Detection of Atrial Fibrillation in ECG Signals Using Machine Learning

by

Shahin Almasi
B.Sc., Azad University of Ahvaz, Iran, 2013

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Detection of Atrial Fibrillation in ECG Signals Using Machine Learning

by

Shahin Almasi
B.Sc., Azad University of Ahvaz, Iran, 2013

Supervisory Committee

Dr. T. Aaron Gulliver, Department of Electrical and Computer Engineering, (Supervisor)

Dr. Amirali Baniasadi, Department of Electrical and Computer Engineering, (Departmental Member)

Abstract

An Electrocardiogram (ECG) records electrical signals from the heart to detect abnormal heart rhythms or cardiac arrhythmias. Atrial Fibrillation (AF) is the most common arrhythmia which leads to a large number of deaths annually. The diagnosis of heart disease is skill-dependent and time-consuming, therefore using an intelligent system is a time- and cost-effective approach which can also enhance diagnostic accuracy.

This study uses several types of Neural Networks (NNs) including the Deep Neural Network (DNN) GoogLeNet, Multi-Layer Perceptron (MLP), Adaptive Neuro-Fuzzy Inference System (ANFIS), and Long Short-Term Memory (LSTM) to identify arrhythmias in AF signals. The results obtained are compared in order to identify the most effective and accurate system for AF diagnosis. The proposed system has two main steps, preprocessing and postprocessing. In the preprocessing step, different approaches based on the classifier network are used. More specifically, for MLP, ANFIS, and LSTM the 1-D Daubechies wavelet is used, and the extracted wavelet coefficients and statistical features are used as input data to the network. For GoogLeNet, the Continuous Wavelet Transform (CWT) is used to create a time-frequency representation of the signal (scalogram) and extract key signal features. In the postprocessing step, the data obtained (extracted features) are used as the input data to classify the signals. Also, the train and test accuracies and the running times are compared. The results obtained indicate that GoogLeNet provides the best accuracy, but its running time is long. Further, although the ANFIS and MLP networks are much faster than LSTM and GoogLeNet, their accuracy is much lower.

Table of Content

List of Tables

List of Figures

List of Acronyms

ANFIS	Adaptive Neuro-Fuzzy Inference
AF	Atrial Fibrillation
AR	Autoregressive
BR	Branching Programs
CWT	Continuous Wavelet Transform
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DWT	Discrete Wavelet Transform
ECG	Electrocardiogram
EMD	Empirical Mode Decomposition
ICA	Independent Component Analysis
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
ML	Machine Learning
MRA	Mammoth Redundancy Approach
MLP	Multi-Layer Perceptron
NN	Neural Network
PCA	Principal Component Analysis
PNN	Probabilistic Neural Network
STFT	Short-Time Fourier Transform
SVT	Supraventricular
VMD	Variational Mode Decomposition
WNN	Wavelet Neural Network
WT	Wavelet Transform
WVT	Wigner–Ville Transform

Chapter 1: Introduction

Cardiovascular diseases are the leading cause of mortality worldwide [1]. Cardiac arrhythmia contributes to 180,000 to 300,000 sudden cardiac deaths every year in the USA [2]. Evidence shows that with aging, the human cardiovascular system becomes weaker and prone to arrhythmia. An abnormal rate or rhythm of a heartbeat is defined as arrhythmia, and delayed or misdiagnosis of cardiac arrhythmia can lead to serious complications or even heart failure [3], [4]. Atrial Fibrillation (AF) is the most common form of cardiac arrhythmia which affects millions of people around the world. The reported prevalence of AF is around 2% of the general population [4]. This type of arrhythmia accounts for significant morbidity and mortality [5]. For instance, AF is responsible for nearly one in five strokes in people aged over 60 [2]. Such statistics highlight the need for the development of efficient and rapid techniques to enhance the accuracy and accelerate the diagnosis of AF. Several intelligent systems have been developed to improve the accuracy of AF diagnosis and its early detection [6]. In the following section, ECGs and the associated analysis techniques are discussed.

The diagnosis of cardiac arrhythmias involves the analysis and interpretation of cardiac signals. An Electrocardiogram (ECG) provides important information for the diagnosis of heart diseases [7]. However, a careful interpretation of ECG results requires specialized skills and a great amount of time. Despite the time and effort for the accurate interpretation of ECG results, there is still the possibility of human error [8]. Therefore, an intelligent system can be used to increase the accuracy of data analysis while reducing the required time and skills. In recent years, Machine Learning (ML) techniques have been employed

to overcome the above issues [9], [10]. Among all ML techniques, Neural Networks (NNs) have been commonly applied in several fields including medicine and biomedical engineering [11], [12]. In this context, several studies have considered the application of NNs in medical image and ECG signal classification [13], [14]. However, an evaluation of different ML methods is needed in order to determine the most effective approach for identifying arrhythmias in AF signals. Therefore, this thesis considers ML methods for the accurate and rapid diagnosis of arrhythmias in ECG signals. A two-step approach is used to detect arrhythmias in ECG signals. The first step (preprocessing), includes balancing the data, removing noise from the signal, and extracting signal features. The second step (postprocessing) includes implementation, training, and testing of the network.

1.1 ECG Signal and Arrhythmia

Cardiac complications are often associated with a change in the electrical activity of the heart [15]. Measuring this activity is a simple and painless procedure that can help in the prevention, management, and treatment of cardiac diseases. An electrocardiogram (ECG) is a common diagnostic test that visualizes electrical signals generated by the heart. This test is done by placing several electrodes (10-15) on the chest, arms, and legs. These electrodes are connected to electrical leads (wires) which are attached to an ECG machine. The machine records the heart electrical activity and visualizes this information on a graph. Figure 1 illustrates a normal ECG signal [16]. This shows that each heartbeat has 5 waves (P, Q, R, S, and T), which are essential for the identification of a normal heartbeat. Also, the QRS complex, PR segment, ST segment, PR interval, and QT interval are shown. ECG signals and their use in the diagnosis of heart diseases have been studied for over 100 years. In 1887, the first recording of an ECG signal was

done using a capillary electrometer, and electrodes were developed by an English physiologist in 1887 [17]. In 1906, the first esophageal electrocardiogram was recorded [18]. Since 1940, electrocardiography has become the most common method in heart diagnosis [19].

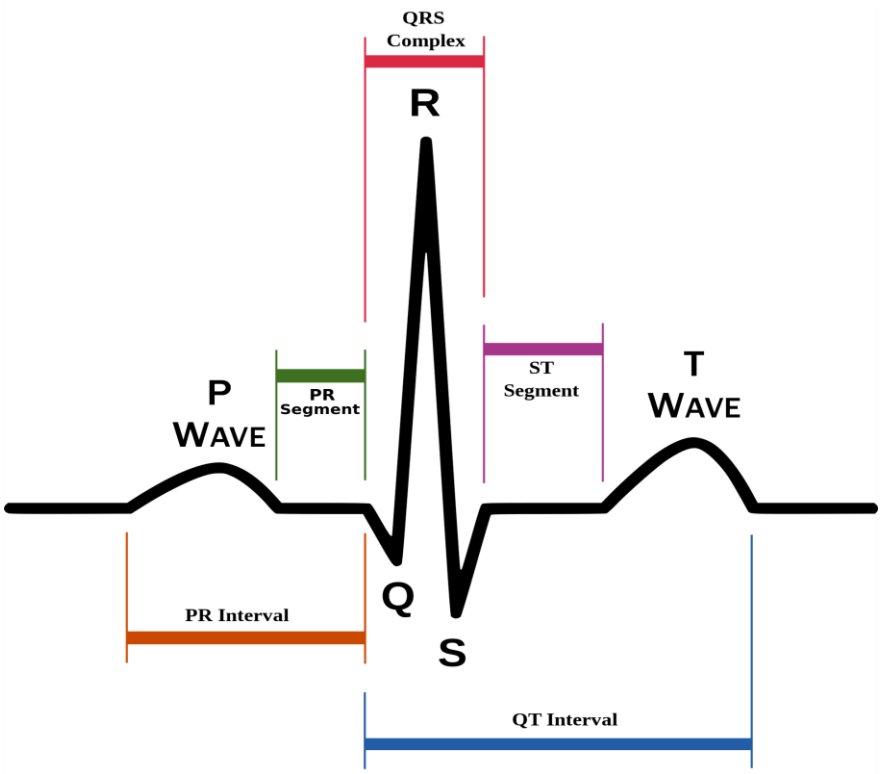


Figure 1. Schematic presentation of a normal ECG signal [20].

As discussed above, an abnormal rate or rhythm of a heartbeat is called an arrhythmia. Heartbeats can be too quick, too slow, or have an irregular pattern. Therefore, arrhythmias are classified based on the rate, mechanism, and duration of the heartbeats. For instance, bradycardia or bradyarrhythmia refers to the condition when the heart beats at fewer than 60 beats/min while tachycardia or tachyarrhythmia denotes accelerated

heartbeats above 100 beats/min. Arrhythmias are also classified based on the location in the heart where the irregular beating occurs. If the irregular heartbeats start in the lower chambers of the heart or the ventricles, they are called ventricular arrhythmias, while arrhythmias beginning in the atria, or upper chambers of the heart, are called Supraventricular (SVT) [20]. This research focuses on the most common form of SVT arrhythmias called AF [4]. In this type of SVT, the atria (the two small upper chambers of the heart), quiver instead of beating effectively. Figure 2 shows ECG signals generated by an AF heart and a healthy normal heart. This illustrates that a normal heartbeat has PQRST waves which are similar in terms of height and distance from each other, while in an abnormal AF heartbeat, the intervals, especially the R-R interval, differ in distance. Some waves can also be missing in AF heartbeats.

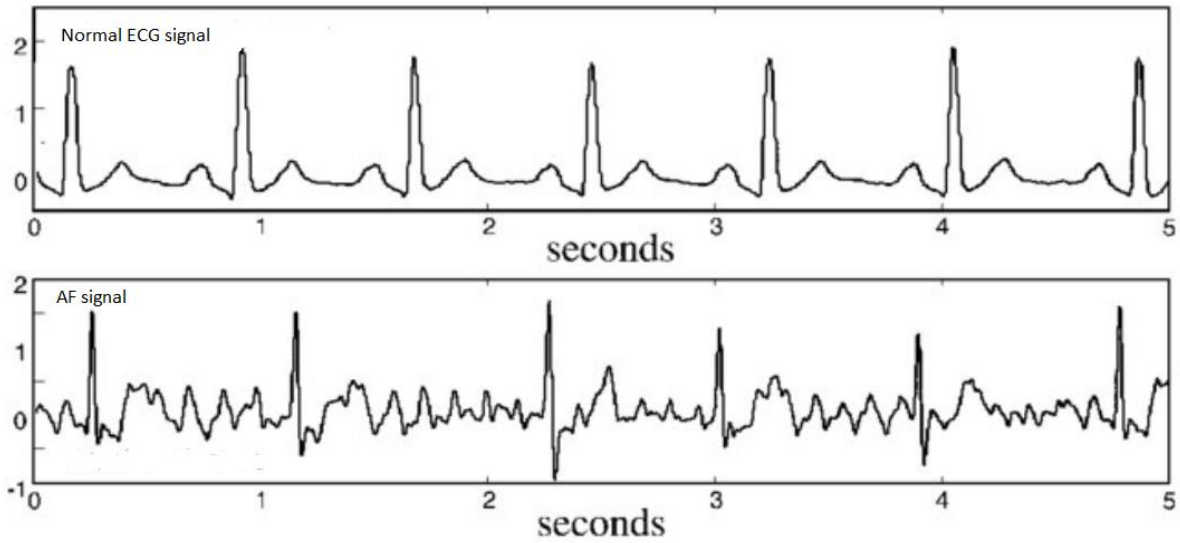


Figure 2. ECG signals from a normal heart and an AF heart [22].

1.2 Arrhythmia Detection Background

A growing number of studies have applied NNs to classify ECG signals in order to facilitate the diagnosis of arrhythmia. In [21], Branching Programs (BR) and NNs were

used to classify ECG signals. In the first method, an Autoregressive (AR) classification algorithm was used to extract AR features. Coefficients of the AR model provide a signal approximation that can be used in automatic feature extraction. Four coefficients of the AR model were used to classify each heartbeat, and a binary decision tree was used to classify the ECG signals. In the second method, an NN with two layers and 6 neurons in each layer was applied for signal classification using the same four AR coefficients as the input. The BR and NN models had an ECG signal classification accuracy of 86.30% and 88.57%, respectively.

A Wavelet Neural Network (WNN) was used in [22] to recognize heartbeat patterns. This study classified ECG data obtained from the MIT-BIH arrhythmia database. The wavelet coefficients for QRS were calculated and then a WNN was used to classify the signals. The results obtained showed a 98.78% accuracy. In [23], the same database was used with wavelet coefficients and the QRS complex as signal features. These features were reduced using factor analysis with and without orthogonal rotation. Then, an LDA classifier was used to classify the signals in the MIT-BIH arrhythmia database and an accuracy of 99.06% was obtained.

A feature reduction method combining Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and a Probabilistic Neural Network (PNN) classifier was proposed in [24] to classify arrhythmia based on ECG beats. In this method, ECG beat samples composed of 200 values (360 Hz sampling rate), around an R peak were extracted from ECG signals. The PNN was then trained to classify different types of ECG beats. The average classification accuracy of the proposed method was 99.71%. In [25], arrhythmias were classified using PCA, LDA, and Independent Component Analysis (ICA)

independently applied on Discrete Wavelet Transform (DWT) sub-bands in order to reduce the number of dimensions. The dimensionality reduced features were then input to Support Vector Machine (SVM) and PNN classifiers. The results obtained showed that a combination of ICA features and a PNN classifier performed better than the PCA and LDA systems. In particular, with ten-fold cross-validation, PNN achieved an average sensitivity, specificity, Positive Predictive Value (PPV), and accuracy of 99.97%, 99.83%, 99.21%, and 99.28%, respectively.

In [26], a new classification method for ECG signals based on a dynamical model of ECG signals was presented. In this method, a fuzzy classifier is used to classify the signals. The simulation results indicated an ECG signal classification accuracy of 93.34%. To improve the performance of this classifier, a genetic algorithm was applied which increased the prediction accuracy to 98.67%. In [27] a one-dimensional Convolutional Neural Network (CNN) was used to classify ECG signals. The CNN was used to divide the ECG classification process into feature extraction and classification blocks. A separate CNN was trained for each patient to improve classification performance. Results showed that the proposed method performs better than previous methods for the classification of ECG beats using the MIT-BIH database.

In [28], two Deep Neural Network (DNN) architectures were proposed for the classification of arbitrary-length ECG signals. The first architecture is a deep CNN with averaging based on feature aggregation across time. The second architecture combines LSTM layers for temporal aggregation of the features and convolutional layers for feature extraction. The efficacy of these networks was evaluated using the AF classification data set provided by Physionet. It was shown that combining these networks increased the accuracy of ECG

signal classification to 82.1%. In [29], the same dataset was used for the classification of short segments of ECG signals. These segments were classified into four groups, AF, normal, other rhythms, and noise. In this study, a state-of-the-art feature-based classifier was compared to a CNN approach. Both methods were trained using the Physionet data. The feature-based classifier obtained an accuracy of 72.0% on the training set with 5-fold cross-validation and 79.0% on the hidden test set. The CNN obtained 72.1% and 83.0% on the augmented database and test set, respectively. The latter method resulted in a final score of 79.0%.

In [30], a CNN technique was presented to automatically detect different ECG segments. The algorithm included an eleven-layer deep CNN with an output layer of four neurons, each representing an ECG class. Two sets of ECG signals were used in this study. The duration of the first group of signals was two seconds while the second group of signals lasted for five seconds. Both types of signals were classified by QRS complex detection. For the first set of signals, the network achieved an accuracy, sensitivity, and specificity of 92.50%, 98.09%, and 93.13%, respectively. For the ECG signals with a duration of five seconds, the accuracy, sensitivity, and specificity were 94.90%, 99.13%, and 81.44%, respectively. In [31], a 1D CNN was used to classify ECG signals. The proposed CNN model consisted of five layers in addition to input and output layers, with two convolution layers, two down sampling layers, and one fully connected layer. This model was used to extract features from ECG signals and classify them into 5 distinct groups of arrhythmia, namely normal, left bundle branch block, right bundle branch block, atrial premature contraction, and ventricular premature contraction. The results were obtained using the MIT-BIH arrhythmia database which contains 48 half-hour excerpts of two-channel

ambulatory ECG recordings obtained from 47 subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979. The results showed that the proposed method achieves a classification accuracy of 97.5% that significantly outperforms several well-known ECG classification methods [31].

In [32], a classification method was introduced for arrhythmia using morphological and dynamic features. A Discrete Wavelet Transform (DWT) was applied to each heartbeat to obtain the morphological features. The RR interval was used as a dynamic feature and the nonlinear dynamics of RR intervals were captured using the Teager Energy Operator (TEO). Moreover, redundancy was removed using an ICA of the DWT subbands. As a result, a total of twelve coefficients were selected as morphological features. Next, the hybrid features were combined and input to an NN for arrhythmia classification. The proposed algorithm was tested using the MIT-BIH arrhythmia and MIT-BIH supraventricular arrhythmia databases. With three-fold cross-validation, this system provided an accuracy of 99.75%. In [33], RhythmNet was developed to automatically classify ECG signals including AF. The RhythmNet architecture is a 21-layer 1D convolutional recurrent neural network with 16 convolution layers to extract features from ECG waveforms followed by three recurrent layers to process ECGs of varying lengths and detect arrhythmia events in long recordings. For training, Physionet was used and large convolutional filters (15 × 1), were applied to effectively learn signal details such as P-waves and QRS complexes within small time frames. An accuracy of 82% in classification was obtained.

In [34], a CNN and DL algorithm was used for the classification of ECG signals. The CNN was trained and tested with the MIT-BIH database. Features were automatically extracted

using a CNN and compared for different activation functions and number of epochs. The results obtained show that the Exponential Linear Unit (ELU) activation function is the best with an accuracy of 93.6%. In [35], an automatic classification system was proposed based on multi-domain features derived from ECG signals in the MIT-BIH arrhythmia database. The feature set has eight Empirical Mode Decomposition (EMD) based features, three Variational Mode Decomposition (VMD) features, and four RR interval features. These 15 features are input to the SVM and C4.5 decision tree classifiers to classify six types of arrhythmias. The proposed method achieved an accuracy of 98.89% compared to the cubic-SVM classifier with an accuracy of 95.35%.

In [36], a 2D CNN model was proposed for the classification of ECG signals into eight classes. This model consists of four convolutional layers and four pooling layers which were designed to extract features from spectrograms. 1D ECG time-series signals are transformed into 2D spectrograms using a short-time Fourier transform. This method achieved a classification accuracy of 99.11% with the MIT-BIH arrhythmia dataset. In [37], an efficient method was developed to predict and select signal features. In this method, the amplitude frequencies including QRS complexes and PR intervals from ECG signals are selected using the Mammoth Redundancy Approach (MRA). The proposed method achieved a 99.75% accuracy with the MIT-BIH arrhythmia database.

1.3 Research Question and Contributions

This thesis presents a comprehensive study of several arrhythmia detection approaches. A comparison of the accuracy and running time of the techniques is provided. The results obtained provide valuable insights into the advantages and disadvantages of each method. Previous work has used data with the same dimensions to identify arrhythmias

in ECG signals. This study focuses on four different systems, namely ANFIS, MLP, LSTM, and DNN. Different feature extraction techniques including CWT and DWT are used to extract features and the accuracies and efficiencies of these methods are compared. Arrhythmia detection is conducted using 2D RGB images and 1D numerical data. Thus, this research provides answers to the following questions.

- How to extract features from ECG signals using wavelet transforms and which wavelet functions have the best performance?
- How does the training data ratio affect the accuracy and running time?
- Which NNs provide the highest accuracy and the lowest running time?
- What are the advantages and disadvantages of each system in classifying ECG signals?

To answer the above questions, different wavelet functions, NNs, and training ratios (10%, 25%, 50%, 75%), are used.

The rest of the thesis is organized as follows. Chapter 2 presents the methodology including wavelet theory and an overview of NNs including MLP, ANFIS, LSTM, and DNN. The preprocessing steps and implementation of each NN are also given. Chapter 3 presents the simulation results obtained using various wavelet functions, training ratios, and NNs, and they are compared in terms of accuracy and running time. Finally, Chapter 4 summarizes the results obtained and presents some future research directions.

Chapter 2: Methodology

In this chapter, an overview of the methods employed is presented. Then the preprocessing steps and identification approaches are discussed in detail. Since the focus is on different approaches for the arrhythmia identification of ECG signals, the preprocessing steps for each network are explained separately.

In MLP, LSTM, and ANFIS preprocessing, after making the signal lengths consistent by truncation, repetition is used to make the number of AF and normal signals equal. Then, data denoising is performed using a low pass filter and a Daubechies wavelet transform is used to extract signal features. These features are input to the MLP, ANFIS, and LSTM for arrhythmia detection. On the other hand, in the preprocessing for GoogLeNet, a CWT is used to generate a scalogram (time-frequency representation of the signals), and then features are extracted from the scalograms. Similar to the other methods, during GoogLeNet preprocessing and before arrhythmias detection, data balancing is performed using repetition. Then, the balanced data are divided into training and testing datasets. For each system, four different training ratios, 10%, 25%, 50%, and 75%, are used and the results obtained are compared. The training ratio is defined as the ratio of training to testing data. Since the preprocessing steps of MLP, ANFIS, and LSTM are different from GoogLeNet, the preprocessing steps are discussed separately. Further, a data set description is provided which also includes an explanation of the data normalizing method.

2.1 Data Description

This study uses the ECG signals in the publicly available Physionet database [40]. This consists of 5788 ECG signals sampled at 300 Hz with 3000 to 18000 samples per record. Out of 5788 records, 738 correspond to AF signals while the remaining 5050 are normal signals.

2.2 Wavelet Functions

The Fourier integral can obscure transient or location-specific features within signals using globally averaged information. To overcome this limitation, a sliding time window of fixed length is used to localize the time-dependent elements. Several time-frequency methods such as the wavelet transform have been commonly used in a variety of areas in science, engineering, and medicine [38], [39].

A wavelet is a small wave with an amplitude that increases from zero to a maximum and then decreases to zero. The advantage of wavelets is their ability to simultaneously illustrate local spectral and temporal information of signals with higher accuracy than a Short-Time Fourier Transform (STFT) [42]. Wavelets effectively transform separate individual signal component by producing a time-frequency decomposition. This provides a local scale-dependent spectral representation of individual signal features with a flexible temporal-spectral aspect and so allows simultaneous capture of low and high-frequency information. Hence, it is particularly useful for the analysis of transient, aperiodic, and other non-stationary signal features. The variety of wavelet functions is a key advantage of these functions. Wavelet transform analysis has been applied to a wide variety of biomedical signals including clinical sounds, respiratory patterns, and ECG signals [38], [40]. Figure 3 shows some Daubechies wavelet functions.

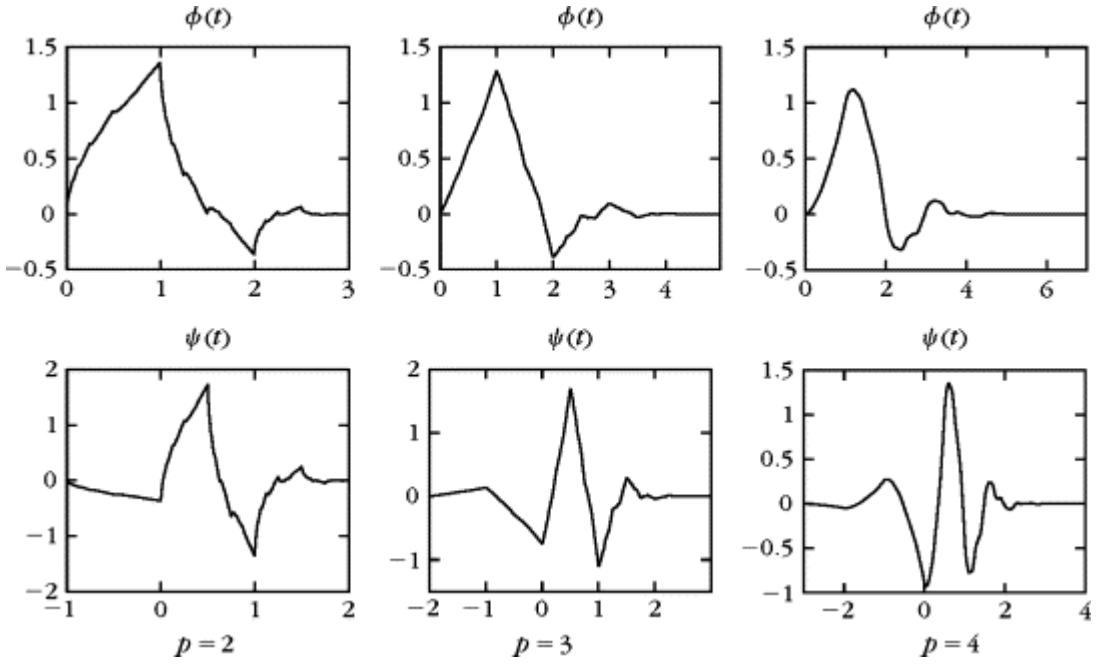


Figure 3. Some Daubechies wavelets [43].

2.3 Wavelet Transforms

The simultaneous interpretation of signals in both the time and frequency by time-frequency signal analysis can increase the clarity of local, transient, or intermittent components. There are several useful Time-Frequency methods for signal analysis such as the Wigner–Ville Transform (WVT), STFT and Wavelet Transform (WT). Since this study focuses on WTs, several different WT methods are introduced below [39], [40].

2.3.1 Continuous Wavelet Transform (CWT)

One of the important differences between an STFT and a CWT is that the CWT is not limited to sinusoidal functions. Rather, a large selection of localized waveforms can be

employed in a CWT. A CWT is a time-frequency method that allows arbitrarily high localization in time of high-frequency signal features. A variable window width is employed to isolate these features. The CWT of a signal $x(t)$ is defined as

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \psi^* \left(\frac{t-b}{a} \right) dt \quad (1)$$

where $\psi(t)$ is the wavelet function, * denotes complex conjugate, and a and b are the dilation and location parameters of the wavelet, respectively.

2.3.2 Discrete Wavelet Transform (DWT)

A DWT employs orthonormal wavelet basis functions and integer power of two scales, a_0 and b_0. This type of wavelet discretization is formulated as

$$\Psi_{m,n}(t) = \frac{1}{\sqrt{a_0^m}} \Psi \left(\frac{t - n b_0 a_0^m}{a_0^m} \right) \quad (2)$$

where the wavelet dilation and translation are controlled by the integers m and n, respectively. b_0 is the location parameter which must be greater than zero and a_0 is a fixed dilation step parameter set to a value greater than 1. For a DWT, the most common values for a_0 and b_0 are 2 and 1, respectively [39], [41].

2.4 Neural Networks (NN)

A Neural Network (NN) can simulate the human brain in order to solve problems and tasks needing an intelligent system. The human brain works like a highly complex and nonlinear computer to process information [42]. Like the human brain, a large group of interconnected computing cells referred to as neurons or processing units are employed in an NN in order to achieve good performance.

Moreover, in the human brain, a phenomenon called plasticity allows a developing nervous system to adapt to the surrounding environment. Therefore, plasticity counts as an essential feature of the brain which allows neurons to function as information processing units. This phenomenon is also present in the artificial neurons of an NN. As a result, the most important advantages of an NN are self adaptation, self organization, and real time operation. In order to achieve the above characteristics, learning algorithms are used to perform the learning process. These algorithms modify the weights of the network in order to attain the desired design objective. The traditional method to modify the NN weights is an adaptive linear filter [43]. An NN can also modify its topology as in the human brain, where neurons can die, and new neuron connections can grow [42], [43].

2.4.1 Multi-Layer Perceptron (MLP)

An MLP is a type of NN that has been widely used in various classification tasks including speech recognition, drug diagnosis, image processing, and signaling [44]. The MLP structure consists of an input layer, hidden layers, and an output layer. Each layer contains one or more processing elements (neurons) which are connected to the neurons of the next layer through weighted connections. The input data is mapped to the neurons of the input layer and neurons of the output layer are mapped to the output of the model. The number of variables specifies the number of neurons in the input and output layers. However, a trial and error method is typically used to determine the number of neurons in the middle layers. Figure 8 shows the architecture of an MLP network [44].

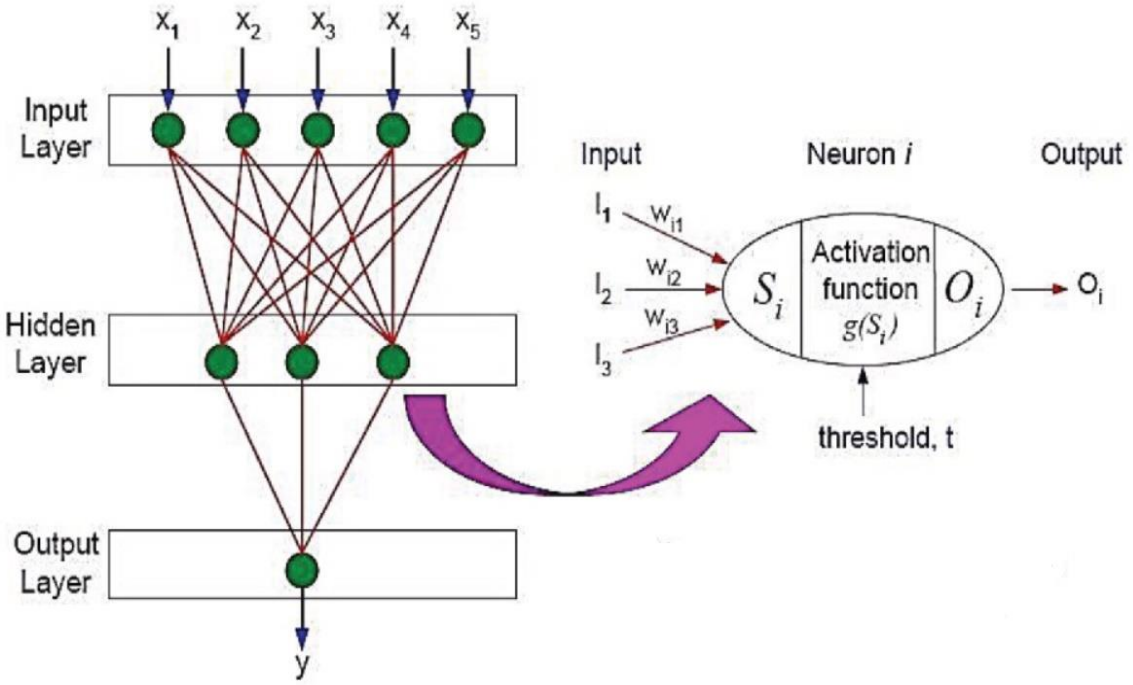


Figure 4. The architecture of an MLP network [47].

The neurons in each layer are connected to neurons in the previous layer through directional connections. To determine the effect of neurons of the previous layers on the neurons in the output layer, each connection is given a weight W_{ij}. To calculate the output of a neuron, the weighted input values are used with an activation function. Linear and sigmoid functions are the most common activation functions used in multi-layer networks [43].

The weights of the network neurons are determined by training through a backpropagation algorithm. First, the network output is calculated by assigning random values to the neuron weights. Then, the error between the network output and the desired output is calculated. This error is used to adjust the weight values. This process is iterated until the network output reaches a desired accuracy.

The error function is

$$E = \sum_0^n (z_0 - y_0) \quad (3)$$

and the weight update function is

$$W_{ij_{(t+1)}} = W_{ij}(t) - \mu \frac{\sigma E}{\sigma W_{ij}} + \beta \ (W_{0j}(t) - W_{ij}(t-1)) W_{ij} \quad (4)$$

where z_0 is the actual output of the last layer, y_0 is the expected output and $W_{ij}(t)$ is the weight of the connection of node i to node j in iteration t. β and μ are constants called the learning rate and intensity rate, respectively [42], [43]. Once the desired error is obtained, the training is stopped. After the network is trained, it is evaluated using the testing data which was not used in training. The network performance is assessed using the accuracy and running time [43], [45].

2.4.2 Adaptive Neuro-Fuzzy Inference System (ANFIS)

Fuzzy-based systems that use if-then rules can model and infer the qualitative aspects of knowledge. This section introduces the concepts of fuzzy sets, fuzzy rules, and fuzzy inference systems.

2.4.2.1 Fuzzy Inference System (FIS)

A FIS is known by the names fuzzy law-based system, fuzzy expert system, fuzzy-related memory or fuzzy controller. Figure 5 shows the five FIS functions which are given below.

1. Fuzzification of the input variables

2. Application of the fuzzy operator (AND or OR) in the antecedent

3. The implication from the antecedent to the consequent

4. Aggregation of the consequents across the rules

5. Defuzzification

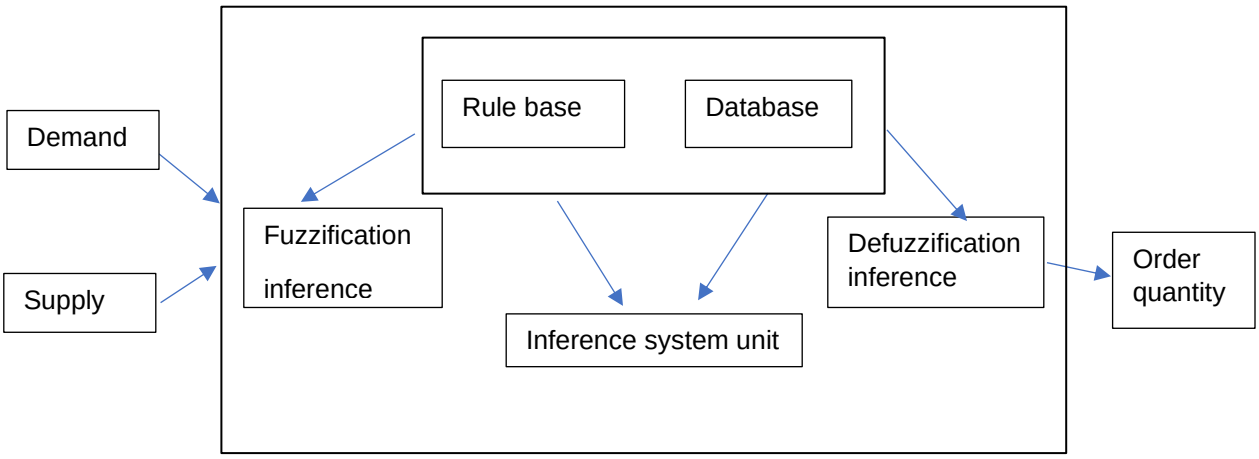


Figure 5. FIS system structure [49].

2.4.2.2 Combination of Neuro-Fuzzy Systems

To benefit from the advantages of both systems, an NN and FIS are combined to generate a neuro-fuzzy system. NNs are automatically trained by learning algorithms until a suitable output is obtained. Because the input and output data are combined by the weights in the network, knowledge is not clearly expressed. This problem is called the NN black box phenomenon which can slow the learning process. This means the knowledge

(rules) cannot be extracted from the trained NN and the previous knowledge gained from the problem cannot be aggregated in the form of if-then rules.

A FIS is desirable because its behavior is described through fuzzy rules. It is also easy to build and includes previously acquired knowledge. This requires defining membership functions, fuzzy operators, and knowledge bases. Finding the right membership functions and fuzzy rules is often a tedious trial and error process which is difficult for large databases. The goal of fuzzy-neural composition is to design a system that uses a fuzzy system to display knowledge and has the learning capability of an NN so it can function as a member and adjust the parameters to improve system performance [46].

The combinations of NN and fuzzy systems can be divided into two groups.

• NNs equipped with fuzzy capabilities. In other words, the NN is the basic structure while the FIS is the second structure. With this combination, the NN can be fuzzy which allows processing fuzzy inputs or speeding up learning by using fuzzy techniques. This hybrid system is called a Neuro-Fuzzy Network (NFN), in which the network inputs/outputs or weights are all fuzzy sets.

• NNs added to FIS. In this structure, FIS is the base structure and the NN is the second structure. This is called a Neuro-Fuzzy System (NFS). In this system, the NN is used to provide inputs to the fuzzy system or to change the output of the fuzzy system. In this study, a Neuro-Fuzzy System called ANFIS is used.

2.4.2.3 Adaptive Neuro-Fuzzy Inference System (ANFIS)

The Adaptive Neuro-Fuzzy Inference System (ANFIS) was first introduced in [47]. This system can approximate any real continuous function on a compact set to any degree of accuracy. This model also has good training and categorizing capabilities while allowing extraction of fuzzy rules from numerical data and production of an adaptive rule-based system. ANFIS is a Sugeno-type fuzzy system, and the fuzzy if-then rules of this system are [48]

$$\text{If } (x \text{ is } A_1) \text{ and } (y \text{ is } B_1), \text{then } f_1 = p_1 x + q_1 y + r_1 \quad (5)$$

$$\text{If } (x \text{ is } A_2) \text{ and } (y \text{ is } B_2), \text{then } f_2 = p_2 x + q_2 y + r_2 \quad (6)$$

where the inputs are x, y, the output is f, and p_i, q_i, r_i are design parameters determined during the training process. A_i and B_i are fuzzy sets based on the membership functions. Figure 6 shows a five-layer ANFIS model with two inputs and two fuzzy rules.

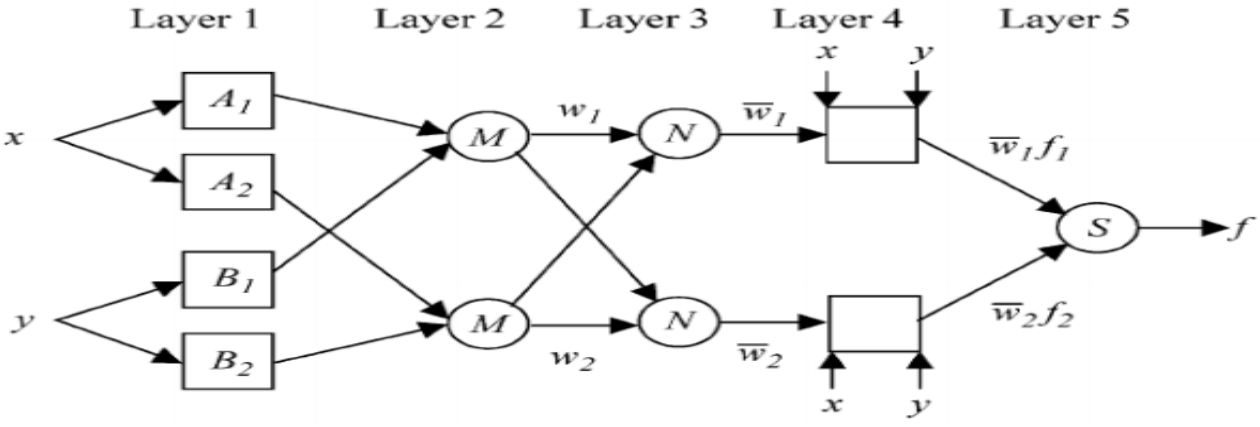


Figure 6. An ANFIS model [53].

In ANFIS, the backpropagation algorithm uses the gradient descent method to train the membership functions. The first hidden layer is used for the fuzzification of input variables. The outputs of the first layer are the fuzzy membership degrees of the input variables which are given by

$$O_i^1 = \alpha_{A_i}(x), \quad i = 1, 2 \tag{7}$$

$$O_i^1 = \alpha_{B_{i-2}}(y), \quad i = 3, 4 \tag{8}$$

There are M nodes in the second layer. The output of the second layer is

$$O_i^2 = w_i = \alpha_{A_i}(x)\, \alpha_{B_i}(y), \quad i = 1, 2 \tag{9}$$

where w_{i} is the firing strength. In the third layer, there are N nodes as shown in Figure 6. The task of this layer is to normalize the intensity of the second layer rules.

The results of the fuzzy rules are defined in the fourth layer. The components defined in the fuzzy rules are obtained from

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i), \quad i = 1, 2 \tag{10}$$

The fifth layer contains one node which calculates the output given by

$$O_i^5 = \sum_{i=1}^{2} \overline{w}_i \; f_i \; \frac{(\sum_{i=1}^{2} w_i f_i)}{w_1 + w_2} \tag{11}$$

The adaptive layers in the ANFIS architecture are the first and fourth. There are three modifiable parameters in the first layer called the basic parameters $a_i,\ b_i,\ c_i$, which are dependent on the form of the membership function. In the fourth layer, there are three modifiable parameters $p_i,\ q_i,\ r_i$, called result parameters. The ANFIS training process adjusts these six parameters so the model can produce an appropriate output. The ANFIS output is

$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 = \bar{w}_1 f1 + \bar{w}_2 f2 \quad (12)$$

2.4.3 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) were introduced in 1986 [49]. They are sequence-based models that establish the temporal relationships between the current state and previous information so the decision at time step $t - 1$ can affect the decision at time step t. RNNs are trained by backpropagation. The LSTM architecture was introduced with a memory cell and further improved with an extra forget gate. It is the most successful RNN architecture and has been employed in many applications [50], [51].

2.4.3.1 The LSTM Model

LSTM networks have been used for language modeling, speech-to-text transcription, machine translation, and other applications. The architecture of the LSTM network depends on the task. For regression tasks, the network includes a sequence input layer, an LSTM layer, and a fully connected layer follow by an output regression layer, while a classification network ends with a fully connected layer, a softmax layer, and a classification output layer. The LSTM network architecture for classification is considered here and is shown in Figure 7.

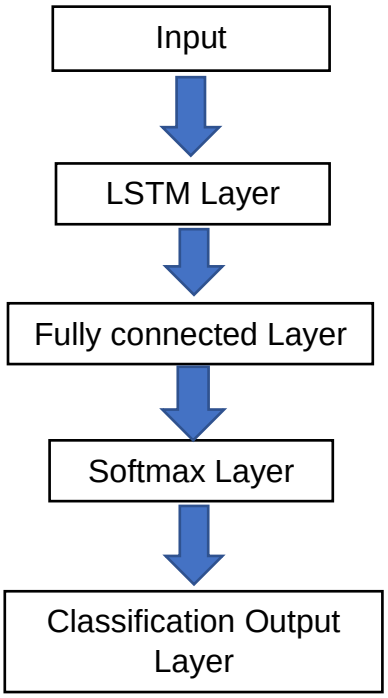


Figure 7. The LSTM architecture.

Let $\{x_1, x_2, \ldots, x_t\}$ be the LSTM input sequence, where $x_t \in R^k$ is the *k*-dimensional vector of real values at time step t. An LSTM block has a memory cell that establishes temporal connections. The state s_{t-1} of this memory cell is linked to the intermediate output h_{t-1} and the current input x_t in order to specify which elements of the internal state vector need to be updated, maintained, or removed. In addition to the internal state, an input node g_t, input gate i_t, forget gate f_t, and output gate o_t are also in an LSTM block and are given by

$$f_t = \sigma\left(W_{fx}\, x_t + W_{fh}\, h_{t-1} + b_f\right) \quad (15)$$

$$i_t = \sigma\left(W_{ix}\, x_t + W_{ih}\, h_{t-1} + b_i\right) \quad (16)$$

$$g_t = \phi\left(W_{gx}\, x_t + W_{gh}\, h_{t-1} + b_g\right) \quad (17)$$

$$o_t = \sigma\left(W_{ox}\, x_t + W_{oh}\, h_{t-1} + b_o\right) \quad (18)$$

$$s_t = g_t \odot i_t + s_{t-1} \odot f_t \quad (19)$$

$$h_t = \phi\left(s_t\right) \odot o_t \quad (20)$$

where $W_{gx}, W_{gh}, W_{ix}, W_{ih}, W_{fx}, W_{fh}, W_{ox},$ and W_{oh} are weight matrices and $\odot$ represents element-wise multiplication, ϕ is the tanh function, and σ is the sigmoid activation function. Figure 8 illustrates the LSTM block structure at time step t [51].

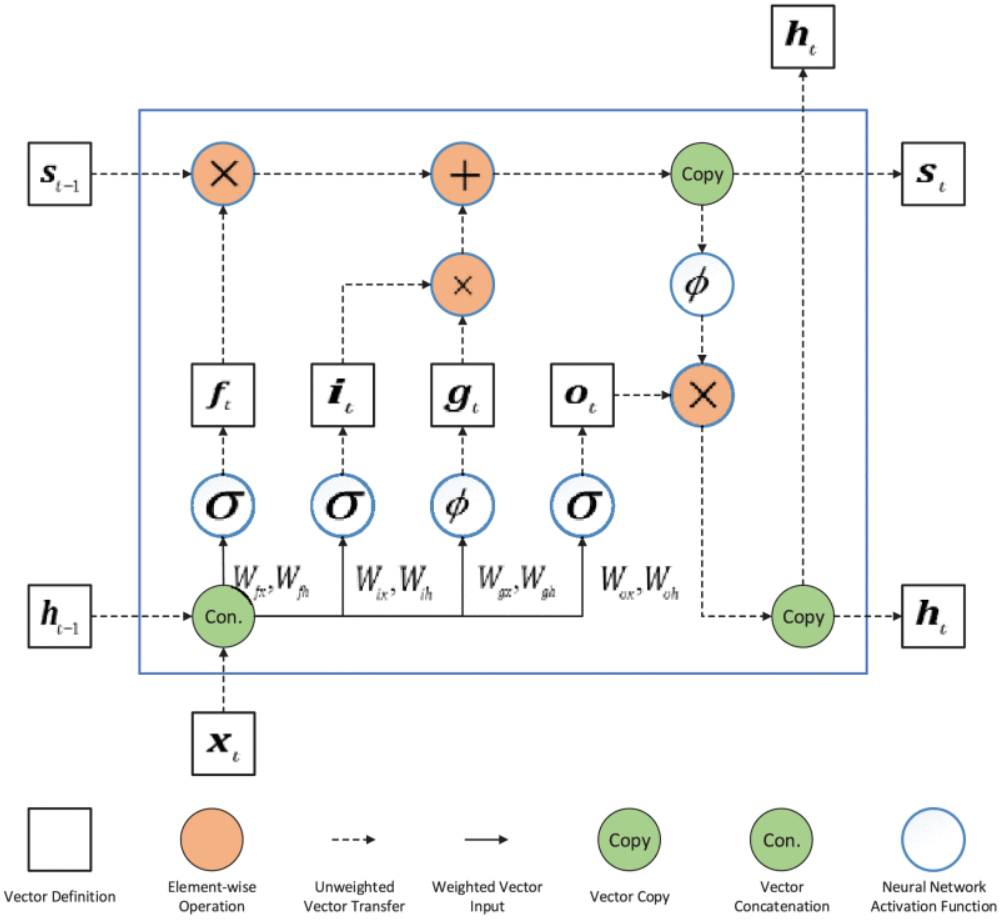


Figure 8. The LSTM block structure at time step *t* [56].

In an LSTM block, there are three sigmoid functions with an output range 0 to 1 which decide if signals can pass the gates. The decision for input gate i, forget gate f, and output gate o depends on the present input x_t and the last output h_{t-1}. The input gate signal controls what is kept in the internal state whereas the forget gate decides what is forgotten from the previous state s_{t-1}. In updating the internal state, the output gate chooses which internal state s_t is the LSTM output h_t. This process is repeated in the next time step. The weights and biases are learned by minimizing the differences between the training data and the LSTM outputs [51].

2.4.4 Deep Learning (DL)

Deep Learning (DL) is an ML technique which employs multiple NN layers [52]. A CNN is the most common form of DNN and has been successfully applied in the analysis of images. A widely used method to improve DNN performance is increasing the network size including the number of layers (depth) and neurons in each layer (width). This has several disadvantages such as increasing the number of training parameters and sequences which requires more computational resources. To solve these problems, a sparsely connected CNN architecture can be used. Figure 9 shows a CNN with convolutional layers, pooling layers, and fully connected layers. In the following, each type of layer is briefly explained [53], [54].

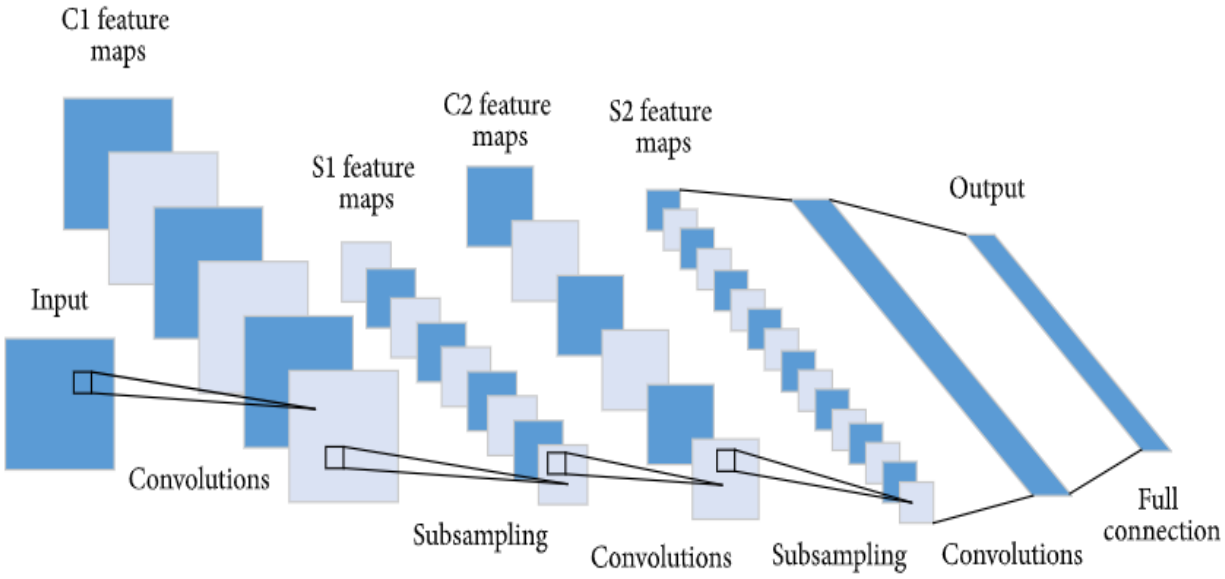


Figure 9. A CNN architecture [62].

2.4.4.1 Convolutional Layers

The input of a CNN is an array of vectors with size the number of input images × image height × image width × number of input channels. By passing the input through a convolutional layer, the image is converted into a feature map with size the number of images × feature map height × feature map width × number of feature map channels. In general, there are several hyperparameters for each convolutional layer within an NN, including the width and height of the convolutional kernels, number of input and output channels, and depth of the convolution kernel. The input is convolved by a convolutional layer and transferred to the next layer [55].

2.4.4.2 Pooling Layers

A CNN can include a pooling layer to reduce data dimensions by combining the outputs at a layer with a single neuron in the next layer. Global pooling works on all neurons of the convolutional layer while local pooling combines small clusters of neurons. There are two common types of pooling, max and average. Average pooling uses the average value of the neuron cluster while max-pooling uses the maximum value of the cluster [55].

2.4.4.3 Fully Connected Layers

Fully connected layers connect every neuron of one layer to all neurons of another layer. These layers are a crucial component of CNNs. Results obtained from different steps of the CNN process including convolution and pooling are fed to a fully connected layer to perform the final classification [55].

2.4.4.4 GoogLeNet

GoogLeNet is a 22-layer deep CNN that contains an input layer, 5 convolutional layers, several subsampling layers, and an output layer. In addition to these layers, there are 5 pooling layers including four max-pooling layers and one average pooling layer. Average pooling with filter size 5x5 and stride 3 is used before the classifier. Stride is defined as the shift of the filter/sliding window over the input image. The GoogLeNet architecture has 9 inception layers which perform different size convolutions and concatenate the filters for the next layer. Pre-trained versions of GoogLeNet have been trained on the ImageNet and Places365 datasets [40]. The network trained on Places365 classifies images into 365 different place categories such as field, park, runway, and lobby while the network trained on ImageNet classifies images into 1000 object categories such as mouse, pencil, and various animals. Both pre-trained networks have an image input size of 224x224 and have learned features for a wide range of images [56]. As discussed above, improving the efficiency of the network by increasing the size of the network results in an increase in the number of parameters and computational cost. To solve this problem, GoogLeNet utilizes an inception module that uses a parallel combination of 1x1, 3x3, and 5x5 convolutions to reduce the amount of data before the expensive 3x3 and 5x5 convolutions [54], [57]. Figure 10 shows one of the nine inception modules and the GoogLeNet architecture.

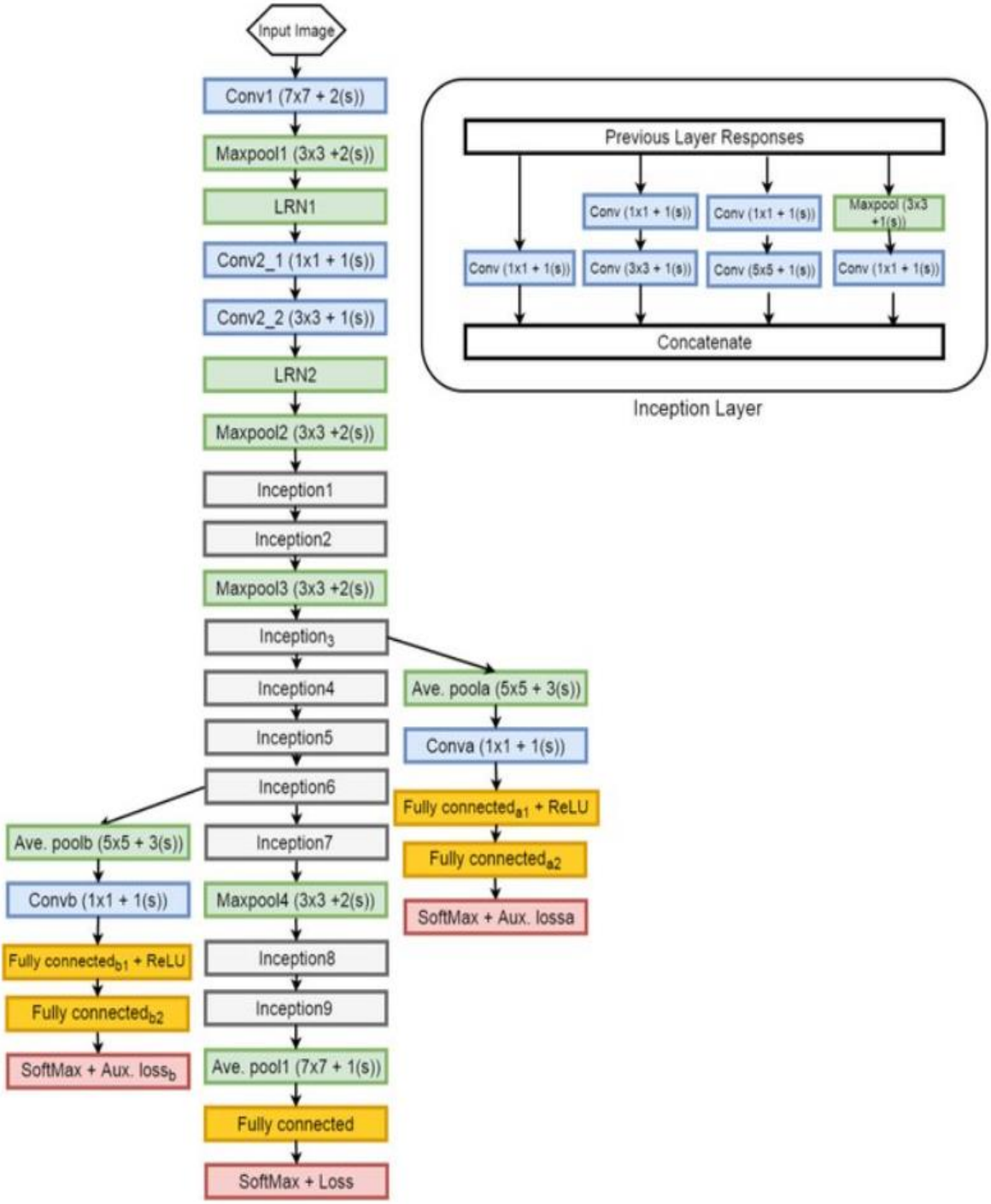


Figure 10. An inception module and the GoogLeNet architecture [62].

2.5 Proposed Method

In this section, the preprocessing and postprocessing methods are introduced. Each step is explained and the motivation is given. Figure 11 shows the proposed framework for the MLP, ANFIS, and LSTM systems.

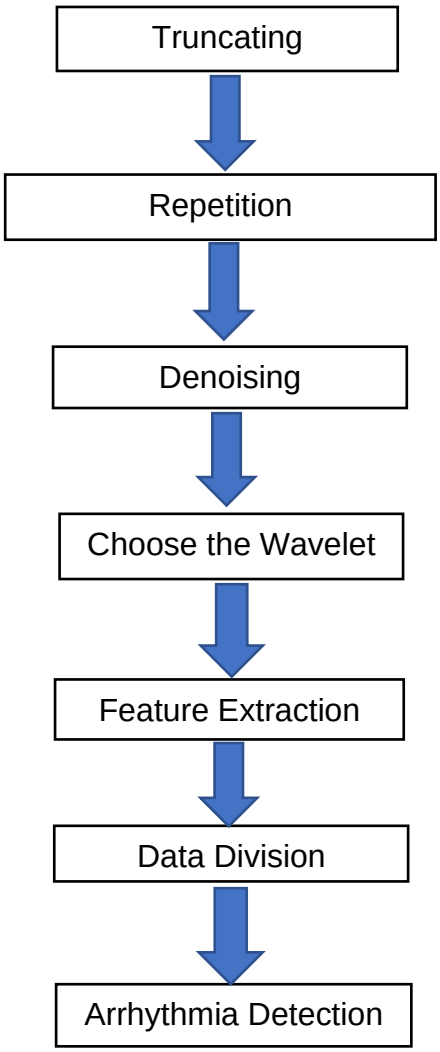


Figure 11. Framework for the MLP, ANFIS, and LSTM systems.

2.5.1 Preprocessing

Preprocessing is used to prepare the data for classification which includes noise reduction, feature extraction, and data division.

2.5.1.1 Balancing the Dataset

The dataset includes 5788 records containing 5000 to 18000 samples per record. To balance the signal lengths, those with a length over 9000 samples are truncated to 9000, and shorter signals are discarded. This signal length was selected based on the histogram of the data in Figure 12 which shows that the majority of signals have a length of 9000 samples. There are different methods to balance the number of AF and normal signals such as Synthetic Minority Oversampling (SMOTE) and down sampling the majority class. In [58], a repetition method was used for validation data. In this study, the AF signals are repeated to obtain an equal number of normal and AF signals. The results obtained show that there are small differences between training and testing accuracies so this repetition does not cause overfitting. Overfitting is indicated when the training accuracy is much better than the testing accuracy.

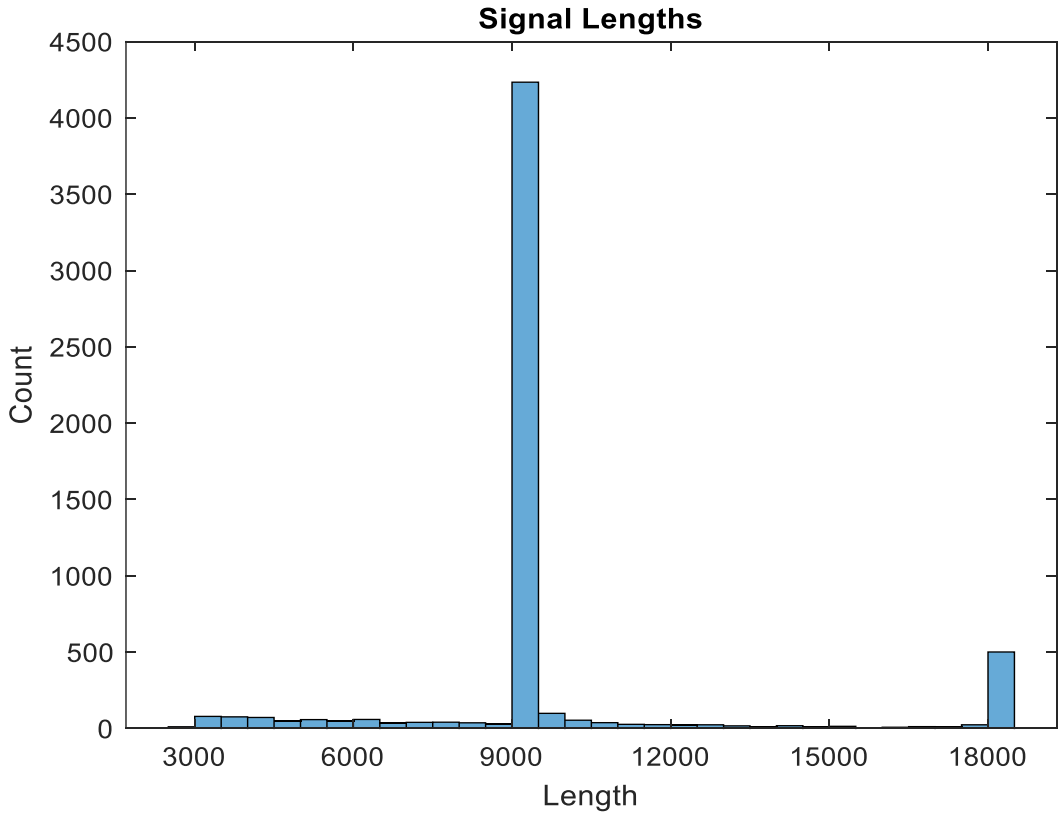


Figure 12. Histogram of the ECG signals.

2.5.1.2 Noise Reduction

A low pass filter is used to eliminate high-frequency noise. In this thesis, a Savitzky-Golay filter is used for this purpose. This smooths the data without distorting it [59].

2.5.1.3 Choosing the Wavelet Function

The three most commonly used wavelet functions, namely Haar, D4, and D10, are used to extract signal features. The reason for using these wavelets is that they have been widely used for signal processing and their shapes differ from each other. To determine the most suitable wavelet, the extracted features are fed as input data to an MLP. Then,

the results obtained are compared and the wavelet with the highest accuracy is chosen. These results are shown in Chapter 3.

2.5.1.4 Feature Extraction

Feature extraction is performed using the DWT method and the D10 wavelet function. This thesis uses 4 decomposition levels so the wavelet is applied or shifted 4 times through each signal. The number of decomposition levels is chosen based on the information provided from each level. Each decomposition level provides an approximation (a) and details (d) of the signal. Figure 13 shows the approximations and details for 4 decomposition levels and compares them to the original signal. The original signal, approximations, and details are shown in red, blue, and green, respectively.

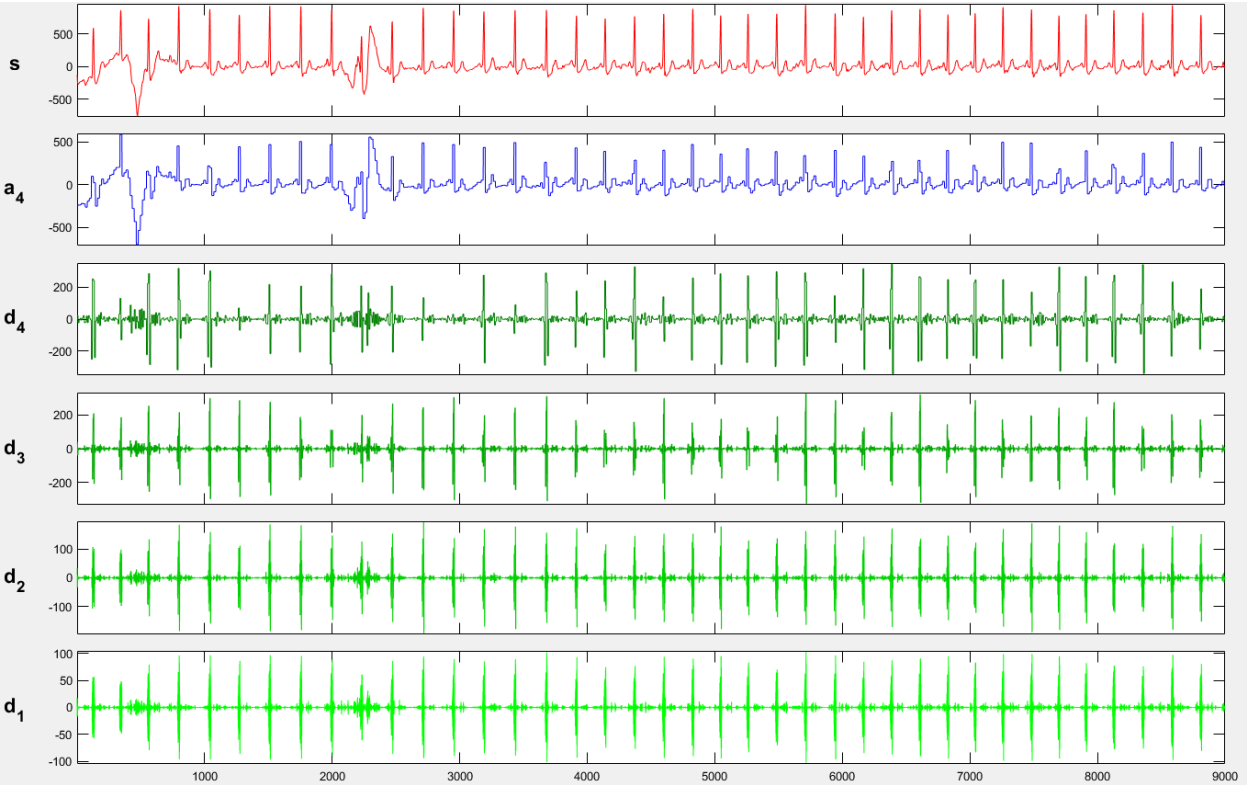


Figure 13. Wavelet decomposition.

In this thesis, the 8 coefficients obtained from the approximations and details of the 4 decomposition levels are used as features. Statistical features, namely mean, standard deviation, skewness, kurtosis, and entropy of the signal are also used. Observation of these features showed that the mean and kurtosis are in the range 2×10^{-8} to 5×10^{-8} which are almost zero. This means that they do not provide sufficient information to be used as features. Therefore, the number of features is reduced to 3 statistical features and 8 wavelet coefficients (11 in total). Once the features are extracted from each signal, the data are ready to be used as input for the ANFIS, MLP, and LSTM classifiers.

2.5.1.5 Data Division

In this study, four different training ratios, namely 10%, 25%, 50%, and 75%, are used and the results compared. To use different data in each run, the data are shuffled before division, and the training and testing data are chosen randomly.

2.5.2 Postprocessing

Postprocessing includes implementation, training, and testing the network. In the following, these steps are discussed for each network.

2.5.2.1 ANFIS

To generate the ANFIS model, Genfis 3 is used which is a Sugeno-type inference system that employs a Fuzzy C-Means (FCM) clustering algorithm [47]. FCM is based on the minimization of an objective function that allows each data point to belong to multiple clusters with different degrees of membership. After implementation by Genfis 3, the network is trained using training data.

2.5.2.2 MLP

This thesis uses an MLP composed of 2 hidden layers, each containing 4 neurons. For implementation, the activation functions of the hidden and output layers must be specified. Tansig and Pureline are used as the hidden layer activation function and output layer activation function, respectively [44]. With this model, the data is divided into train, test, and validation datasets.

2.5.2.3 LSTM

The LSTM consists of a sequence input layer of size 11, a bidirectional LSTM layer, and 2 fully connected layers followed by a SoftMax activation function and a classification layer. The size of the sequence input layer is the same as the number of input dimensions. A bidirectional LSTM layer with 100 hidden units and a classification output layer employing cross-entropy as the loss function are used. After implementing the LSTM, the data is divided into train and test datasets.

2.5.2.4 GoogLeNet

This study uses a pre-trained CNN called GoogLeNet with 22 layers that was trained to classify images into 1000 object categories. Thus, to identify arrhythmias in the ECG signals, GoogLeNet should be retrained. To prevent overfitting, a dropout layer is used which randomly sets input elements to zero with a given probability. The final classification layer and the last learnable layer are used to classify images. These layers combine features that the network extracts into class probabilities, a loss value, and predicted labels. They are replaced with new layers adapted to the ECG data. The last

learnable layer is replaced with a fully connected layer with the number of filters equal to the number of classes. The classification layer is replaced with a layer without class labels. Once GoogLeNet is retrained, a CWT is used to generate a scalogram in order to extract signal features. A scalogram is the absolute value of the CWT coefficients of a signal. The scalograms are saved as RGB images with an array of size 224x224x3. Then the signal features are automatically extracted by the first layers of GoogLeNet. After feature extraction, network training is done using the training data. In contrast to the MLP, ANFIS, and LSTM, in GoogLeNet, the data division step is done before feature extraction.

2.5.2.5 Testing and Validation

There are several approaches for testing and validating ML systems. One of the main issues which occurs in training is overfitting. This means that the model closely fits the training data but is not accurate for the testing data. In other words, the training accuracy is much better than the testing accuracy. Train/test split and cross-validation are commonly used to prevent or minimize overfitting. In train/test split, the dataset is divided into separate training and testing sets for training and testing. Cross-validation is similar to train/test split but uses more subsets as it splits the data into *k* subsets. Training is done using *k*-1 subsets and the last is used for testing. This approach is applied with each subset used for testing [60]. In this study, the train/test split method is used.

Chapter 3: Results and Discussion

In this chapter, the signal features extracted using different wavelet functions are compared, and features achieving the highest accuracy are used in the rest of the thesis. Then, the extracted features are input to the ANFIS, MLP, LSTM, and GoogLeNet systems, and results are obtained for various training ratios. Finally, these results are compared in terms of accuracy, time, and the number of iterations.

3.1 Wavelet Comparison

The features obtained using the 3 wavelet functions Haar, D10, and D4 are used as input to the MLP classifier. Then, the wavelet features with the highest accuracy are used in the remainder of this thesis. For this, an MLP classifier with a training ratio of 50% is used, and the average for 10 runs is considered. Tables 1 to 3 show the results obtained for each wavelet function. To validate the results and show the statistical significance, a single factor ANOVA test was done, and the results are given in Table 4.

Value	Time (s)	Training Accuracy (%)	Testing Accuracy (%)	Number of Iterations
Min	10.4	75.5	75.1	1000
Max	10.2	86.4	86.2	1000
Average	10.2	82.2	80.8	1000

Table 1. Results for the Haar wavelet function.

Value	Time (s)	Training Accuracy (%)	Testing Accuracy (%)	Number of Iterations
Min	10.2	87.6	84.8	1000
Max	10.7	89.9	89.5	1000
Average	10.2	89.0	87.9	1000

Table 2. Results for the D10 wavelet function.

Value	Time (s)	Training Accuracy (%)	Testing Accuracy (%)	Number of Iterations
Min	10.9	78.6	79.2	1000
Max	10.2	89.3	88.6	1000
Average	8.1	84.6	82.5	1000

Table 3. Results for the D4 wavelet function.

Wavelet Function	Count	Sum	Average	Variance		
D10	10	879	87.9	1.60		
D4	10	824	82.5	14.1		
Haar	10	808	80.8	17.8		
Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Between Groups	277	2	138	12.4	0.000151	3.35
Within Groups	302	27	11.2			
Total	580	29				

Table 4. The ANOVA test results.

Tables 1 to 3 show that the classification accuracy of the D10 wavelet is higher than with the D2 and Haar wavelets. Therefore, in the rest of this thesis, the D10 wavelet is used for feature extraction. Table 4 shows the one-way Analysis of Variance (ANOVA) test

results for the three wavelet functions where SS, df, and MS denote sum of squares, degrees of freedom, and mean square, respectively. In an ANOVA test, significant differences among group means are calculated using the F statistic, which is the ratio of the mean sum of squares to the mean square error. If the F statistic is higher than the critical value (the value of F that corresponds with the P-value), then the difference among the groups is deemed statistically significant.

The F value 12.4 is bigger than F critical 3.35. Thus, it can be concluded that the results are statistically significant so there is a statistical difference in mean score among the results for the wavelet functions. Furthermore, the lower variance of D10 compared to the other two wavelet functions supports the hypothesis.

3.2 Arrhythmia Detection Results

In this section, the results obtained for the four systems with training ratios 10%, 25%, 50%, and 75% are provided. For each number of iterations, each system was run 10 times, and the standard deviation, and training and testing accuracies for the runs are presented using box plots. The average running time, and training and testing accuracies are presented in the corresponding tables. The boxplots illustrate the distribution of data based on minimum, first quartile, median, third quartile, and maximum values, and circles show the outliers. For LSTM, the accuracy versus the number of epochs is shown. An epoch consists of a number of iterations and the number of iterations in an epoch depends on the training ratio and minibatch size.

3.2.1 10% Training Ratio

The results obtained for the MLP, ANFIS, LSTM, and GoogLeNet systems with a 10% training ratio are presented in Figures 14 to 19. The average accuracies and running

times with different numbers of iterations for the MLP, ANFIS, and LSTM systems are given in Tables 5 to 7. For LSTM, an epoch is 28 iterations. However, for GoogLeNet, due to its high running time, 10 runs for each number of iterations was not feasible, so the results for only a single run per number of iterations are presented in Table 8.

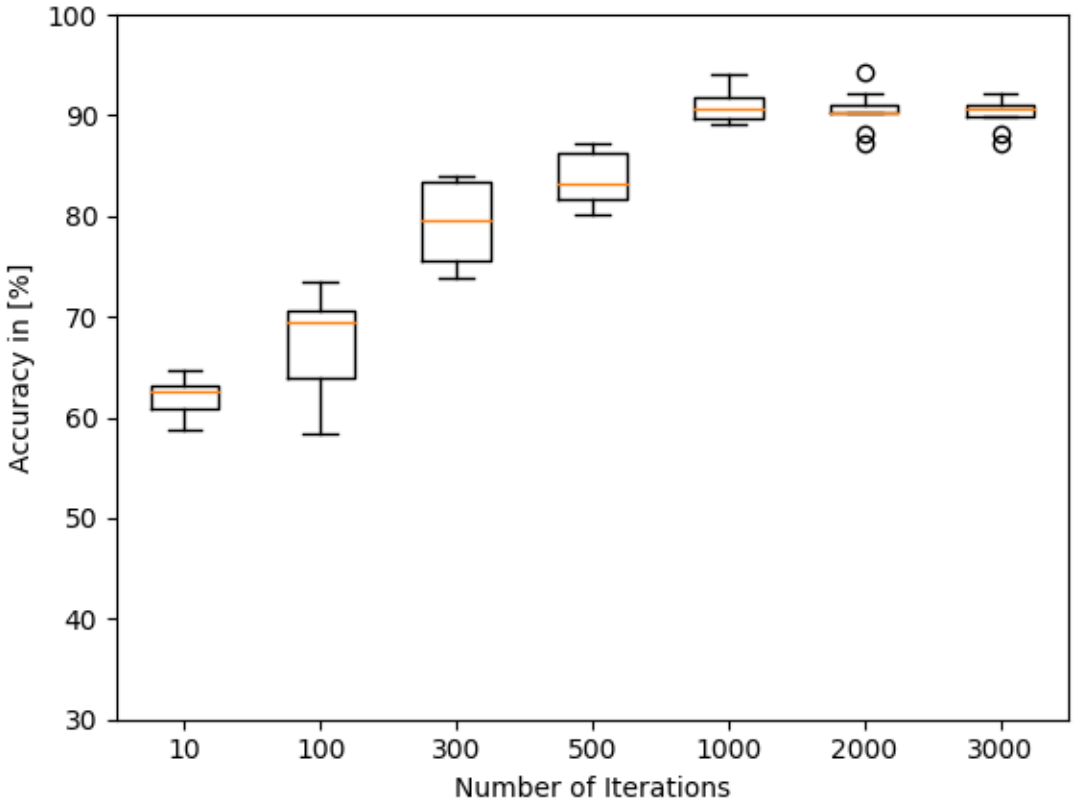


Figure 14. Training accuracy versus the number of iterations for MLP with a 10% training ratio.

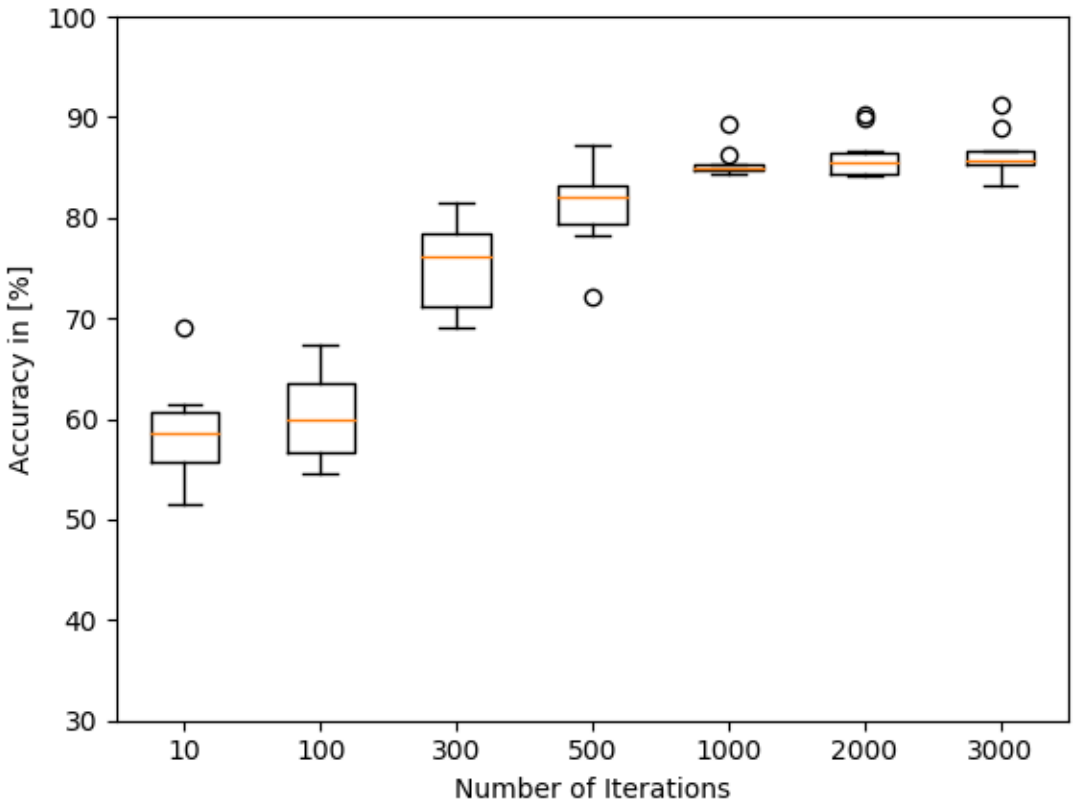


Figure 15. Testing accuracy versus the number of iterations for MLP with a 10% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.1	61.2	58.3
100	2.3	65.9	60.2
300	4.2	81.2	61.5
500	6.7	83.7	81.2
1000	9.8	90.9	85.5
2000	14.2	90.5	86.1
3000	26.8	90.3	86.2

Table 5. Average training and testing accuracies for MLP with a 10% training ratio.

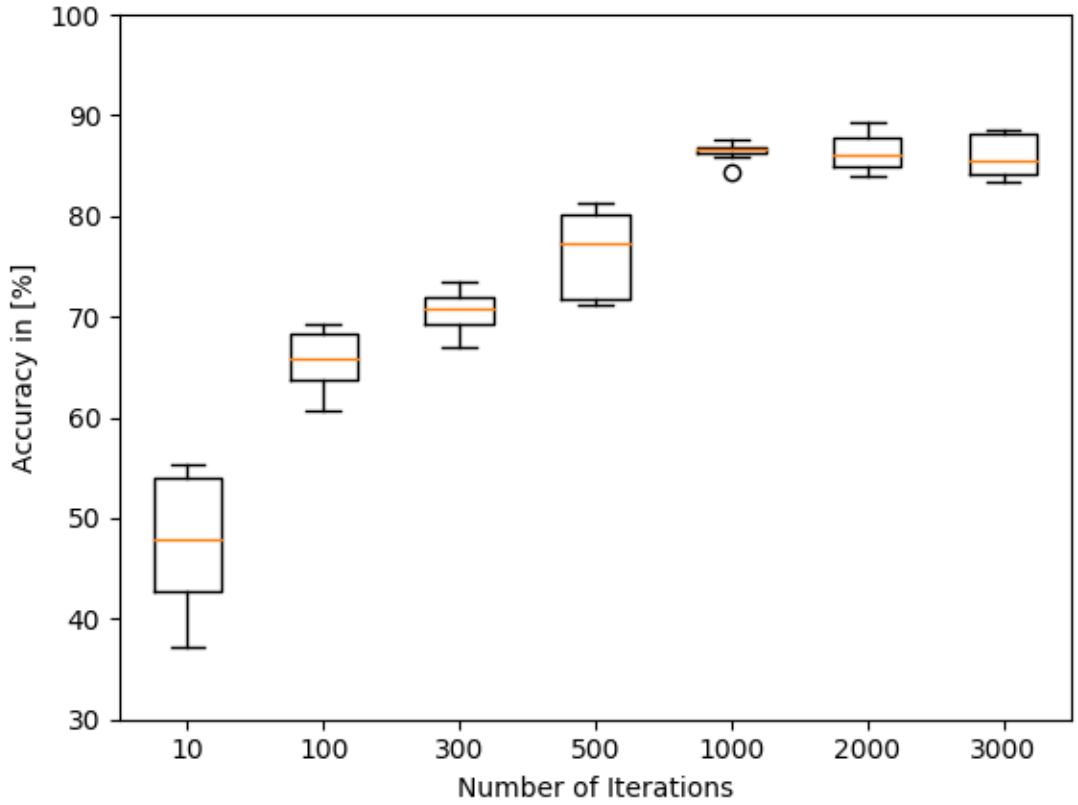


Figure 16. Training accuracy versus the number of iterations for ANFIS with a 10% training ratio.

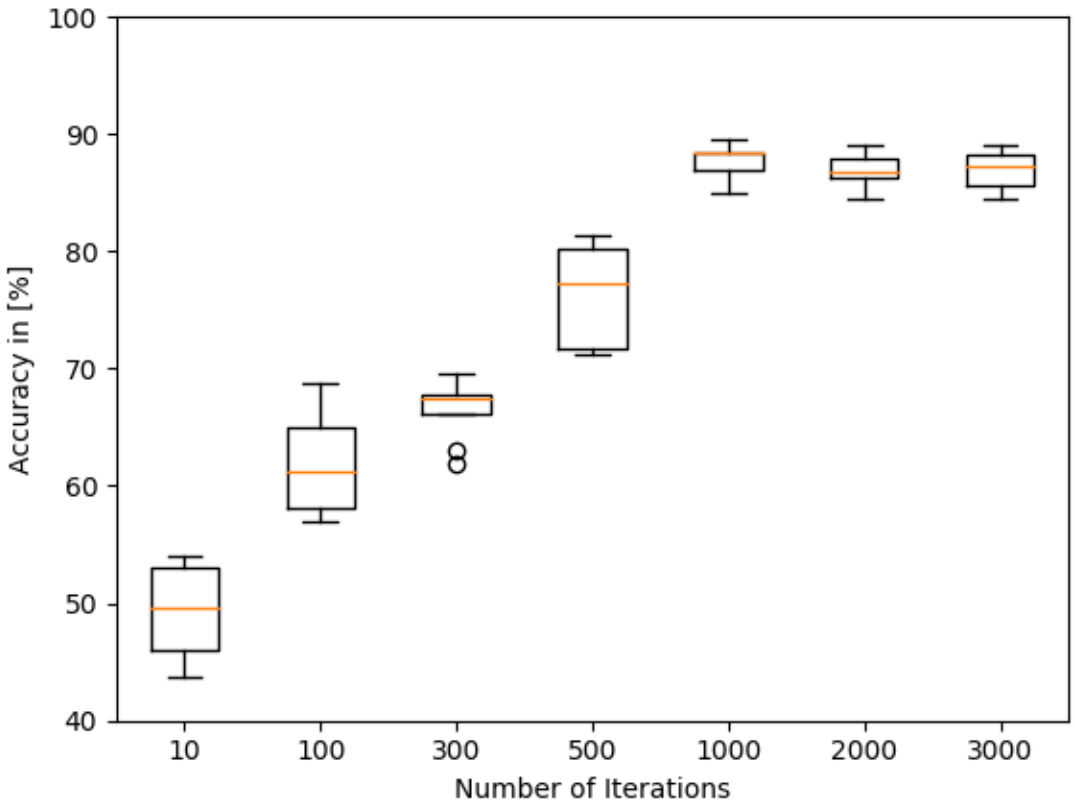


Figure 17. Testing accuracy versus the number of iterations for ANFIS with a 10% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.00	45.6	46.2
100	1.00	65.2	56.5
300	1.00	66.8	61.2
500	1.56	77.1	76.3
1000	2.68	88.3	86.5
2000	3.56	86.9	86.4
3000	5.34	87.0	86.0

Table 6. Average training and testing accuracies for ANFIS with a 10% training ratio.

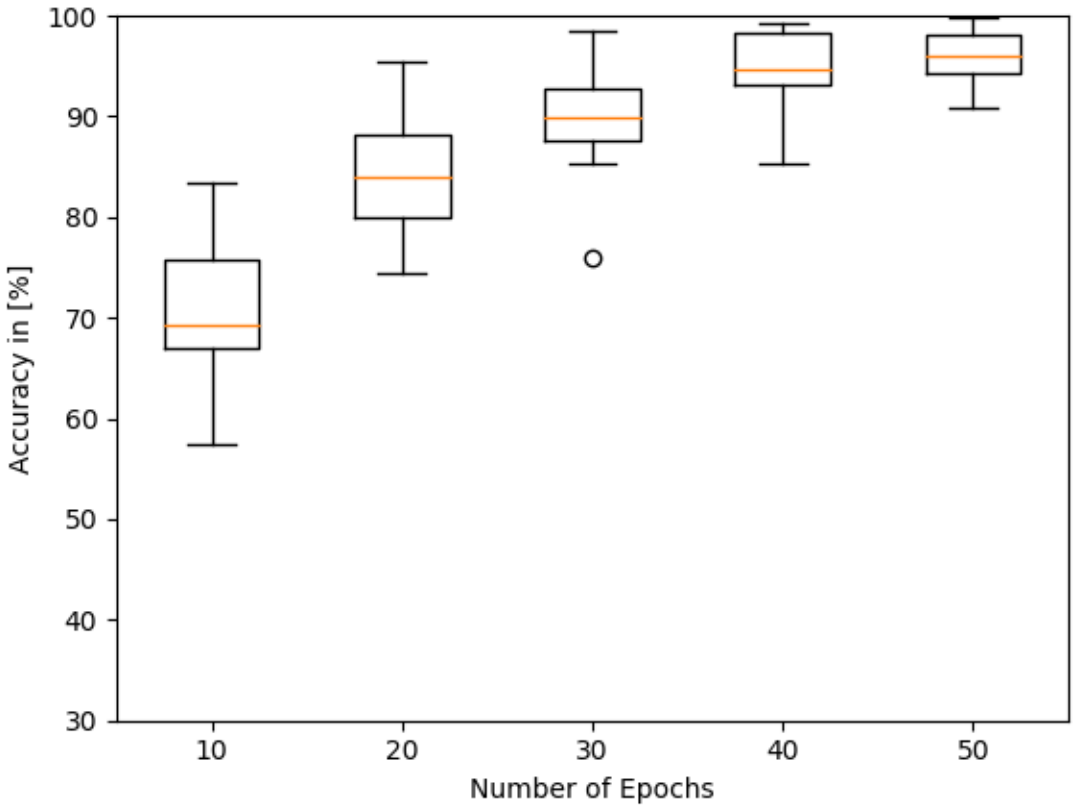


Figure 18. Training accuracy versus the number of epochs for LSTM with a 10% training ratio.

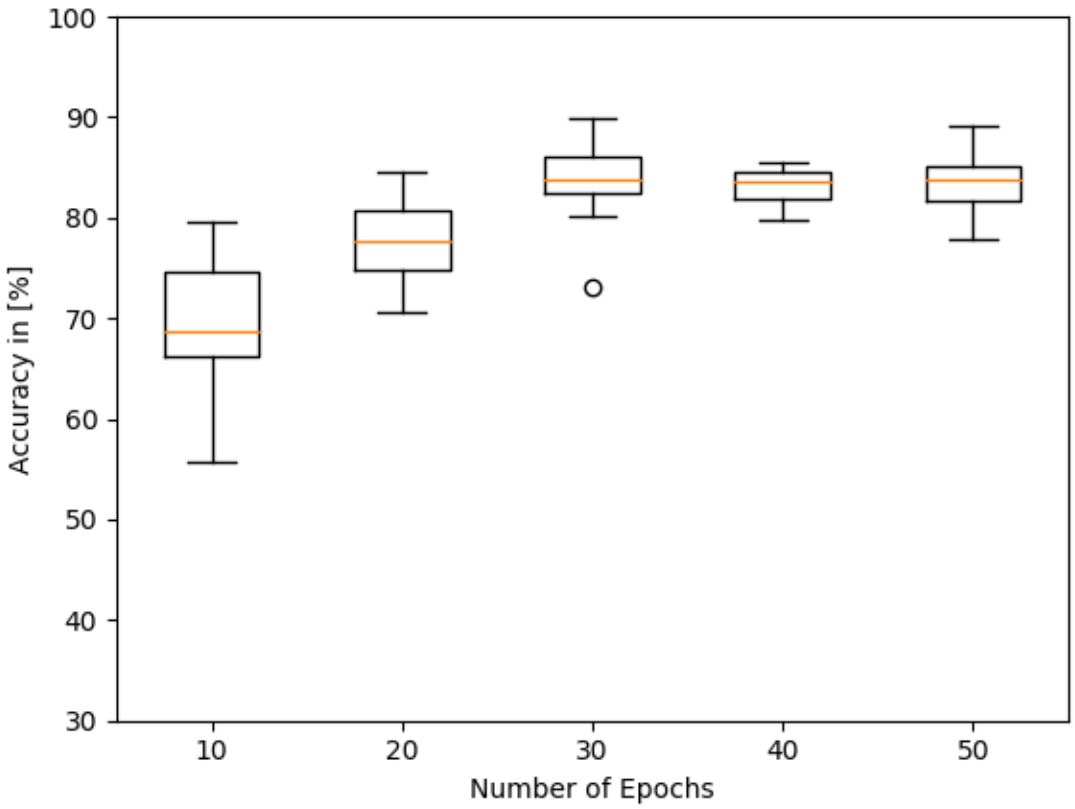


Figure 19. Testing accuracy versus the number of epochs for LSTM with a 10% training ratio.

Number of Epochs	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	63.0	70.7	69.3
20	79.2	83.9	77.7
30	120	90.0	83.8
40	159	94.7	83.2
50	205	96.1	83.5

Table 7. Average training and testing accuracies for LSTM with a 10% training ratio.

Number of Iterations	Running Time (min)	Training Accuracy (%)	Testing Accuracy (%)
3	18.45	84.16	85.9
6	27.5	87.3	86.5
9	34.5	87.4	86.7
11	28.2	87.3	86.5
12	41.2	87.3	86.5
15	51.4	87.2	86.3
22	36.2	86.4	86.5
30	59.5	87.3	87.6
33	43.1	87.3	86.9
37	47.2	87.3	87.3
44	51.3	87.5	86.9
45	82.3	87.6	87.3
55	73.5	89.2	89.8
56	60.2	90.9	89.1
66	64.4	88.1	88.9
74	86.3	90.9	90.9
88	78.4	90.1	89.6
110	89.3	92.3	90.1
111	101	93.1	93.1
112	112	93.4	92.2
148	135	93.7	93.7
168	158	93.6	93.1
185	177	94.3	94.3
224	207	93.8	93.5
280	281	94.2	93.9

Table 8. GoogLeNet training and testing accuracies with a 10% training ratio.

3.2.2 25% Training Ratio

The results obtained for MLP, ANFIS, LSTM, and GoogLeNet with a 25% training ratio are given in Figures 20 to 25. Tables 9 to 11 present the average accuracies and running times with different numbers of iterations for MLP, ANFIS, and LSTM. For LSTM, an epoch is 37 iterations. As in the previous subsection, the GoogLeNet results are for a single run per number of iterations and are given in Table 12.

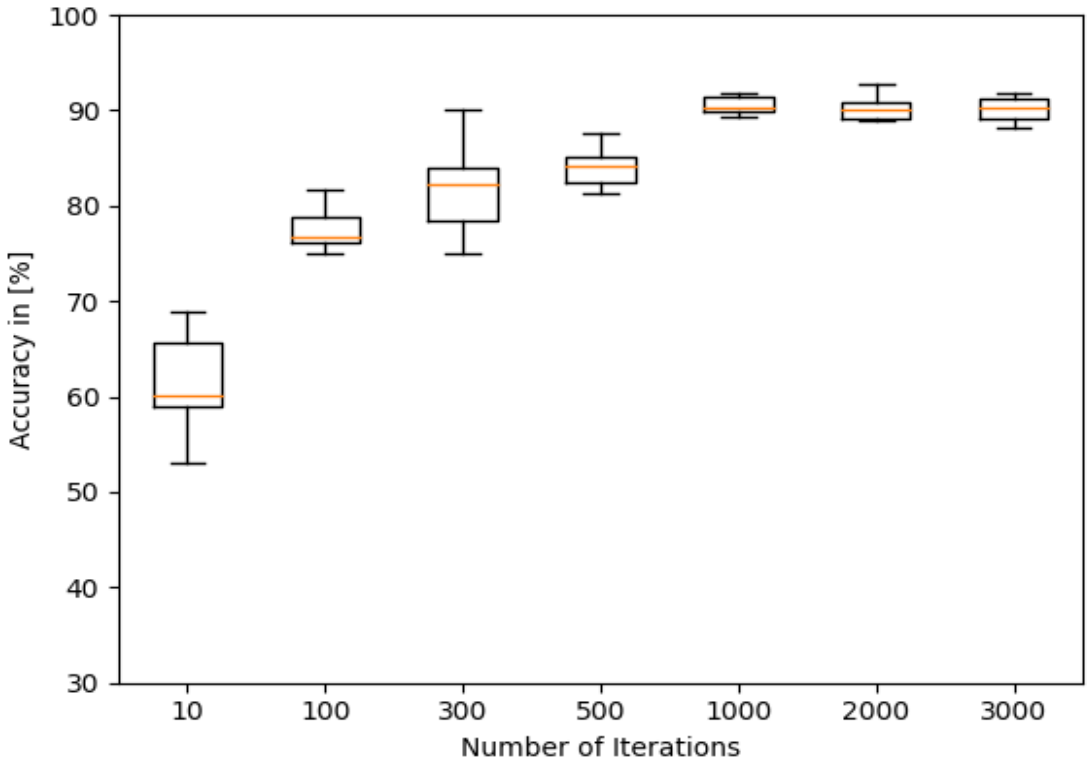


Figure 20. Training accuracy versus the number of iterations for MLP with a 25% training ratio.

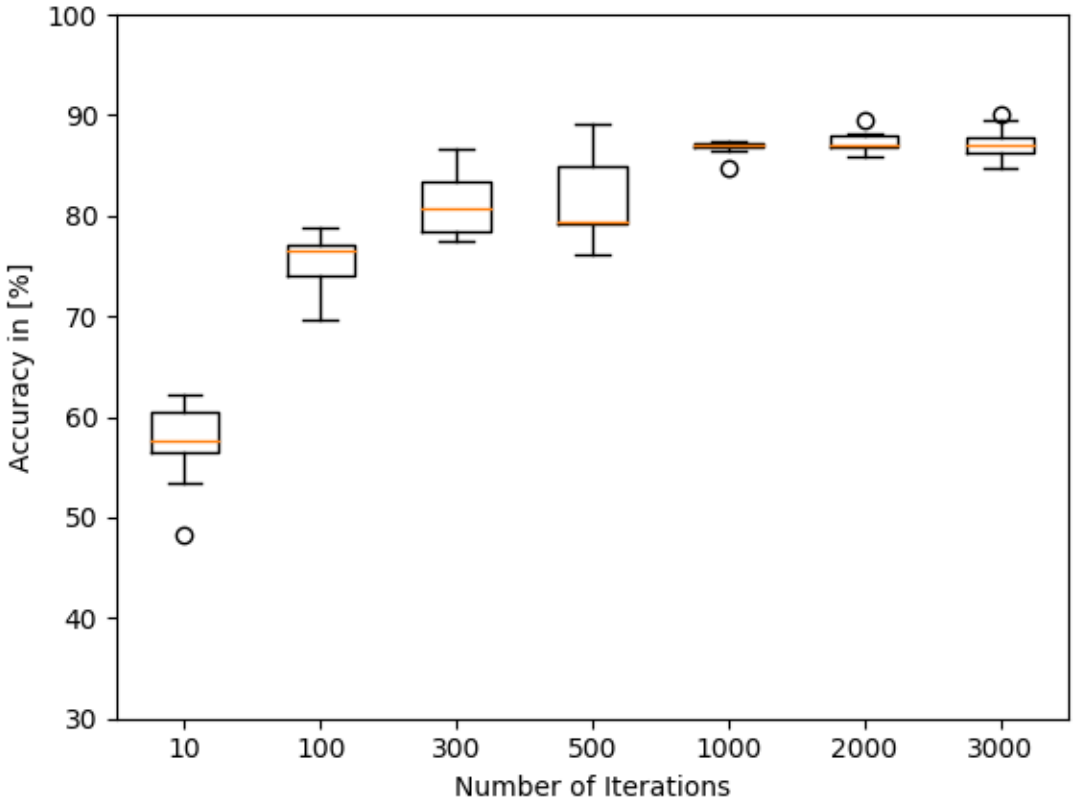


Figure 21. Testing accuracy versus the number of iterations for MLP with a 25% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.0	59.3	65.2
100	2.12	78.9	73.5
300	5.10	82.8	80.2
500	7.45	83.9	81.4
1000	10.7	90.9	86.8
2000	13.3	90.3	87.4
3000	20.6	90.2	87.2

Table 9. Average training and testing accuracies for MLP with a 25% training ratio.

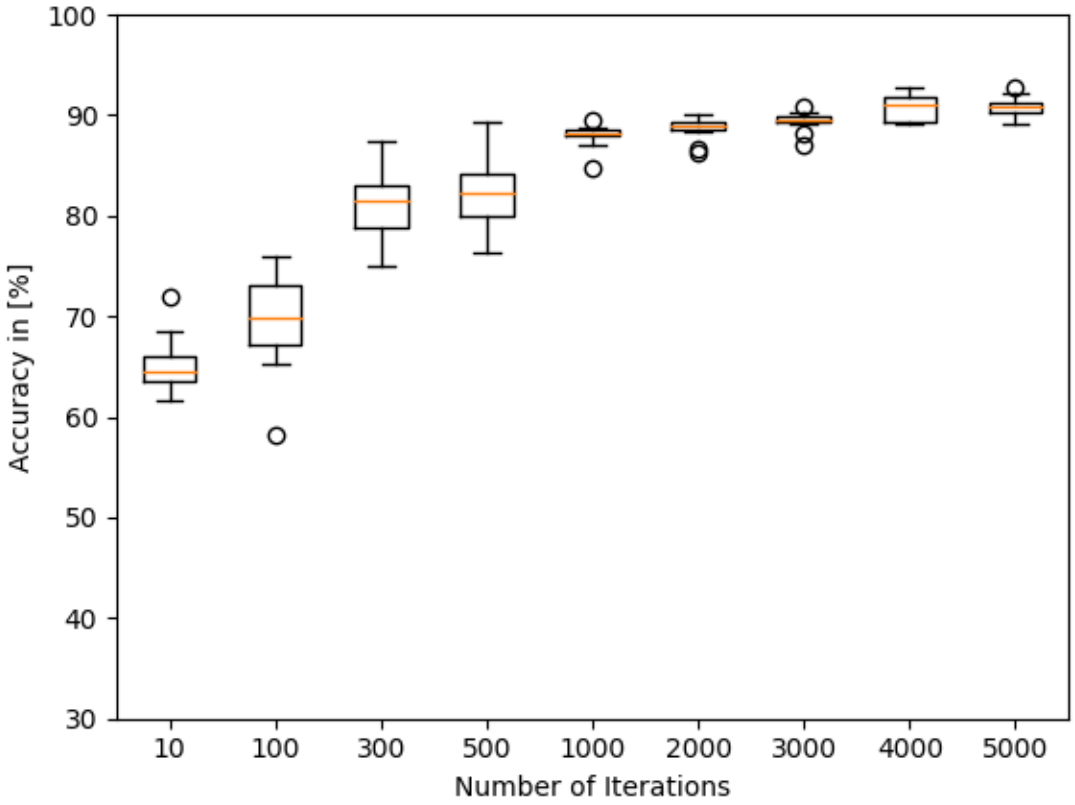


Figure 22. Training accuracy versus the number of iterations for ANFIS with a 25% training ratio.

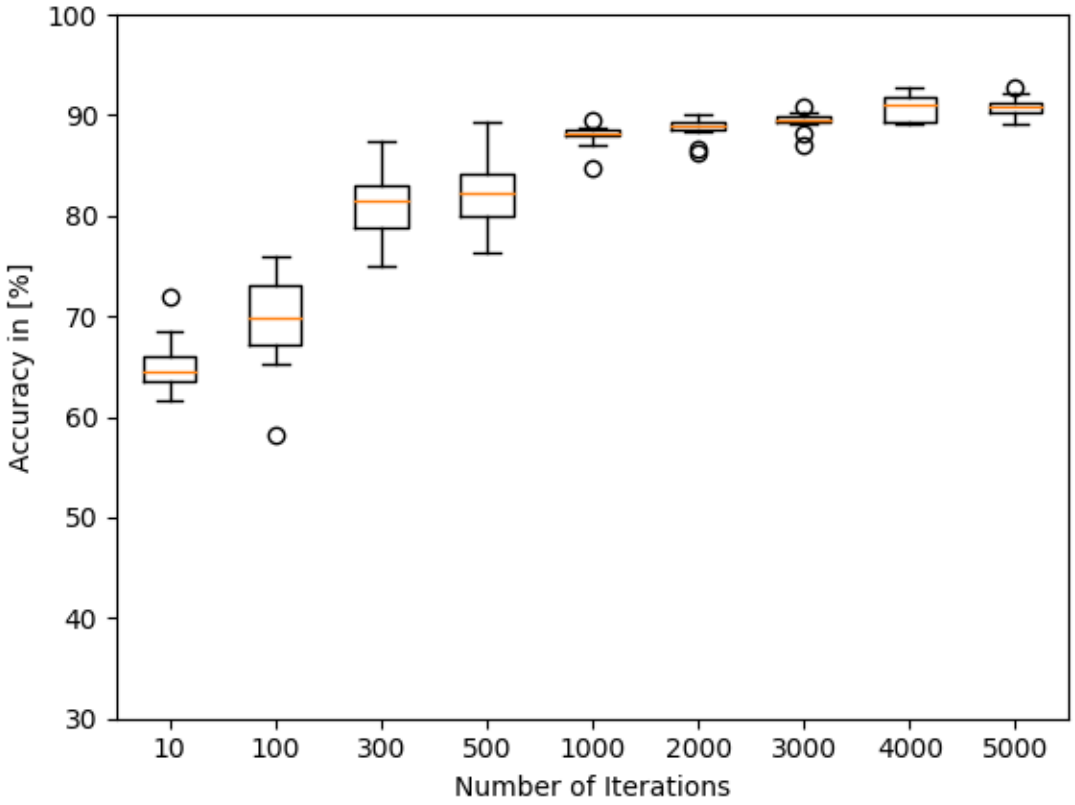


Figure 23. Testing accuracy versus the number of iterations for ANFIS with a 25% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.00	52.1	51.2
100	1.00	70.3	63.5
300	1.23	77.2	72.3
500	1.76	78.1	77.7
1000	2.81	87.2	86.7
2000	4.01	87.7	87.0
3000	6.50	87.8	87.3

Table 10. Average training and testing accuracies for ANFIS with a 25% training ratio.

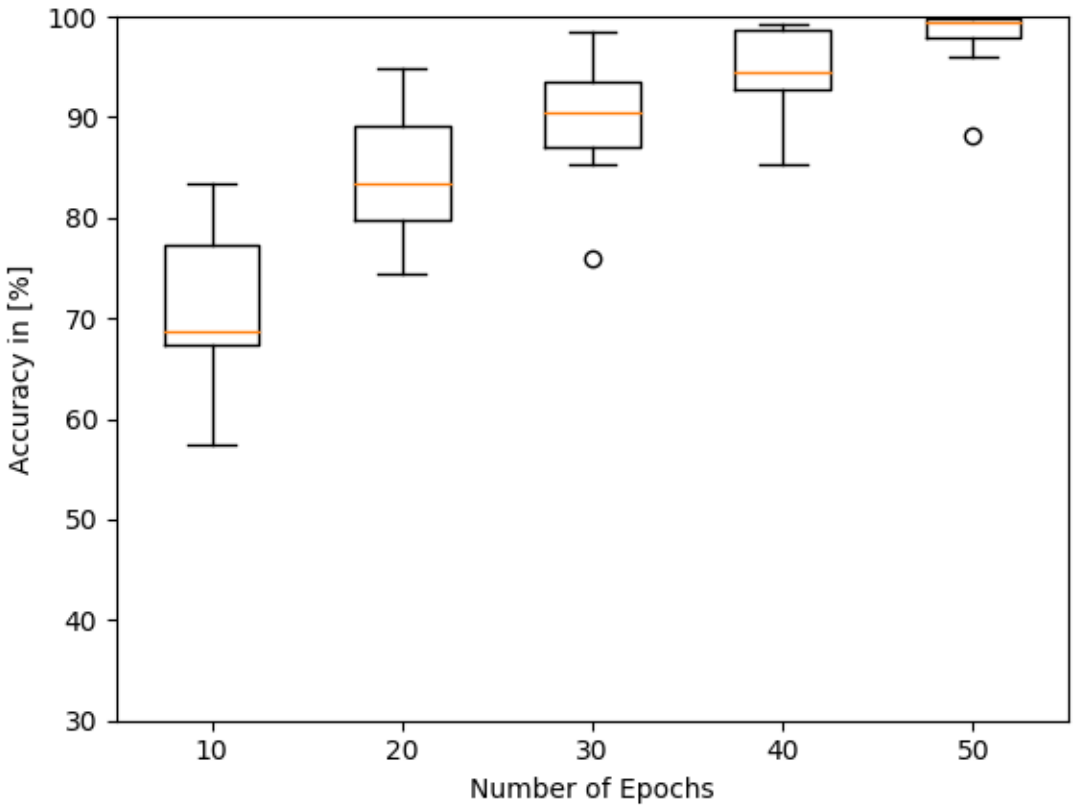


Figure 24. Training accuracy versus the number of epochs for LSTM with a 25% training ratio.

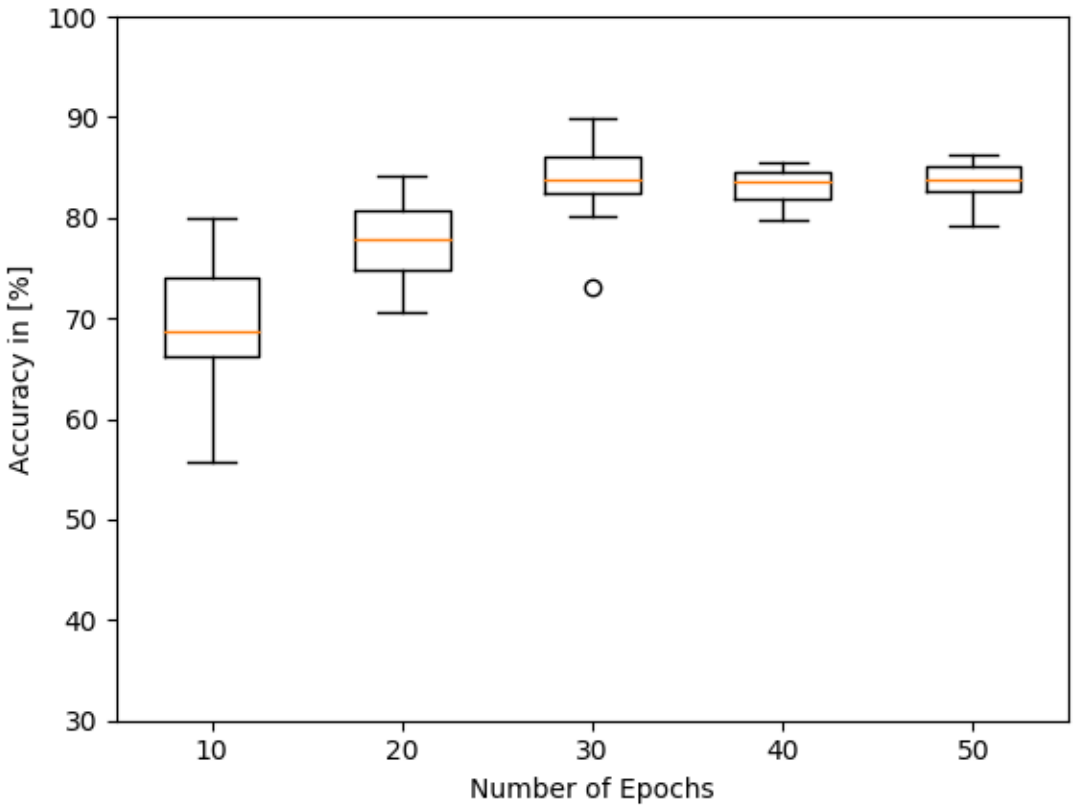


Figure 25. Testing accuracy versus the number of epochs for LSTM with a 25% training ratio.

Number of Epochs	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	55.9	70.8	69.2
20	77.2	84.0	77.8
30	120	90.0	83.7
40	161	94.7	83.2
50	228	97.9	83.7

Table 11. Average training and testing accuracies for LSTM with a 25% training ratio.

Number of Iterations	Running Time (min)	Training Accuracy (%)	Testing Accuracy (%)
7	22.5	86.9	86.7
94	91.5	93.5	93.5
141	154	94.5	95.3
282	257	95.3	94.5

Table 12. GoogLeNet training and testing accuracies with a 25% training ratio.

3.2.3 50% Training Ratio

The results obtained with MLP, ANFIS, LSTM, and GoogLeNet for a 50% training ratio are presented in Figures 26 to 31. The average accuracies and running times with different numbers of iterations for MLP, ANFIS, and LSTM are given in Tables 13 to 15. For LSTM, an epoch is 71 iterations. Since GoogLeNet has a very high running time, results for only a single run are given in Table 16.

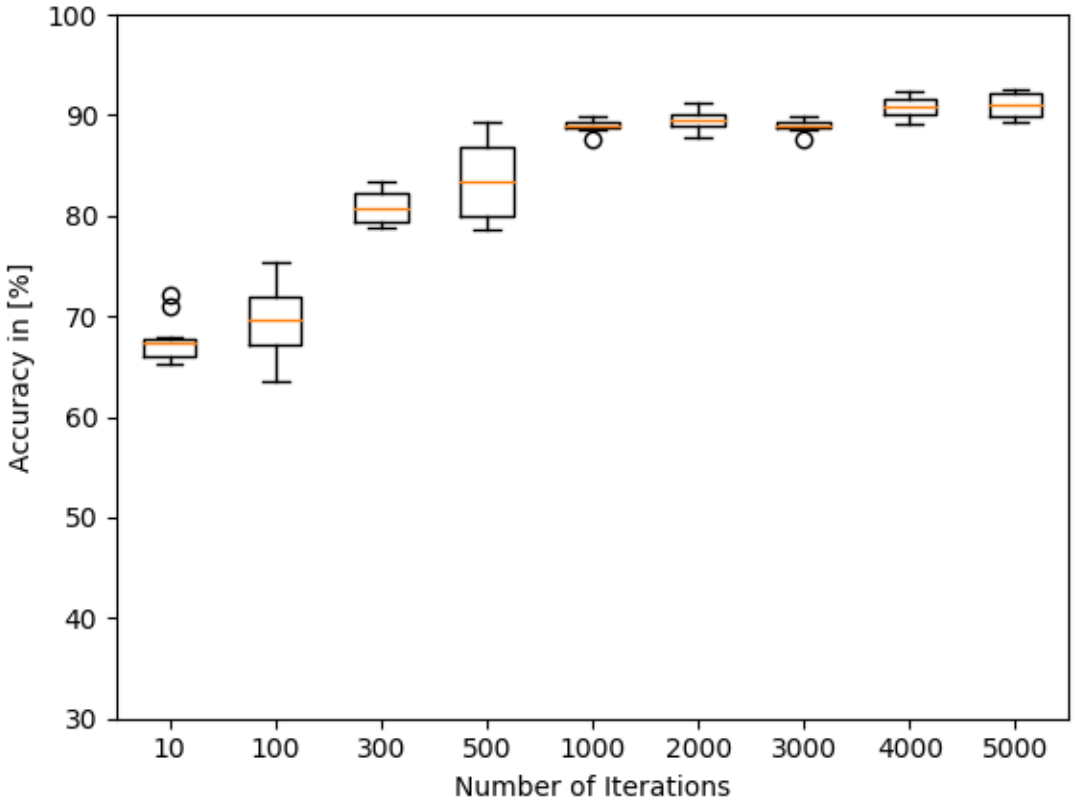


Figure 26. Training accuracy versus the number of iterations for MLP with a 50% training ratio.

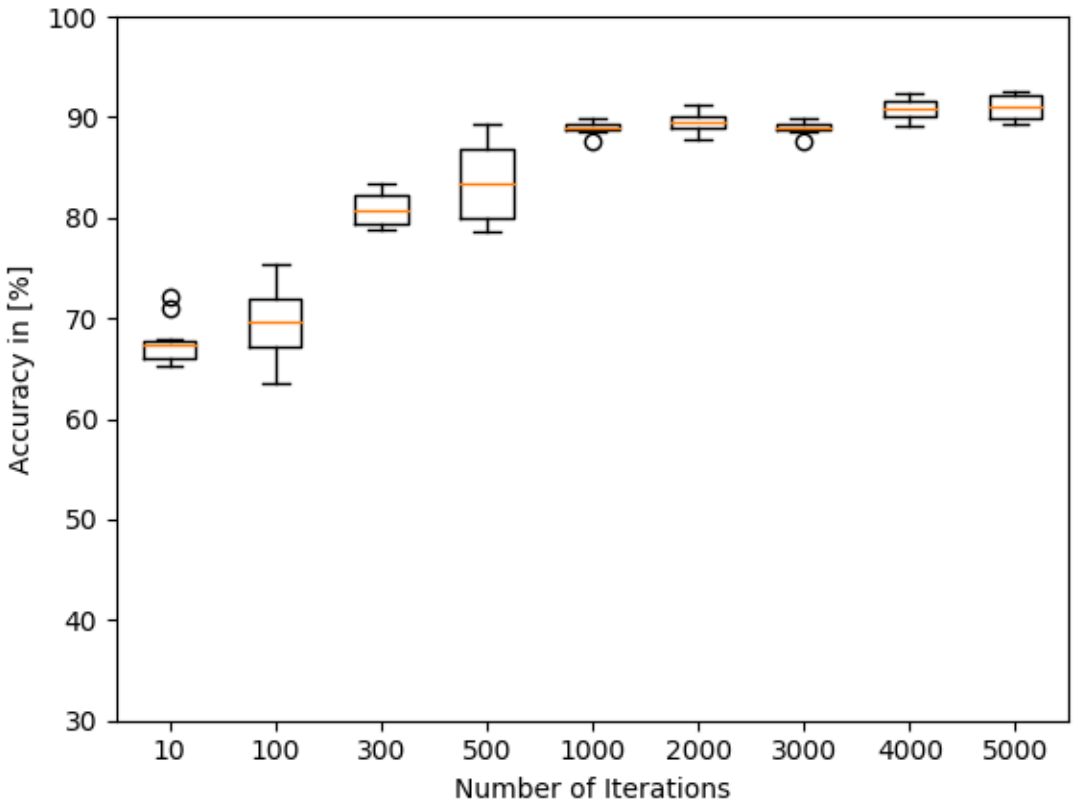


Figure 27. Testing accuracy versus the number of iterations for MLP with a 50% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.00	67.1	65.2
100	1.12	69.6	64.9
300	5.20	81.3	79.2
500	8.10	83.3	82.4
1000	10.2	89.0	87.9
2000	15.4	89.5	88.7
3000	23.3	89.1	89.4
4000	28.5	91.9	90.8
5000	91.6	91.6	37.3

Table 13. Average training and testing accuracies for MLP with a 50% training ratio.

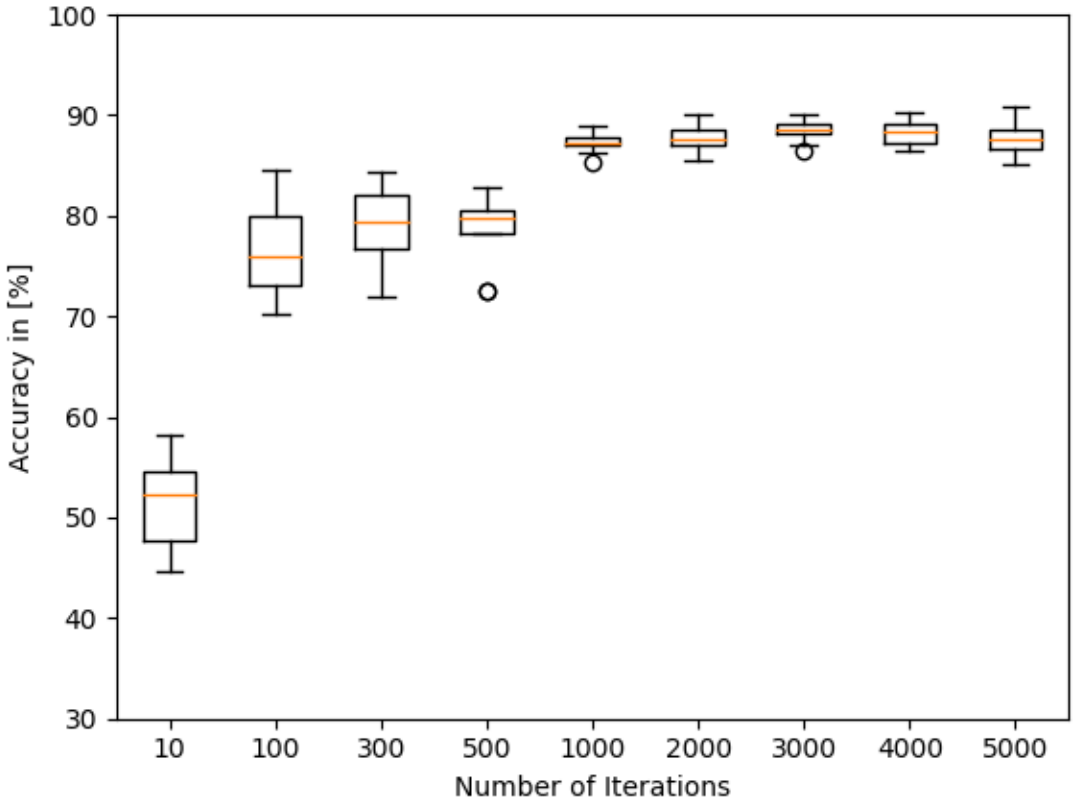


Figure 28. Training accuracy versus the number of iterations for ANFIS with a 50% training ratio.

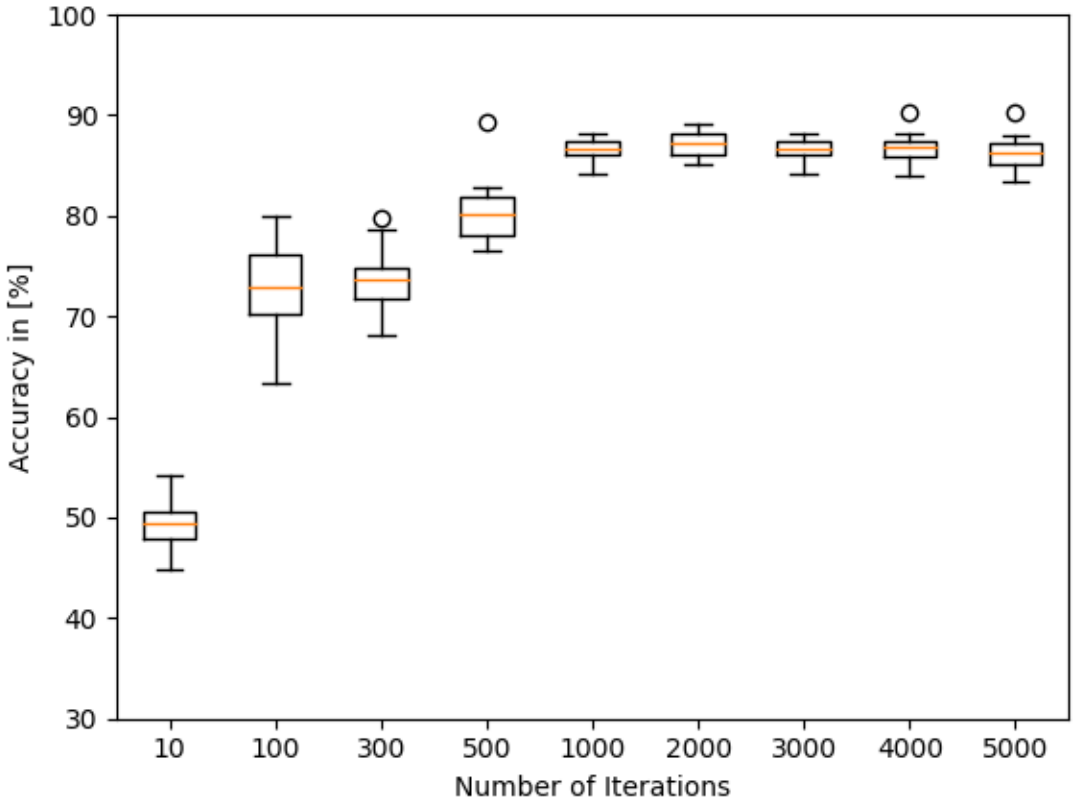


Figure 29. Testing accuracy versus the number of iterations for ANFIS with a 50% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.00	50.5	49.4
100	1.00	75.2	72.9
300	1.00	77.2	73.6
500	2.08	78.6	80.5
1000	2.85	87.3	86.9
2000	4.57	87.7	87.2
3000	6.50	88.5	86.6
4000	9.30	87.7	86.8
5000	14.2	88.5	86.3

Table 14. Average training and testing accuracies for ANFIS with a 50% training ratio.

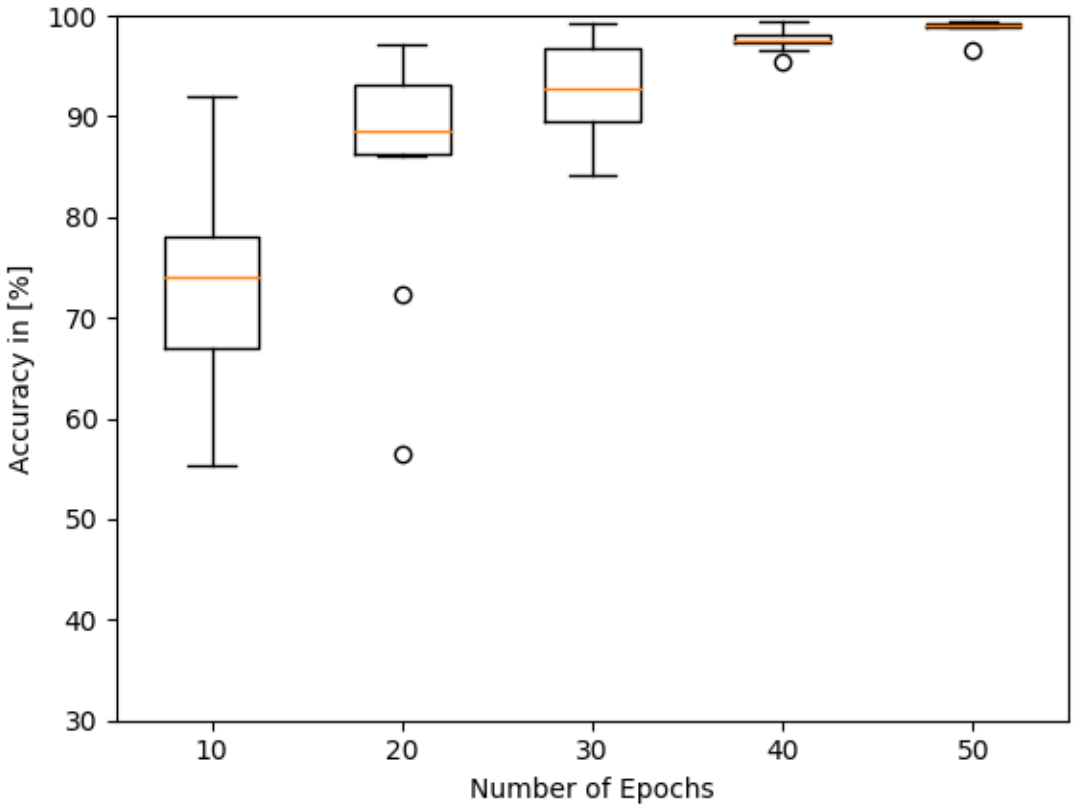


Figure 30. Training accuracy versus the number of epochs for LSTM with a 50% training ratio.

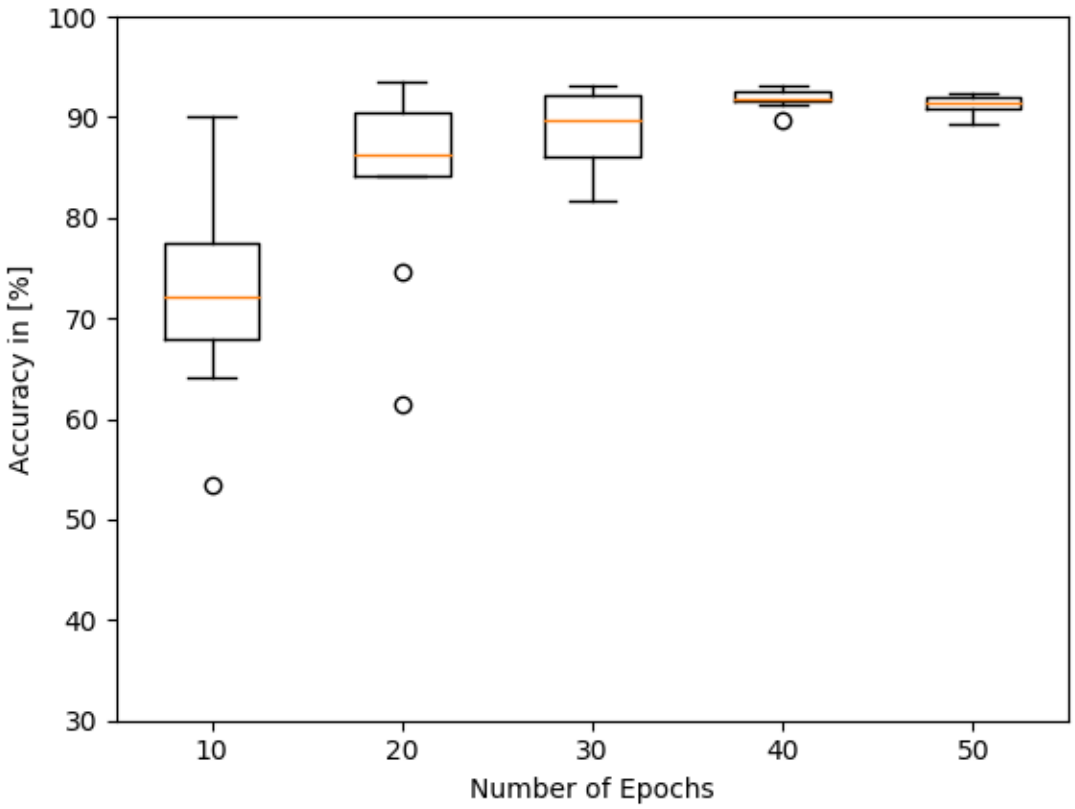


Figure 31. Testing accuracy versus the number of epochs for LSTM with a 50% training ratio.

Number of Epochs	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	101.2	74.1	53.5
20	197.5	86.1	84.3
30	291.4	92.9	92.4
40	421.8	97.6	93.2
50	598.5	98.9	90.9

Table 15. Average training and testing accuracies for LSTM with a 50% training ratio.

Number of Iterations	Running Time (min)	Training Accuracy (%)	Testing Accuracy (%)
7	35.57	87.3	86.7
188	134.3	93.9	94.0
282	159.3	94.2	95.2

Table 16. GoogLeNet training and testing accuracies for a 50% training ratio.

3.2.3 75% Training Ratio

The results obtained with MLP, ANFIS, LSTM, and GoogLeNet and a 75% training ratio are presented in Figures 32 to 37. The average accuracies and running times with different numbers of iterations for MLP, ANFIS, and LSTM are given in Tables 17 to 19. For LSTM, an epoch is 106 iterations. For GoogLeNet, results for only a single run per number of iterations are presented in Table 20.

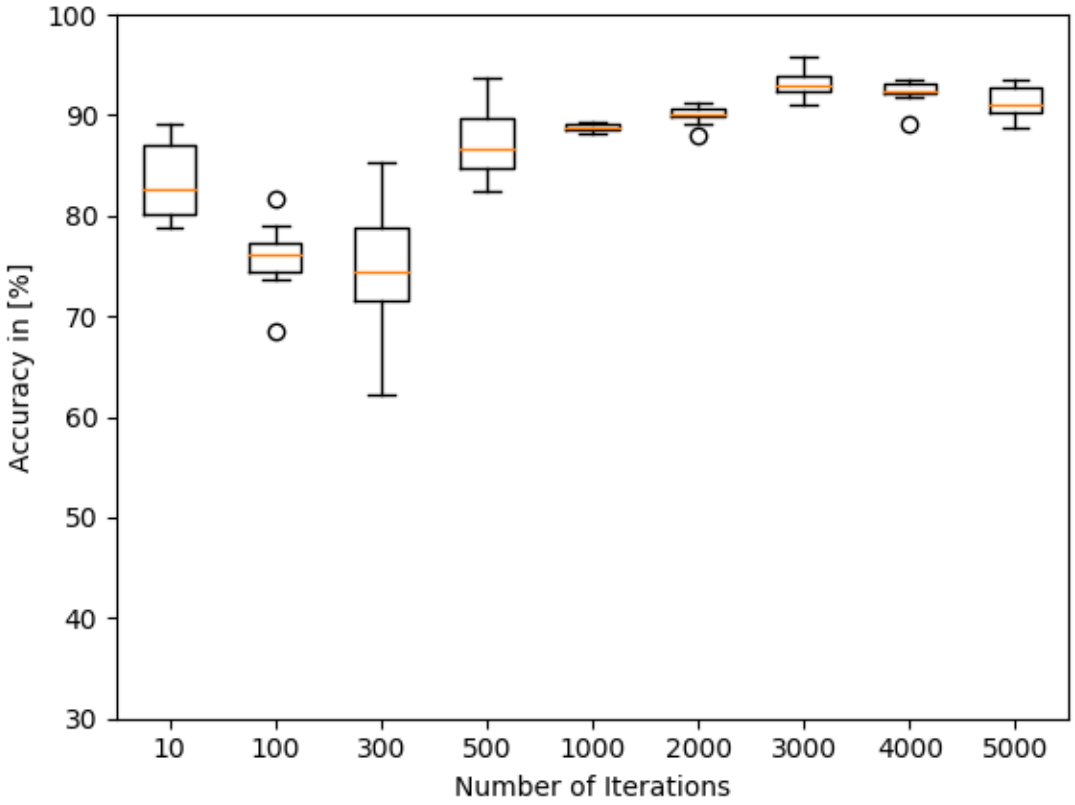


Figure 32. Training accuracy versus the number of iterations for MLP with a 75% training ratio.

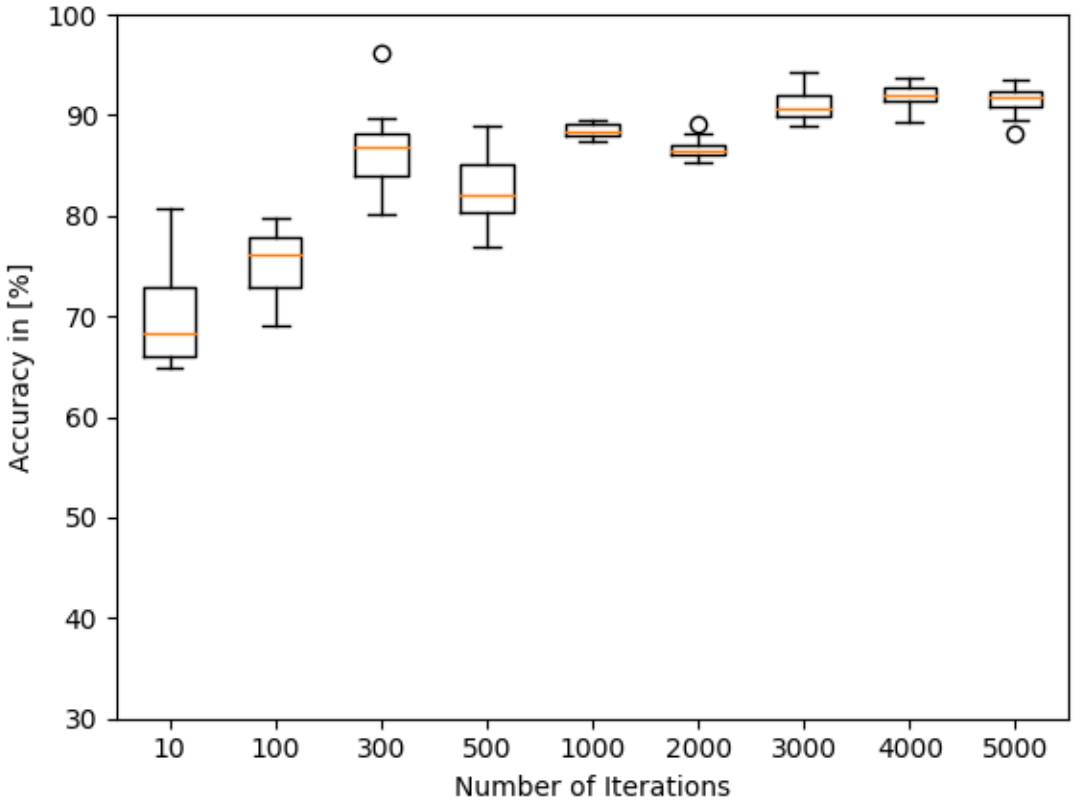


Figure 33. Testing accuracy versus the number of iterations for MLP with a 75% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.00	75.2	65.2
100	2.10	77.9	69.6
300	4.30	88.1	81.0
500	7.70	83.3	82.4
1000	9.60	89.0	87.9
2000	18.6	89.5	88.7
3000	34.9	89.0	89.4
4000	37.9	90.9	90.8
5000	53.2	91.0	90.9

Table 17. Average training and testing accuracies for MLP with a 75% training ratio.

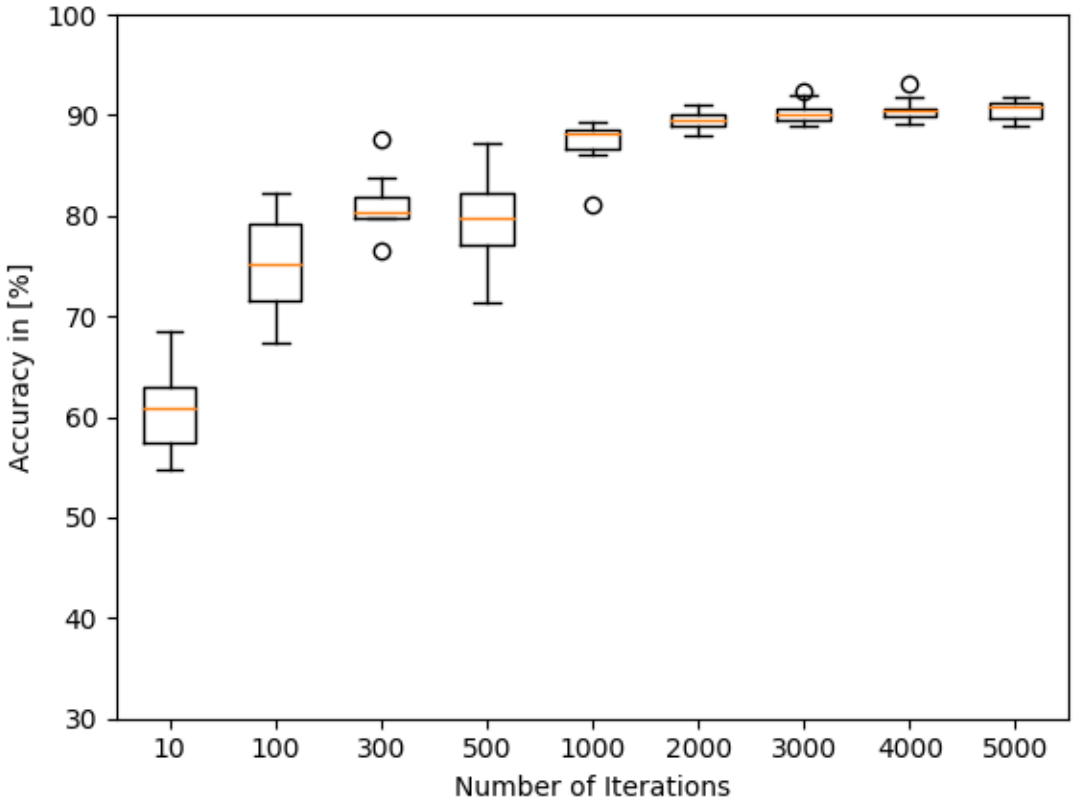


Figure 34. Training accuracy versus the number of iterations for ANFIS with a 75% training ratio.

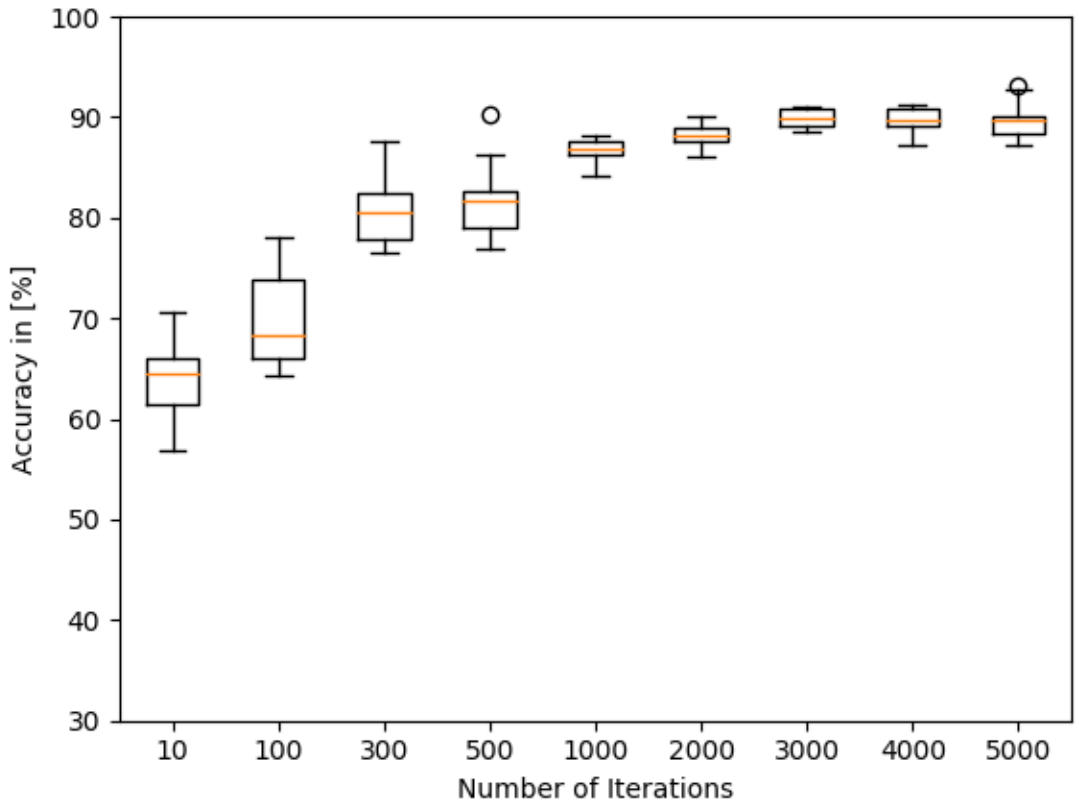


Figure 35. Testing accuracy versus the number of iterations for ANFIS with a 75% training ratio.

Number of Iterations	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	1.00	61.2	63.6
100	1.00	73.6	69.9
300	1.00	79.1	80.6
500	1.90	79.8	81.7
1000	3.50	87.3	86.8
2000	5.30	89.6	88.3
3000	7.70	90.3	90.0
4000	9.90	90.6	89.7
5000	12.5	90.5	89.7

Table 18. Average training and testing accuracies for ANFIS with a 75% training ratio.

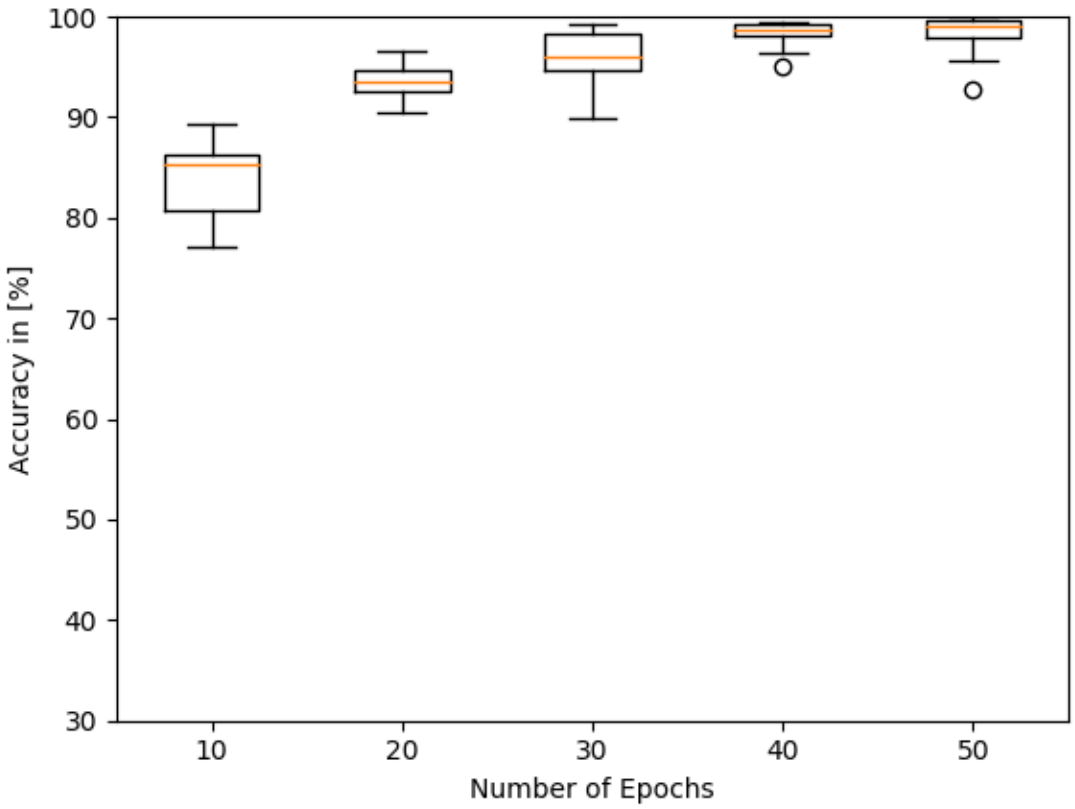


Figure 36. Training accuracy versus the number of epochs for LSTM with a 75% training ratio.

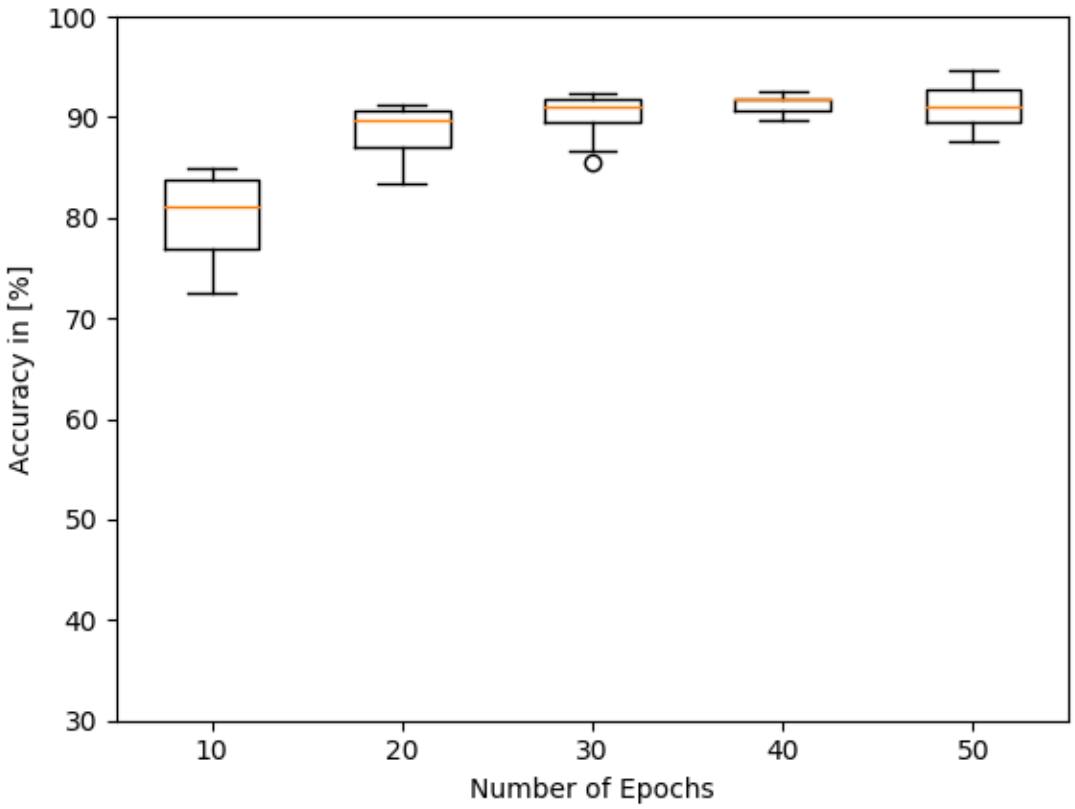


Figure 37. Testing accuracy versus the number of epochs for LSTM with a 75% training ratio.

Number of Epochs	Average Running Time (s)	Average Training Accuracy (%)	Average Testing Accuracy (%)
10	121	83.7	80.1
20	186	93.6	88.8
30	283	96.1	90.1
40	445	98.3	91.3
50	485	98.2	91.1

Table 19. Average training and testing accuracies for LSTM with a 75% training ratio.

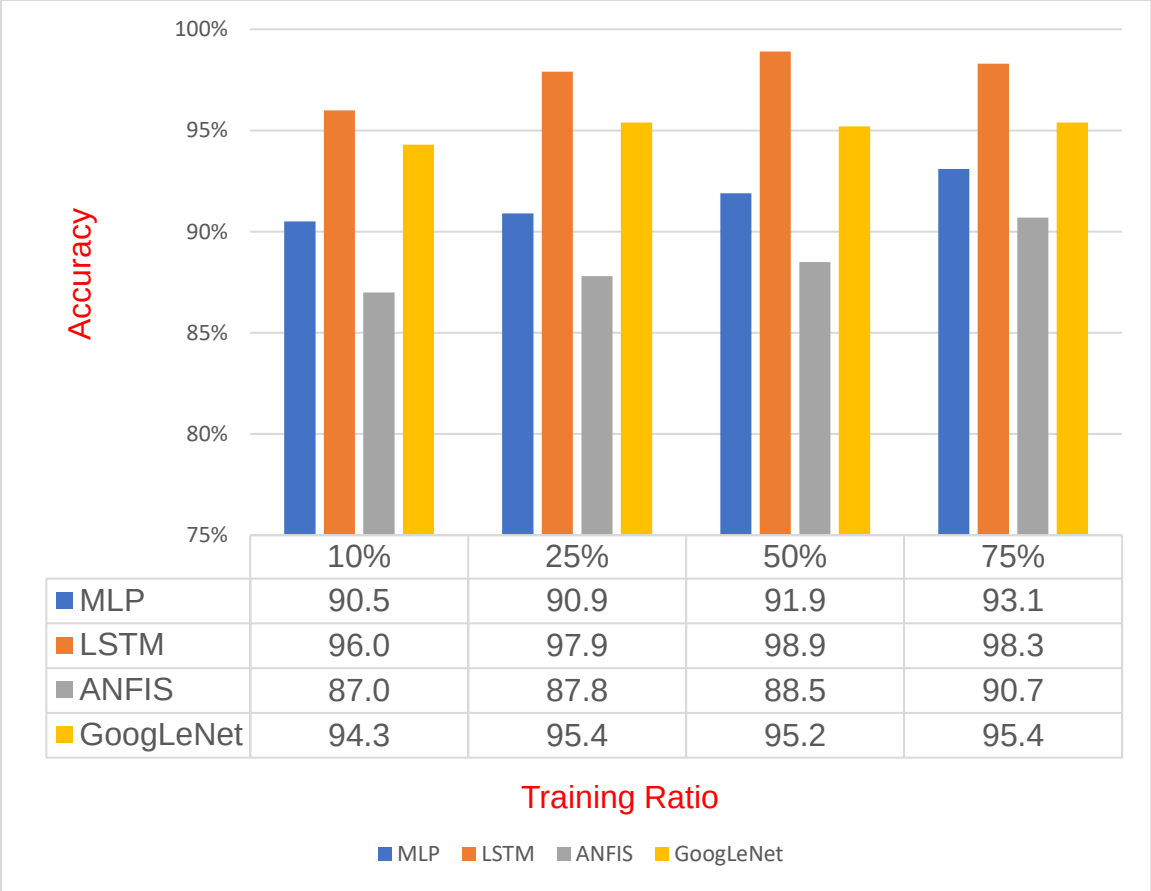
Number of Iterations	Running Time (min)	Training Accuracy (%)	Testing Accuracy (%)
7	35.5	87.3	86.7
282	117	95.1	95.3
424	152	94.8	95.4
1505	500	96.9	97.1

Table 20. GoogLeNet training and testing accuracies for a 75% training ratio.

Figures 14 to 37 show that with 10%, 25%, 50%, and 75% training ratios, the MLP, ANFIS, and LSTM training and testing accuracies increase with the number of iterations. Further, the standard deviations of the training and testing accuracies decrease with the number of iterations. For GoogLeNet, the results also show that the testing and training accuracies increase with the number of iterations. Table 20 shows that for GoogLeNet with a 75% training ratio, increasing the number of iterations from 424 to 1505 only improves the accuracy by around 2%.

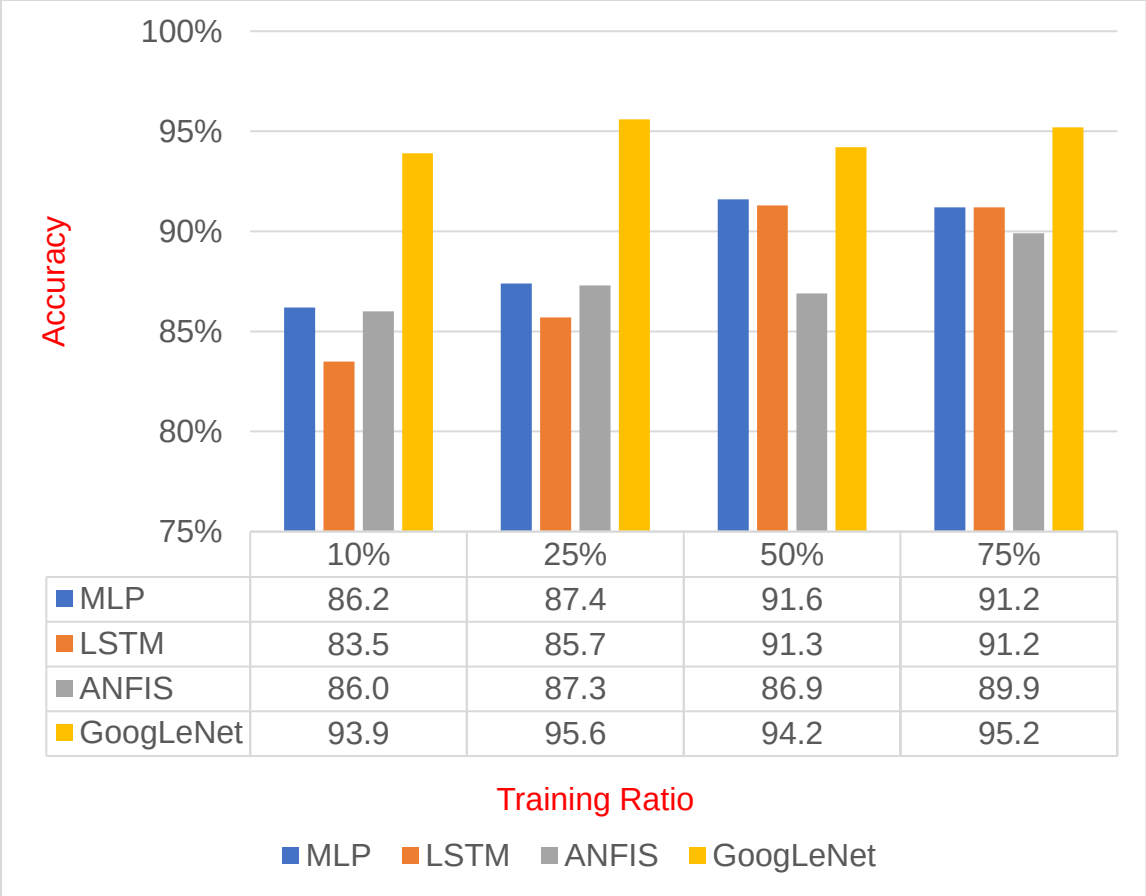
3.3 Discussion

Figures 38 and 39 give the training and testing accuracies of the four systems for the four training ratios. Figures 45 to 48 show the training accuracies of these systems on a logarithmic scale versus running time for the four training ratios. The large differences between the running times, especially GoogLeNet, is the reason for using a logarithmic scale.



	10%	25%	50%	75%
MLP	90.5	90.9	91.9	93.1
LSTM	96.0	97.9	98.9	98.3
ANFIS	87.0	87.8	88.5	90.7
GoogLeNet	94.3	95.4	95.2	95.4

Figure 38. Average training accuracy of the four systems with four different training ratios.



	10%	25%	50%	75%
MLP	86.2	87.4	91.6	91.2
LSTM	83.5	85.7	91.3	91.2
ANFIS	86.0	87.3	86.9	89.9
GoogLeNet	93.9	95.6	94.2	95.2

Figure 39. Average testing accuracy of the four systems with four different training ratios.

Figure 38 presents the training accuracy of the four systems with four different training ratios. This shows that for all ratios, the highest training accuracy in order belongs to LSTM, GoogLeNet, and MLP. Further, increasing the training ratio from 10% to 75% improves the training accuracies of MLP, ANFIS, and LSTM by about 3 to 4%. However, the GoogLeNet training accuracy only increases by 1%.

Figure 39 presents the testing accuracies of the systems for different training ratios. These results show that the highest testing accuracies for all training ratios belong to GoogLeNet and MLP, respectively. Further, increasing the training ratio from 50% to 75%,

resulted in similar testing accuracies with MLP and LSTM, while for 10% and 25% training ratios the MLP testing accuracy is higher than LSTM. On the other hand, increasing the training ratio from 10% to 75% improved the LSTM and MLP testing accuracies from 83% to 91%, and 86% to 91%, respectively. However, for ANFIS and GoogLeNet, the testing accuracies increased by only around 2 to 3%. In addition, for 10% and 25% training ratios, the testing accuracy of ANFIS is higher than LSTM while for 50% and 75% training ratios the LSTM testing accuracy is higher.

A comparison between Figures 38 and 39 shows that in general, the training accuracy of all systems is higher than their testing accuracy. Further, the training ratio has a greater effect on the training and testing accuracies of MLP and LSTM compared to ANFIS and GoogLeNet. Averaging the training and testing accuracies of the systems shows that the highest values belong to GoogLeNet, LSTM and, MLP, respectively. Interestingly, for a 10% training ratio, LSTM has the highest training accuracy while its testing accuracy is the lowest, which may be due to overfitting.

Figures 40 to 43 illustrate the training accuracy of the systems on a logarithmic scale for the running time with different training ratios. These results indicate that the main advantage of GoogLeNet is its high accuracy despite the long training time. In addition, GoogLeNet has high testing and training accuracies for low training ratios, showing its low dependency on the training ratio. On the other hand, ANFIS is very fast but has low accuracy, whereas MLP has a low running time and high accuracy. Figures 38 and 39 show that another advantage of MLP is that its accuracy is less affected by the training ratio than LSTM and ANFIS.

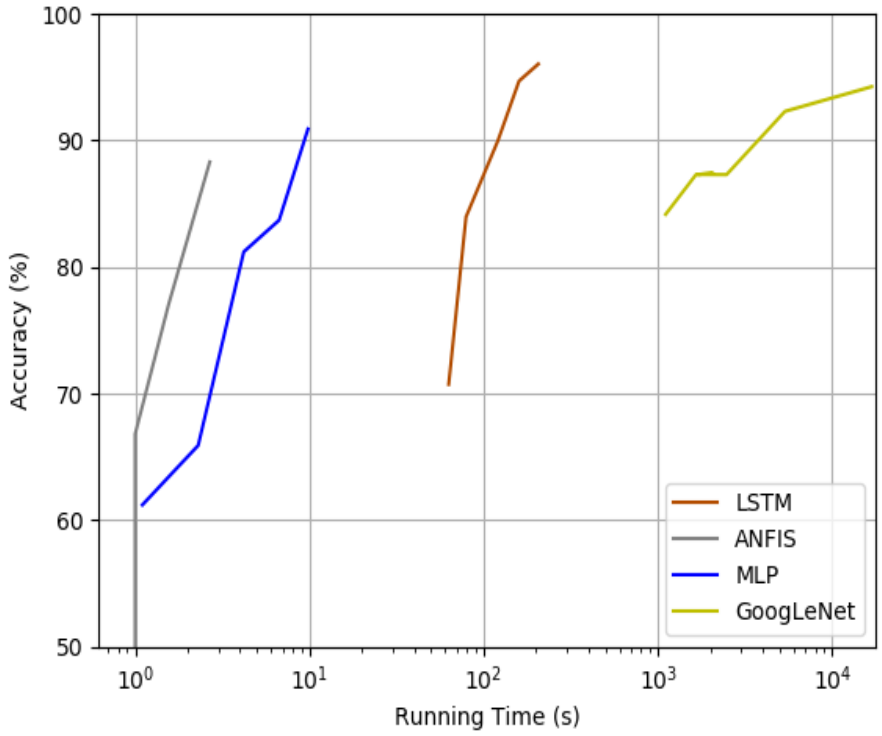


Figure 40. Running time versus average training accuracy for the four systems with a 10% training ratio.

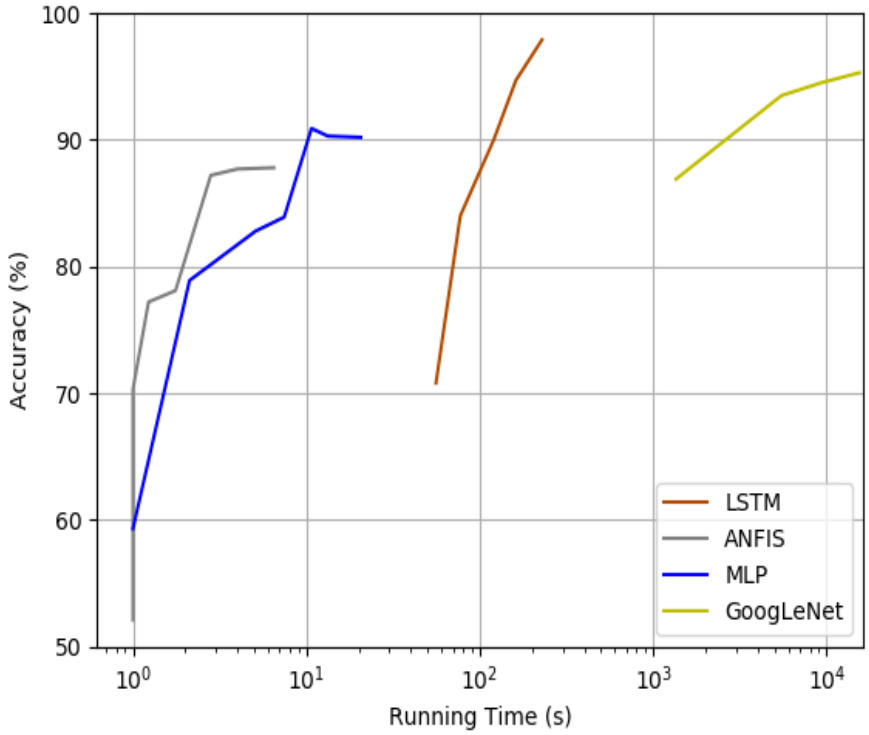


Figure 41. Running time versus average training accuracy for the four systems with a 25% training ratio.

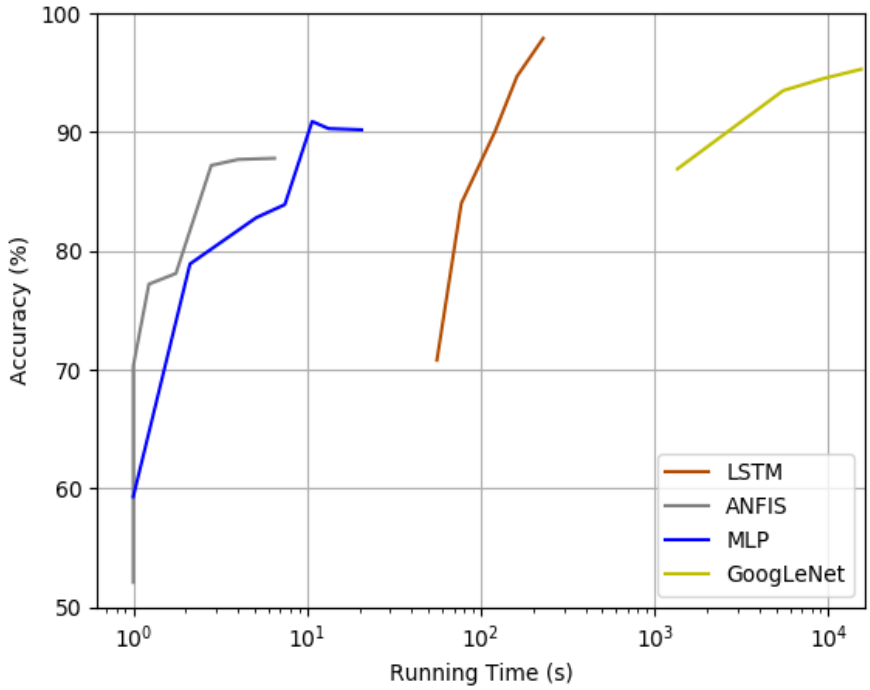


Figure 42. Running time versus average training accuracy for the four systems with a 50% training ratio.

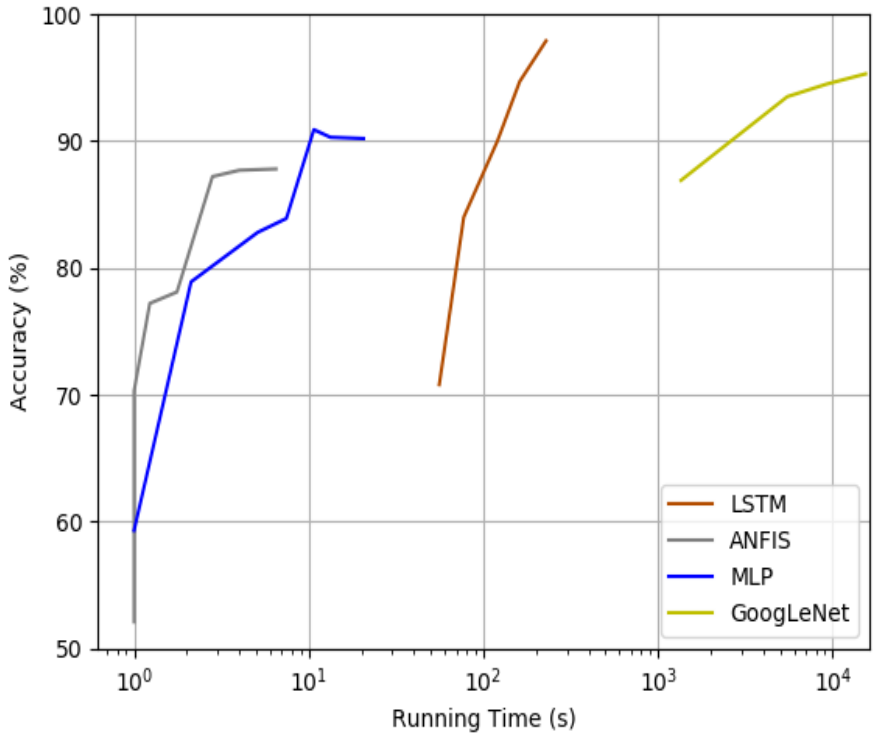


Figure 43. Running time versus average training accuracy for the four systems with a 75% training ratio.

Figures 40 to 43 show that the training times of ANFIS and MLP are lower than LSTM. Also, the GoogLeNet training time is much higher than that of the other systems for all training ratios. These results indicate that in contrast to MLP, LSTM, and GoogLeNet, the training accuracy of ANFIS never reaches 90%. Furthermore, MLP and LSTM reach 90% training accuracy after 10 and 100 seconds, respectively, while GoogLeNet takes around 3000 seconds to reach this training accuracy. The high running time for GoogLeNet is due to the high dimension of the input data. Since GoogLeNet works with 2D image data, it takes more time to process each image. Also, GoogLeNet extracts features while in the other three systems, feature extraction is done during preprocessing. Since feature extraction for the MLP, ANFIS, and LSTM systems takes very little time, the time for preprocessing is ignored.

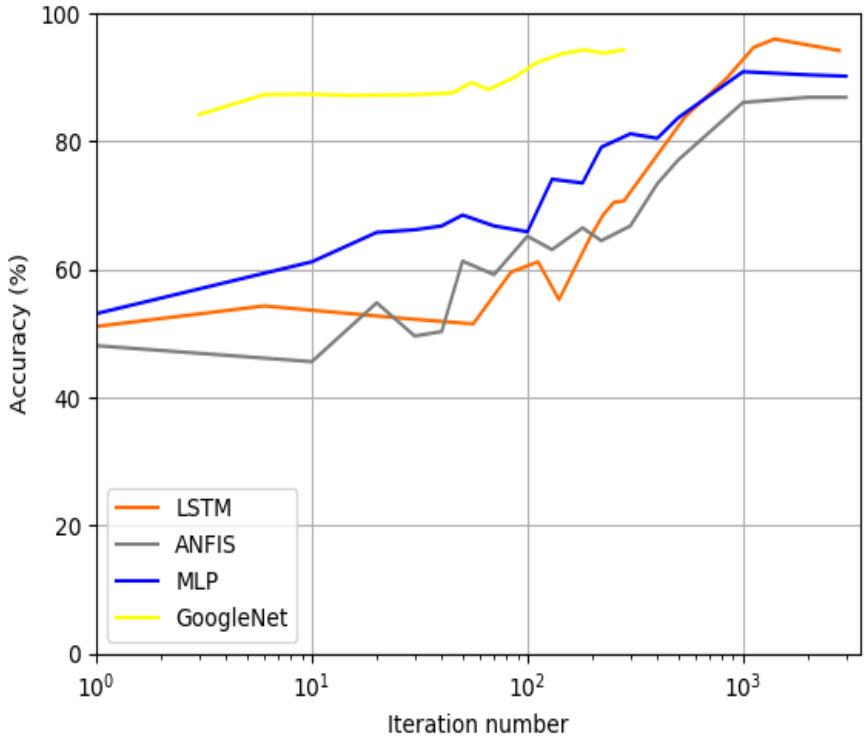


Figure 44. Number of iterations versus average training accuracy for the four systems with a 10% training ratio.

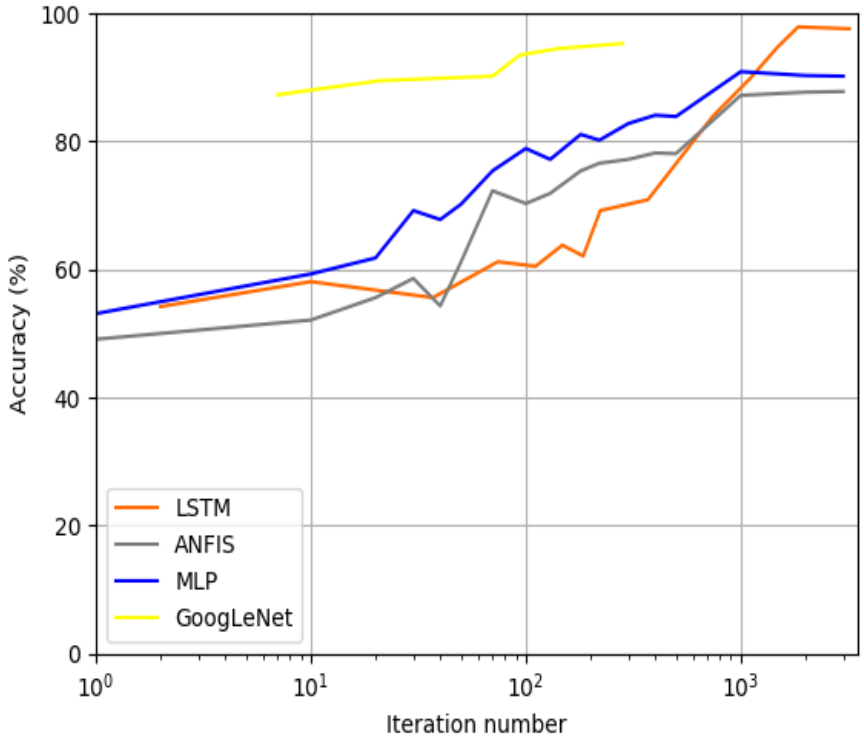


Figure 45. Number of iterations versus average training accuracy for the four systems with a 25% training ratio.

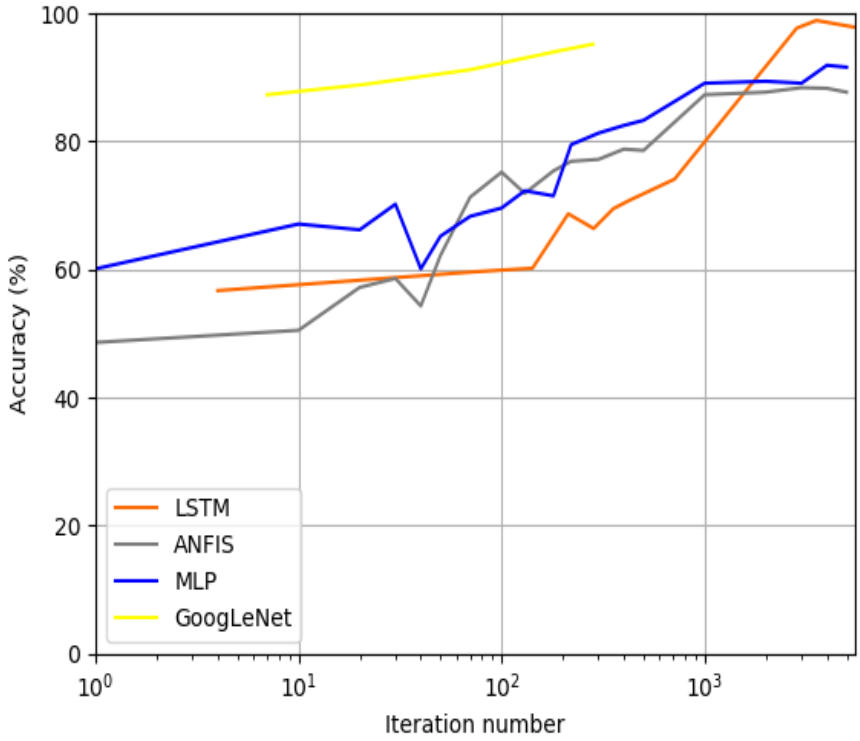


Figure 46. Number of iterations versus average training accuracy for the four systems with a 50% training ratio.

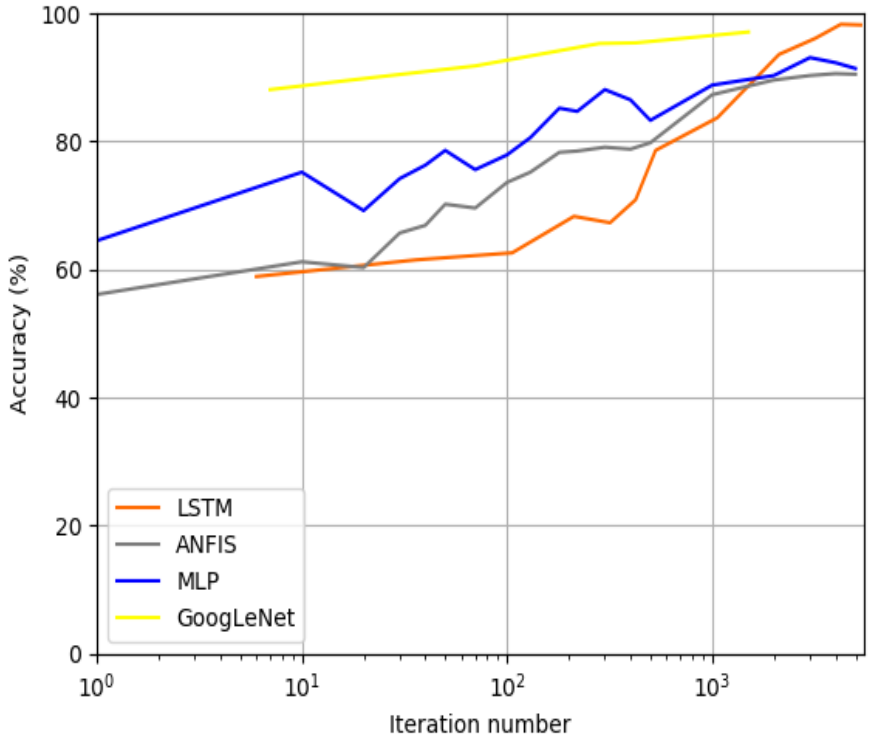


Figure 47. Number of iterations versus average training accuracy for the four systems with a 75% training ratio.

The average training accuracies versus the number of iterations are presented in Figures 44 to 47. These results show that the training accuracy of each system increases with the number of iterations. Figure 47 shows that after 10 iterations, GoogLeNet has an accuracy of around 85% while the corresponding accuracy of MLP, ANFIS, and LSTM is about 70%, 62%, and 60%, respectively. The results also show that the accuracies of the latter three systems after 1000 iterations is between 80% and 85%. After 3000 iterations, the lines cross each other. Figure 44 shows that the number of iterations for convergence with ANFIS and MLP is around 1000 while for LSTM it is around 2000. Further, the number of iterations for convergence increases with the training ratio. Comparing the results for all training ratios, MLP and ANFIS with 10% and 25% training ratios converge

after about 3000 iterations while with 50% and 75% training ratios, convergence requires about 5000 iterations.

3.4 Testing Time

A system is typically trained once and then tested multiple times. Although the training time is much longer than the testing time, the testing time may be significant in applications such as medical diagnosis. The results obtained show that the testing time of ANFIS, and MLP is less than a second while for LSTM and GoogLeNet it is 2 to 5 s, and 6 to 8 s respectively. The testing accuracy results also showed that the training ratio and the number of iterations does not affect the testing times.

Chapter 4: Conclusions and Future Work

In this chapter, the results obtained are interpreted and some conclusions are provided. Then, some future research directions are suggested.

4.1 Conclusions

In this study, Arrhythmias in ECG signals were identified using four different systems and the results were compared. The impact of the training ratio and the number of iterations on the accuracy of each system was investigated. The results obtained showed that LSTM has the highest training accuracy while the best testing accuracy was with GoogLeNet. The highest accuracies in order belong to GoogLeNet, LSTM, and MLP. On the other hand, the fastest system for the classification of ECG signals was ANFIS while MLP was the second fastest. Comparing the training times of these systems showed that although GoogLeNet has the highest accuracy, it is the most time-consuming system. Among the systems, the accuracy and training time of LSTM suggest that it is a time-efficient system for the accurate classification of ECG signals.

The results obtained showed that the accuracy is affected more by an increase in the number of iterations than the training ratio. Further, by increasing the number of iterations, the standard deviation decreases. For LSTM and GoogLeNet, the number of iterations in each epoch depends on the training ratio and minibatch size. Therefore, to achieve the same number of iterations per epoch for all training ratios, the minibatch size was increased and the results were compared for similar numbers of iterations. For example, with a training ratio of 25% and minibatch size of 100, the number of iterations per epoch is 50. In order to have the same number of iterations with a 50% training ratio, the

minibatch size should be increased to 200. Therefore, for a low number of iterations, i.e., 1 to 5, with a 75% training ratio, a large increase in the minibatch size was required. Since this study used a Core i5-7200U CPU computer with 12 GB of RAM, increasing the minibatch size above 500 was not possible due to insufficient memory.

4.2 Future Work

Classification of signals is a challenging task and ML can be applied to different biological signals such as brain and cardiac signals. The following topics can be considered in the future.

1. Classifying other signals such as brain signals using ML techniques.
2. Investigating the effect of the number of hidden layers and neurons on the accuracy and running time of MLP for signal classification.
3. Developing methods for reducing the GoogLeNet running time for processing data.
4. Extracting features from time-frequency images of signals using wavelets and Fourier transforms and comparing the results.

References

[1] K. McNamara, H. Alzubaidi, and J. Jackson, "Cardiovascular Disease as a Leading Cause of Death: How are Pharmacists Getting Involved?", Integrated Pharmacy Research and Practice, no. 8, pp. 1-11, 2019.

[2] N. T. Srinivasan and R. J. Schilling, "Sudden Cardiac Death and Arrhythmias", Arrhythmia Electrophysiological Review, vol. 7, no. 2, pp. 111-117, 2018.

[3] W. H. Maisel and L. W. Stevenson, "Atrial Fibrillation in Heart Failure: Epidemiology, Pathophysiology, and Rationale for Therapy", American Journal of Cardiology, vol. 91, no. 6, pp. 2-8, 2003.

[4] C. R. Wyndham, "Atrial Fibrillation: The most Common Arrhythmia", Texas Heart Institute Journal, vol. 27, no. 3, pp. 257-267, 2000.

[5] E. J. Benjamin, P. A. Wolf, R. B. D'Agostino, H. Silbershatz, W. B. Kannel, and D. Levy, "Impact of Atrial Fibrillation on the Risk of Death: The Framingham Heart Study", Circulation, vol. 98, no. 10, pp. 946-952, 1998.

[6] E. Y. Ding, G. M. Marcus, and D. D. McManus, "Emerging Technologies for Identifying Atrial Fibrillation", Circulation Research, vol. 127, no. 1, pp. 128-142, 2020.

[7] P. J. Podrid and P. R. Kowey, "Cardiac Arrhythmia: Mechanisms, Diagnosis, and Management", 2nd ed., Philadelphia, PA, USA, Lippincott Williams & Wilkins, 2001, pp. 973.

[8] A. Moskalenko, "Basic Mechanisms of Cardiac Arrhythmias," in Cardiac Arrhythmias - Mechanisms, Pathophysiology, and Treatment, W. S. Aronow, Ed., New York, NY, USA, Intech, 2014, sec.5, pp. 111-139.

[9] H. Zhu et al., "Automatic Multilabel Electrocardiogram Diagnosis of Heart Rhythm or Conduction Abnormalities with Deep Learning: A Cohort Study", The Lancet Digital Health, vol. 2, no. 7, pp. 1-10, 2020.

[10] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, "A Review on Deep Learning Methods for ECG Arrhythmia Classification", Expert Systems with Applications, vol. 7, pp. 2-6, 2020.

[11] A. V. Vasilakos, Y. Tang, and Y. Yao, "Neural Networks for Computer-Aided Diagnosis in Medicine: A Review", Neurocomputing, vol. 216, pp. 1-9, 2016.

[12] R. Nayak, L. Jain, and B. Ting, "Artificial Neural Networks in Biomedical Engineering: A Review", Computational Mechanics–New Frontiers for the New Millennium, pp. 887-892, 2001.

[13] B. J. Erickson, P. Korfiatis, Z. Akkus, and T. L. Kline, "Machine Learning for Medical Imaging", Radiographics, vol. 37, no. 2, pp. 505-515, 2017.

[14] J. Li, Y. Si, T. Xu, and S. Jiang, "Deep Convolutional Neural Network Based ECG Classification System using Information Fusion and One-Hot Encoding Techniques", Mathematical Problems in Engineering, vol. 2018, art. 7354081, 2018.

[15] C. J. Vrints, "Focus on Cardiac Arrhythmias and Conduction Disorders", European Heart Journal: Acute Cardiovascular Care, vol. 8, no. 2, pp. 101-103, 2019.

[16] X. L. Yang et al., "The History, Hotspots, and Trends of Electrocardiogram", Journal of Geriatric Cardiology, vol. 12, no. 4, pp. 448-456, 2015.

[17] A. H. Sykes, "A D Waller and the Electrocardiogram, 1887", British Medical Journal, vol. 294, pp. 231-245, 1987.

[18] E. N. Prystowsky, E. L. Pritchett, and J. J. Gallagher, "Origin of the Atrial Electrogram Recorded from the Esophagus", Circulation, vol. 61, no. 5, pp. 1017-1023, 1980.

[19] A. J. Moss, "A Renaissance in Electrocardiography", Annals of Noninvasive Electrocardiology, vol. 9, no. 1, pp. 1-3, 2004.

[20] D. S. Desai and S. Hajouli, "Arrhythmias." StatPearls. https://www.ncbi.nlm.nih.gov/books/NBK558923/ (Accessed Dec. 25, 2020).

[21] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, and T. Schneider, "Privacy-Preserving ECG Classification with Branching Programs and Neural Networks", IEEE Transactions on Information Forensics and Security, vol. 6, no. 2, pp. 452-468, 2011.

[22] R. Benali, F. B. Reguig, and Z. H. Slimane, "Automatic Classification of Heartbeats using Wavelet Neural Network", Journal of Medical Systems, vol. 36, no. 2, pp. 883-892, 2012.

[23] M. Kaur and A. Arora, "Classification of ECG Signals using LDA with Factor Analysis Method as Feature Reduction Technique", Journal of Medical Engineering and Technology, vol. 36, no. 8, pp. 411-420, 2012.

[24] J. Wang, W. Chiang, Y. Hsu, and Y. C. Yang, "ECG Arrhythmia Classification using a Probabilistic Neural Network with a Feature Reduction Method", Neurocomputing, vol. 116, pp. 38-45, 2013.

[25] R. J. Martis, U. R. Acharya, and L. C. Min, "ECG Beat Classification using PCA, LDA, ICA and Discrete Wavelet Transform", Biomedical Signal Processing and Control, vol. 8, no. 5, pp. 437-448, 2013.

[26] M. Vafaie, M. Ataei, and H. R. Koofigar, "Heart Diseases Prediction Based on ECG Signals' Classification using a Genetic-Fuzzy System and Dynamical Model of ECG Signals", Biomedical Signal Processing and Control, vol. 14, pp. 291-296, 2014.

[27] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks", IEEE Transactions on Biomedical Engineering, vol. 63, no. 3, pp. 664-675, 2015.

[28] M. Zihlmann, D. Perekrestenko, and M. Tschannen, "Convolutional Recurrent Neural Networks for Electrocardiogram Classification", Proceeding of Computing in Cardiology, pp. 1-4, 2017.

[29] F. Andreotti, O. Carr, M. A. Pimentel, A. Mahdi, and M. De Vos, "Comparing Feature-Based Classifiers and Convolutional Neural Networks to Detect Arrhythmia from Short Segments of ECG", Computing in Cardiology, vol. 44, pp. 1-4, 2017.

[30] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, and M. Adam, "Automated Detection of Arrhythmias using Different Intervals of Tachycardia ECG Segments with Convolutional Neural Network", Information Science, vol. 405, pp. 81-90, 2017.

[31] D. Li, J. Zhang, Q. Zhang, and X. Wei, "Classification of ECG Signals Based on 1D Convolution Neural Network", Proceedings of the IEEE International Conference on e-Health Networking, Applications and Services, 2017.

[32] S. M. Anwar, M. Gul, M. Majid, and M. Alnowami, "Arrhythmia Classification of ECG Signals using Hybrid Features", Computational and Mathematical Methods in Medicine, vol. 2018, art.1380348, 2018.

[33] Z. Xiong, M. P. Nash, E. Cheng, V. V. Fedorov, M. K. Stiles, and J. Zhao, "ECG Signal Classification for the Detection of Cardiac Arrhythmias using a Convolutional Recurrent Neural Network", Physiological Measurement, vol. 39, no. 9, art. 094006, 2018.

[34] A. Rajkumar, M. Ganesan, and R. Lavanya, "Arrhythmia Classification on ECG using Deep Learning", Proceedings of the International Conference on Advanced Computing & Communication Systems, pp. 365-369, 2019.

[35] S. Sahoo, A. Subudhi, M. Dash, and S. Sabut, "Automatic Classification of Cardiac Arrhythmias Based on Hybrid Features and Decision Tree Algorithm", International Journal of Automation and Computing, vol. 17, no. 4, pp. 551-561, 2020.

[36] A. Ullah, S. M. Anwar, M. Bilal, and R. M. Mehmood, "Classification of Arrhythmia by using Deep Learning with 2-D ECG Spectral Image Representation", Remote Sensing, vol. 12, no. 10, art. 1685, 2020.

[37] G. Jayagopi and S. Pushpa, "Automatic Detection of Arrhythmia using Optimized Feature Selection", Proceedings of the International Conference on Data Science, Machine Learning and Applications, pp. 1731-1742, 2020.

[38] P. S. Addison, The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance, 1st ed., Boca Raton, FL, USA, CRC Press, 2002, pp. 2-14.

[39] P. S. Addison, "Wavelet Transforms and the ECG: A Review", Physiological Measurement, vol. 26, no. 5, pp. 155-199, 2005.

[40] R. Merry, "Wavelet Theory and Applications: A Literature Study", M.S. Thesis, Mechanical Engineering, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2005.

[41] V. Seena and J. Yomas, "A Review on Feature Extraction and Denoising of ECG Signal using Wavelet Transform", Proceedings of the International Conference on Devices, Circuits and Systems, pp. 1-6, 2014.

[42] S. S. Haykin, Neural Networks and Learning Machines, 3rd ed., Hoboken, NJ, USA, Pearson Education, 2009.

[43] F. Bouaziz and D. Boutana, "Automated ECG Heartbeat Classification by Combining a Multilayer Perceptron Neural Network with Enhanced Particle Swarm Optimization Algorithm", Research on Biomedical Engineering, vol. 35, no. 2, pp. 143-155, 2019.

[44] A. Baghban, R. Hekmati, M. Hajiali, M. Janghorban Lariche, and M. Kamyab, "Application of MLP-ANN as Novel Tool for Estimation of Effect of Inhibitors on Asphaltene Precipitation Reduction", Petroleum Science and Technology, vol. 36, no. 16, pp. 1272-1277, 2018.

[45] M. W. Gardner and S. Dorling, "Artificial Neural Networks (the Multilayer Perceptron)-a Review of Applications in the Atmospheric Sciences", Atmospheric Environment, vol. 32, no. 14-15, pp. 2627-2636, 1998.

[46] A. Abraham, "Adaptation of Fuzzy Inference System using Neural Learning," in Fuzzy Systems Engineering, Springer, Berlin, Germany, pp. 53-83, 2005.

[47] J. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System", IEEE Transactions on Systems, Man, Cybernetics, vol. 23, no. 3, pp. 665-685, 1993.

[48] S. Emamgholizadeh, K. Moslemi, and G. Karami, "Prediction the Groundwater Level of Bastam Plain (Iran) by Artificial Neural Network (ANN) and Adaptive Neuro-Fuzzy Inference System (ANFIS)", Water Resources Management, vol. 28, no. 15, pp. 5433-5446, 2014.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors", Nature, vol. 323, no. 6088, pp. 533-536, 1986.

[50] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", Neural Computing, vol. 9, no. 8, pp. 1735-1780, 1997.

[51] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network", IEEE Transactions on Smart Grid, vol. 10, no. 1, pp. 841-851, 2017.

[52] R. Vargas, A. Mosavi, and R. Ruiz, "Deep Learning: A Review", Advances in Intelligent Systems and Computing, 2017.

[53] D. Xie, L. Zhang, and L. Bai, "Deep Learning in Visual Computing and Signal Processing", Applied Computational Intelligence and Soft Computing, vol. 2017, art. 1320780, 2017.

[54] R. Sustika, A. Subekti, H. F. Pardede, E. Suryawati, O. Mahendra, and S. Yuwana, "Evaluation of Deep Convolutional Neural Network Architectures for Strawberry Quality

Inspection", International Journal of Engineering Research and Technology, vol. 7, no. 4, pp. 75-80, 2018.

[55] H. Habibi Aghdam, E. Jahani Heravi and D. Puig, "A Practical Approach for Detection and Classification of Traffic Signs using Convolutional Neural Networks", Robotics and Autonomies Systems, vol. 84, pp. 97-112, 2016.

[56] C. Li, H. Zhang, P. Wu, Y. Yin, and S. Liu, "A Complex Junction Recognition Method Based on GoogLeNet Model", Transactions in Geographic Information Science, vol. 24, no. 6, pp. 1756-1778, 2020.

[57] S. P. K. Karri, D. Chakraborty, and J. Chatterjee, "Transfer Learning Based Classification of Optical Coherence Tomography Images with Diabetic Macular Edema and Dry Age-Related Macular Degeneration", Biomedical Optics Express, vol. 8, no. 2, pp. 579-592, 2017.

[58] F. Basso, L. J. Basso, F. Bravo, and R. Pezoa, "Real-time Crash Prediction in an Urban Expressway using Disaggregated Data", Transportation Research Part C: Emerging Technologies, vol. 86, pp. 202-219, 2018.

[59] D. Acharya, A. Rani, S. Agarwal, and V. Singh, "Application of Adaptive Savitzky–Golay Filter for EEG Signal Processing", Perspectives in Science, vol. 8, pp. 677-679, 2016.

[60] M. de Rooij and W. Weeda, "Cross-Validation: A Method Every Psychologist Should Know", Advances in Methods and Practices in Psychological Science, vol. 3, no. 2, pp. 248-263, 2020.