in big data: A comparative case study,” *Lect. Notes Inst. Comput. Sci. Soc.*

Telecommun. Eng. LNICST, vol. 220, pp. 116–125, 2018, doi: 10.1007/978-3-319-73712-6_12.

[56] P. D. Lax, *Linear Algebra Linear Algebra*, no. November. 2018.

[57] I. Horace H-S, *Digital image processing and computer vision*, vol. 8, no. 3. 1990.

[58] H. C. Andrews, "Digital Image Processing," *Computer (Long. Beach. Calif).*, vol. 7, no. 5, pp. 17–19, 1974, doi: 10.1109/MC.1974.6323522.

[59] D. J. Hemanth, O. Deperlioglu, and U. Kose, "An enhanced diabetic retinopathy detection and classification approach using deep convolutional neural network," *Neural Comput. Appl.*, vol. 0, 2019, doi: 10.1007/s00521-018-03974-0.

[60] "Contrast Limited Adaptive Histogram Equalization - MATLAB & Simulink." https://www.mathworks.com/help/visionhdl/ug/contrast-adaptive-histogram-equalization.html (accessed Nov. 25, 2023).

[61] C. N. Rao, S. S. Sastry, K. Mallika, H. S. Tiong, and K. B. Mahalakshmi, "Co-Occurrence Matrix and Its Statistical Features as an Approach for Identification Of Phase Transitions Of Mesogens," *Int. J. Innov. Res. Sci. Eng. Technol. (An ISO*, vol. 3297, no. 9, pp. 2319–8753, 2007, [Online]. Available: www.ijirset.com.

[62] L. Armi and S. Fekri-Ershad, "Texture image analysis and texture classification methods - A review," vol. 2, no. 1, pp. 1–29, 2019, [Online]. Available: http://arxiv.org/abs/1904.06554.

[63] M. H. Bharati, J. J. Liu, and J. F. MacGregor, "Image texture analysis: Methods and comparisons," *Chemom. Intell. Lab. Syst.*, vol. 72, no. 1, pp. 57–71, 2004, doi: 10.1016/j.chemolab.2004.02.005.

[64] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural

Features for Image Classification," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973, doi: 10.1109/TSMC.1973.4309314.

[65] M. Partio, B. Cramariuc, M. Gabbouj, and A. Visa, "Rock texture retrieval using gray level co-occurrence matrix," *Proc. 5th Nord. Signal Conf.*, vol. 75, pp. 1–5, 2002, doi: https://doi.org/10.1049/iet-ipr.2018.6440.

[66] S. Sachar and A. Kumar, "Survey of feature extraction and classification techniques to identify plant through leaves," *Expert Syst. Appl.*, vol. 167, p. 114181, Apr. 2021, doi: 10.1016/J.ESWA.2020.114181.

[67] dkk 2018) richard oliver (dalam Zeithml., "済無No Title No Title No Title," *Angew. Chemie Int. Ed. 6(11), 951–952.*, vol. 5, no. 4, pp. 2013–2015, 2021.

[68] H. Taud and J. Mas, "Multilayer Perceptron (MLP), in Geomatic Approaches for Modeling Land Change Scenarios," *Geomat. approaches Model. L. Chang. Scenar.*, pp. 451–455, 2018, doi: 10.1007/978-3-319-60801-3_27.

[69] A.-N. Sharkawy, "Principle of Neural Network and Its Main Types: Review," *J. Adv. Appl. Comput. Math.*, vol. 7, pp. 8–19, 2020.

[70] S. B. H. Gmbh, *Soft-Organization and Associative Memory*. 2001.

[71] K. G. Kim, "Deep Learning," vol. 22, no. 4, pp. 351–354, 2016.

[72] M. A. Ranzato, *Deep Learning for Vision*, no. October. 2013.

[73] A. Arithmetical and F. Independence, *Neural Networks AND STATISTICAL and*, vol. 59. 1937.

[74] A. M. Alhassan and W. M. N. W. Zainon, "Brain tumor classification in magnetic resonance image using hard swish-based RELU activation function-convolutional neural

network," *Neural Comput. Appl.*, vol. 33, no. 15, pp. 9075–9087, 2021, doi: 10.1007/S00521-020-05671-3.

[75] H. Faris, I. Aljarah, and S. Mirjalili, "Training feedforward neural networks using multi-verse optimizer for binary classification problems," *Appl. Intell.*, vol. 45, pp. 322–332, 2016, doi: 10.1007/s10489-016-0767-1.

[76] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[77] G. Swapna, R. Vinayakumar, and K. P. Soman, "Diabetes detection using deep learning algorithms," *ICT Express*, vol. 4, no. 4, pp. 243–246, 2018, doi: 10.1016/j.icte.2018.10.005.

[78] M. A. Amirabadi, M. H. Kahaei, and S. A. Nezamalhosseini, "Novel suboptimal approaches for hyperparameter tuning of deep neural network [under the shelf of optical communication]," *Phys. Commun.*, vol. 41, p. 101057, 2020, doi: 10.1016/j.phycom.2020.101057.

[79] A. L. A. D. R. S. M. S. S. V. G. F. V. L. C. G. Fragata, *Artificial Intelligence: Theory and Applications*. 2022.

[80] G. Yang *et al.*, "Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 17084–17097, Dec. 2021.

[81] B. Zoph and Q. V Le, "Searching for activation functions," *6th Int. Conf. Learn. Represent. ICLR 2018 - Work. Track Proc.*, pp. 1–13, 2018.

[82] C. C. Aggarwal, "Neural Networks and Deep Learning," *Neural Networks Deep Learn.*, 2018, doi: 10.1007/978-3-319-94463-0.

[83] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1–40, 2009.

[84] S. Hikmat Haji and A. Mohsin Abdulazeez, "comparison of

optimization techniques based on gradient descent algorithm: a review pjaee, 18 (4) (2021) comparison of optimization techniques based on gradient descent algorithm: a review comparison of Optimization Techniques Based On Gradient Descent Al," *J. Archaeol. Egypt/Egyptology*, vol. 18, no. 4, pp. 2715–2743, 2021.

[85] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016, [Online]. Available: http://arxiv.org/abs/1609.04747.

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

[87] Y. Gao, J. Li, Y. Zhou, F. Xiao, and H. Liu, "Optimization Methods for Large-Scale Machine Learning," *2021 18th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. ICCWAMTIP 2021*, vol. 60, no. 2, pp. 304–308, 2021, doi: 10.1109/ICCWAMTIP53232.2021.9674150.

[88] P. Li, "Optimization Algorithms for Deep Learning," pp. 1–10, 2017, [Online]. Available: https://www.algorithms.com/deep/learning/optimization/.

[89] I. Goodfellow, "Book: Deep Learning," *Prmu*, pp. 1–10, 2016, [Online]. Available: www.deeplearningbook.org.

[90] R. Gençay and M. Qi, "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging," *IEEE Trans. Neural Networks*, vol. 12, no. 4, pp. 726–734, 2001, doi: 10.1109/72.935086.

[91] C. Szegedy and S. G. Com, "Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Int. Conf. Mach. Learn.*, vol. 37, 2015.

[92] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[93] C. C. Aggarwal, *Neural Networks and Deep Learning*. 2018.

[94] R. Tibshirani, “Lasso Tibshirani.pdf,” *Journal of the Royal Statistical Society, Series B (Methodogical),* vol. 58, no. 1. pp. 267–288, 1996.

[95] S. Wang, F. Zheng, and L. Xu, “Comparison between particle swarm optimization and genetic algorithm in artificial neural network for life prediction of NC tools,” *J. Adv. Manuf. Syst.*, vol. 7, no. 1, pp. 1–7, 2008, doi: 10.1142/S0219686708001073.

[96] M. Kumar, M. Husain, N. Upreti, and D. Gupta, “Genetic Algorithm: Review and Application,” *SSRN Electron. J.*, vol. 2, no. 2, pp. 451–454, 2020, doi: 10.2139/ssrn.3529843.

[97] K. Deb, “Genetic Algorithm in Search and Optimization: The Technique and Applications,” *Proc. Int. Work. Soft Comput. Intell. Syst.*, pp. 58–87, 1998, [Online]. Available: http://repository.ias.ac.in/82743/.

[98] Y. Fang, M. A. Cohen, and T. G. Kincaid, “Dynamic analysis of a general class of winner-take-all competitive neural networks,” *IEEE Trans. Neural Networks*, vol. 21, no. 5, pp. 771–783, 2010, doi: 10.1109/TNN.2010.2041671.

[99] L. Rutkowski, K. Cpałka, R. Nowicki, A. Pokropińska, and R. Scherer, *Neuro-fuzzy systems*, vol. 9781461418. 2012.

[100] P. Baldi, “Autoencoders, Unsupervised Learning, and Deep Architectures,” *ICML Unsupervised Transf. Learn.*, pp. 37–50, 2012, doi: 10.1561/2200000006.

[101] M. Hossin and M. N. Sulaiman, “A review on evaluation metrics for data classification evaluations,” *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 1–11, 2015.

[102] K. R. Shahapure and C. Nicholas, “Cluster Quality Analysis Using Silhouette Score,” *Am. J. o*, vol. 15, no. 2, pp. 7–22, 2020.

[103] D. Gholamiangonabadi, N. Kiselov, and K. Grolinger,

"Deep Neural Networks for Human Activity Recognition with Wearable Sensors: Leave-One-Subject-Out Cross-Validation for Model Selection," *IEEE Access*, vol. 8, pp. 133982–133994, 2020, doi: 10.1109/ACCESS.2020.3010715.

[104] K. Pal and B. V. Patel, "Data Classification with k-fold Cross Validation and Holdout Accuracy Estimation Methods with 5 Different Machine Learning Techniques," *Proc. 4th Int. Conf. Comput. Methodol. Commun. ICCMC 2020*, no. Iccmc, pp. 83–87, 2020, doi: 10.1109/ICCMC48092.2020.ICCMC-00016.

[105] Dua D and Karra Taniskidou, "UCI Machine Learning Repository," *University of California, Irvine, School of Information and Computer Sciences*, 2017. .

[106] E. Decencière *et al.*, "Feedback on a publicly distributed image database: The Messidor database," *Image Anal. Stereol.*, vol. 33, no. 3, pp. 231–234, 2014, doi: 10.5566/ias.1155.

[107] H. Naz and S. Ahuja, "Deep learning approach for diabetes prediction using PIMA Indian dataset," *J. Diabetes Metab. Disord.*, vol. 19, no. 1, pp. 391–403, 2020, doi: 10.1007/s40200-020-00520-5.

[108] M. R. Rajput and S. S. Khedgikar, "Diabetes prediction and analysis using medical attributes: A Machine learning approach," *J. Xi'an Univ. Archit. Technol.*, vol. 14, no. 1, pp. 98–103, 2022, doi: 10.37896/JXAT14.01/314405.

[109] A. Krishnaswamy Rangarajan and R. Purushothaman, "Disease Classification in Eggplant Using Pre-trained VGG16 and MSVM," *Sci. Reports 2020 101*, vol. 10, no. Going, pp. 1–11, Feb. 2020, doi: 10.1038/s41598-020-59108-x.

[110] K. M. Hosny, M. A. Kassem, and M. M. Fouad, "Classification of Skin Lesions into Seven Classes Using Transfer Learning with AlexNet," *J. Digit. Imaging 2020*

335, vol. 33, pp. 1325–1334, Jun. 2020, doi: 10.1007/S10278-020-00371-9.

[111]C. Szegedy *et al.*, “Going deeper with convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.

[112]F. Fernández-Navarro, M. Carbonero-Ruz, D. B. Alonso, and M. Torres-Jimenez, “Global sensitivity estimates for neural network classifiers,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 11, pp. 2592–2604, Nov. 2017, doi: 10.1109/TNNLS.2016.2598657.

الخلاصة

داء السكري هو اضطراب أيضي مزمن يصيب الملايين من الأفراد في جميع أنحاء العالم .ويشكل هذا المرض، بمضاعفاته، خطرا صحيا كبيرا .هناك العديد من النماذج الناجحة للتنبؤ بمرض السكري ومضاعفاته .علاوة على ذلك، لم يتعامل أي من هذه النماذج مع مصادر بيانات متعددة .تقترح هذه الأطروحة نهجا جديدا للتعامل مع عدم تجانس مصادر البيانات، بما في ذلك أنواع البيانات الجدولية، وتخطيط القلب، والصور .علاوة على ذلك، تُستخدم نماذج التعلم العميق المعدلة للتنبؤ بمرض السكري ومضاعفاته (قرحة القدم السكرية، واعتلال الشبكية السكري) من خلال استخدام مصادر بيانات متعددة.

يعالج النهج المقترح كل نوع من البيانات باستخدام نموذج مخصص، حيث يحتوي كل نموذج على ثلاث مراحل رئيسية: مرحلة المعالجة المسبقة للبيانات، مرحلة استخراج الميزات ومرحلة التنبؤ .هناك أربعة نماذج مقترحة .النموذج الأول، المسمى جهاز التشفير التلقائي الكامل مع التنظيم والشبكة العصبية العميقة(CAER-DNN) ، مصمم للتنبؤ بمرض السكري من خلال التعامل مع مصادر البيانات الجدولية .تتمثل المساهمة الرئيسية لهذا النموذج في استخدام تقنية غير خاضعة للرقابة لتجديد الميزات الأساسية التي تعمل كمدخل لمصنف DNN. أما النموذج الثاني، المسمى-ECG DNN، فهو مقترح للتعامل مع مصادر بيانات تخطيط القلب .هذا النموذج فريد من نوعه من خلال الاستفادة من الأدوات الإحصائية وتقنيات الشبكة العصبية التلافيفية لاستخراج ميزات مهمة من إشارة تقلب معدل ضربات القلب .أما النموذج الثالث المسمى (CNN_GLCMNet) فيتناول التنبؤ بقرحة القدم السكرية من خلال الاستفادة من مصادر بيانات الصورة .أصل هذا النموذج هو مزيج من قوة طريقة مصفوفة التكرار المشترك للمستوى الرمادي (GLCM) وشبكة CNN لاستخراج الميزات .بالإضافة إلى التنبؤ، يقوم هذا النموذج بتجميع حالة DFU إلى ثلاث مجموعات بناءً على خطورتها .النموذج الرابع، المسمىCLAHE_CNN ، متخصص في التنبؤ باعتلال الشبكية السكري باستخدام مصدر بيانات الصورة .يستفيد هذا النموذج من معادلة الرسم البياني التكيفي المحدودة التباين (CLAHE) لتحسين تباين الصورة، إلى جانب CNN لاستخراج الميزات.

وأخيرا، تم اقتراح النموذج مجمع لتجميع قرارات النماذج والحصول على القرار النهائي .تم تصميم هذا النموذج لأخذ مصادر بيانات متعددة في الاعتبار .لمعالجة البيانات الجدولية، يتم استخدام نموذج CAER-DNN ويحقق دقة تبلغ 98.90% و98.48% على مجموعات بيانات PID وMID، على التوالي .للتعامل مع بيانات مخطط كهربية القلب(ECG) ، تم تطبيق نموذج ECG-DNN بدرجة دقة 99.88%.

لمعالجة البيانات الصورية، تم استخدام نموذجين الاول GLCM-CNN للتنبؤ بـ DFU، حيث حقق النموذج GLCM-CNN دقة قدرها 97.43% . اما بالنسبة لعملية العنقدة للصور المصابة ب DFU حيث حقق معدل silhouette 0.429 للتجميع . بالنسبة للتنبؤ بـDR ، يحقق نموذج CLAHE_CNN دقة تبلغ 94.0%.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعــــة بـابــــل

كلية تكنولوجيا المعلومات -قسم البرمجيات

نماذج تصنيف التعلم العميق لمرض السكري ومضاعفاته من مصادر متعددة

اطروحة مقدمة إلى

مجلس كلية تكنولوجيا المعلومات -جامعة بابل كجزء من متطلبات

نيل درجة الدكتوراه فلسفة في تكنولوجيا المعلومات / برمجيات

من قبل

حسين علي إسماعيل حميد

بإشــراف

أ.د. نبيل هاشم الاعرجي

أ.د. بهيجة خضير شكر الحلو

1445 هـ 2023 م