CHAPTER 6

PERCEPTRON BOOSTING AND CONVOLUTIONAL NEURAL MEMETIC OPTIMIZED U-NET LEARNING FOR DISEASE PREDICTION

6.1 INTRODUCTION

The spread of disease across the country is varied and has had a significant impact on the lives of people. The slow and delayed prediction leads to a rapid increase in disease cases in a given geographical region. A fast and accurate diagnostic test helps quickly detect infected patients and provide them with the necessary treatment. During a pandemic, the most important concern is to identify patients at risk of high mortality at an early stage and to administer suitable treatments. Numerous research studies have been conducted to predict diseases in a presymptomatic stage. Disease prediction in the presymptomatic stage refers to the ability to recognise the likelihood of developing a disease before any symptoms appear. Predicting a disease initially can substantially enhance treatment outputs, reduce healthcare costs, and enhance individual and public health through early intervention and informed decision-making. However, accurate and time-efficient predictions are a major challenging issue. To overcome these problems, new techniques are needed for accurate disease prediction.

A novel Emphasis Perceptron Boosting Classification (EPBC) technique is proposed for disease prediction with better accuracy. The proposed EPBC technique takes patient data with selected features (patterns) as input. With this input, accurate grouping is achieved with the assistance of boosting algorithm. The introduced EPBC is designed to enhance the accuracy of disease prediction in a shorter timeframe. With this intent, initially, weak classifiers, i.e., perceptron binary classifiers, are constructed through the weighted sum. The designed classifiers spilt the patterns through zero training error. Weight, as well as feature vector objective function, is derived from providing forecasted outcomes at the premature phase. From that, real classification outcomes are obtained in a precise way.

A time-dependent Cox regressive Levenberg-Marquardt convolutional neural learning technique (TCLMCNL) is proposed for classifying patient data in disease prediction. The TCLMCNL Technique includes an input layer, an output layer, and one or more hidden layers. First, several relevant and significant features with associated information are considered as input in the input layer. Later, the input is passed to a hidden layer for classification. In this layer, time-dependent Cox regression is used to compute Cramér's phi correlation function. According to the regression results, precise classification is performed. For each time step, Huber loss is determined for forecasting and obtaining observed results. Additionally, the Levenberg–Marquardt method is examined to minimise the mistake. Finally, this output layer returns the prediction results for disease identification. As a result, the accuracy of prediction is improved using TCLMCNL Technique.

Memetic Optimized U-Net Deep Learning (MO-UNetDL) classifier technique is developed for disease prediction by means of classification process. MO-UNetDL technique consists of many layers for analyzing patient data samples. Input layer takes selected features from database as input. Then, the weights are assigned to a set of input information samples in convolution layer to incorporate bias function. In this stage, Wilcox's index coefficient is employed to determine the similarity. Then index coefficient is provided to soft step activation in the hidden layer where it examines the index coefficient value and provides the outcomes as any one of ‘1 or ‘0’. If the result of activation function is ‘1’, then the disease is appropriately diagnosed. Subsequently, max-pooling operation is carried out to decrease dimension of data samples to the next step where higher activation function values are considered for the next process. After that, the upsampling is carried out to improve the dimension of data as well as lastly classification results are obtained. To minimize loss of data classification, memetic optimization is applied in MO-UNetDL for tuning the hyperparameters. Initially, the numbers of individuals are initialized, and their fitness is measured. The truncation selection selects the top individual, two-point crossover produces novel offspring, and the input bit string is randomly exchanged through bit-flip mutation. Once genetic operators are executed, the selected individual is estimated, and fitness is verified. This process is repeated until an optimal solution is

achieved. From this, the memetic-optimised U-Net deep learning classifier detects an optimal hyperparameter to reduce error, resulting in increased accuracy of disease prediction.

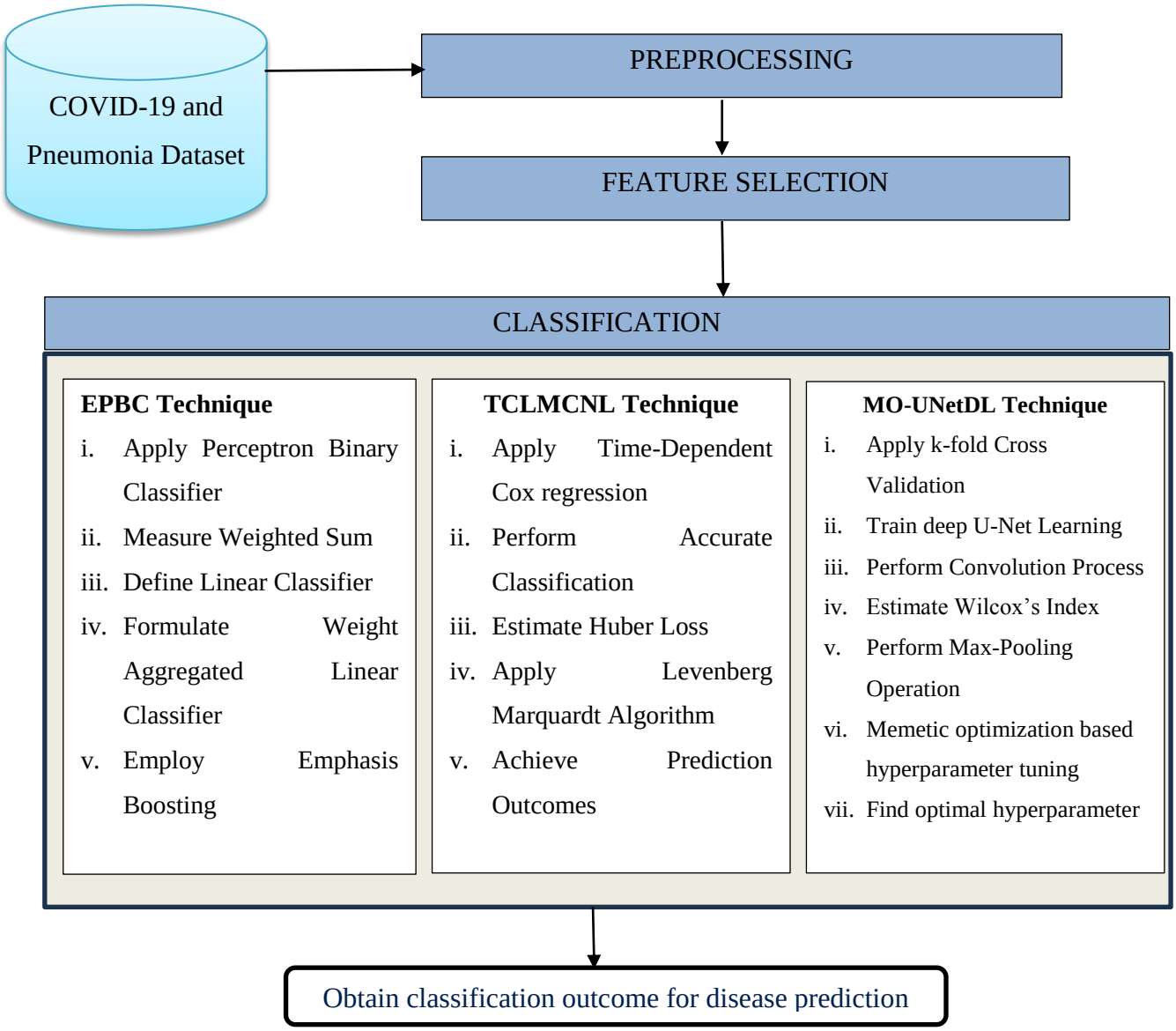

Figure 6.1 Step-by-step Classification Framework

Figure 6.1 shows the step-by-step classification framework. After preprocessing and feature selection, the classification techniques EPBC, TCLMCNL, and MO-UNetDL, have been developed for accurate disease identification with greater accuracy with reduced time.

6.2 PROPOSED EMPHASIS PERCEPTRON BOOSTING CLASSIFICATION

The accurate clinical classification of patient data enables disease forecasting at an initial stage. The classification of data is carried out using optimal and error-minimised relevant features. The primary aim of accurate and early classification is to facilitate instant observation of patients. Hence, normal people are not tainted, and through the premature classification, rapid recovery is ensured. Motivated by this, the EPBC technique is proposed to categorise patient data with superior accuracy in minimal time. EPBC technique is employed on chosen pertinent patterns to classify disease patient files.

Boosting is an ML method applied to enhance the performance of a predictive model. It works by combining multiple weak learners into a single strong learner, providing robust classification results. In the proposed EPBC technique, Emphasis Boost Classifier is employed to categorize patient information. The Emphasis Boost Classifier utilises the perceptron binary classifier as a weak learner to categorise patient information. These classified results are merged using the Emphasis Boost Classifier to create a robust classifier.

In the proposed EPBC technique, a perceptron binary classifier is employed to categorize patient data files. It observes whether input is represented through the vector of numbers regarding exact class (diseased or normal) or not. A perceptron binary classifier categorises data into various groups, such as diseased or non-diseased. Perceptron binary classifier performs detection according to linear predictor function incorporating weights through feature vector. With this, disease prediction is achieved with greater accuracy in the shortest time possible.

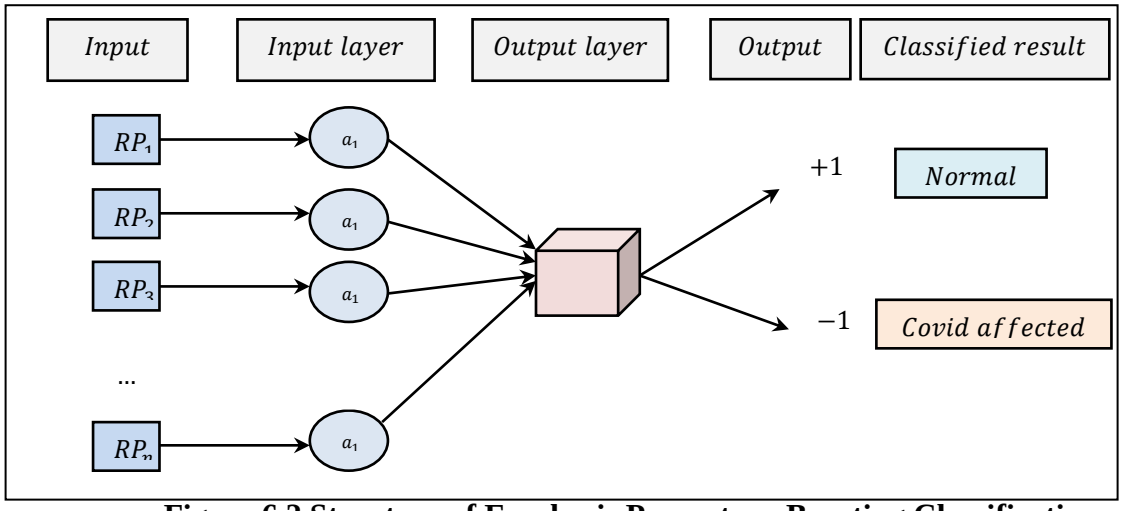

Figure 6.2 Structure of Emphasis Perceptron Boosting Classification

Figure 6.2 depicts the Structure of emphasis perceptron binary classifier, for classifying the patient data with higher accuracy. Perceptron binary classifier is taken as weak classifier, includes four parts. They are input values (i.e., relevant patterns 'RP'), weight ('W'), bias ('B') and activation function ('AF'). As shown in Figure 6.1, each input values or 'RP' are multiplied through their respective weight 'W' as follows.

$$WS = \sum_{i=1}^{n} RP_i W_i = (rp_1 w_1 + rp_2 w_2 + \cdots + rp_n w_n) \tag{6.1}$$

By the above equation (6.1), weighted sum 'WS' is acquired and is depended on index 'i' as well as sigma for summation. As mentioned earlier, perceptron are weak learner through linearly separable problems. A linear classifier which partitions pattern through zero training error are presented as follows.

$$p + p_0 \begin{cases} > 0, \forall a \; in \; P_i(\; class\; [+1]) \\ < 0, \forall a \; in \; P_i(\; class\; [-1] \end{cases} \tag{6.2}$$

From the above equation (6.2), 'p', and 'p_0' indicates parameters which model linear classifier. Lastly, 'WS' through feature vectors 'RP' are merged to provide linear classifier by zero training error and it expressed as follows.

$$fun\;(a; p) = sign\;(a^T, p) \forall i = 1,2,3, \ldots, n \tag{6.3}$$

Lastly, Emphasis Perceptron Boosting algorithm reduces objective function for detecting the correct label for pertinent patterns, as expressed as follows.

$$obj = -\sum_{i=1}^{n} [b_i fun\;(a; p)] 1 \left\{ b_{\;i} \neq fun\;(a; p) \right\} \tag{6.4}$$

If forecasted value '$fun\;(a; p)$' as well as labels 'b_i' contain similar sign then dot product '$b_i fun\;(a; p)$' would '> 0'. This states with above defined linear classifier '$fun\;(a; p)$' and the predicted outcomes is proper and accurate for 'a_i' data-point.

//**Algorithm 6.1: Emphasis Perceptron Boosting Classification**
Input: Dataset 'DS', district level counts 'DLC', Patient-wise data 'PD', patient number 'PID' **Output**: Accurate classified results
step 1: **Initialize** relevant patterns '$RP = rp_1, rp_2, \dots, rp_n$', Weight '$W = w_1, w_2, \dots, w_n$' **step 2**: **Begin** **step 3**:**For** each Dataset 'DS' with district level counts 'DLC' obtained from Patient-wise data 'PD'patient files with patient number 'PID' **step 4**: Evaluate weighted sum as in equation (6.1) **step 5**: Formulate linear classifier as in equation (6.2) **step 6**: Formulate weight aggregated linear classifier as in equation (6.3) **step 7**: Evaluate objective function as in equation (6.4) **step 8**: **Return** classified results 'b_i' **step 9**: **End for** **step10**: **End**

Algorithm 6.1 illustrates the Emphasis Perceptron Boosting Classification, aiming to achieve early disease prediction with higher accuracy. First, a weak classifier, depending on the perceptron function, is modelled. After that, the emphasis boosting is generated with the aid of a weighted sum, which partitions the pattern through zero training error. Lastly, the objective function is formulated by combining the weight and feature vector to provide forecasted results at the premature phase. From that, actual classified outputs are acquitted in a precise way.

6.3. TIME-DEPENDENT COX REGRESSIVE LEVENBERG-MARQUART CONVOLUTIONAL NEURAL LEARNING TECHNIQUE

A TCLMCNL Technique is proposed to classify patient data for early identification of disease. The TCLMCNL technique is a deep learning classifier that examines aspects in depth via numerous layers and provides precise categorisation outcomes. The Cox regression is employed in a convolutional classifier for feature analysis and yields output consequences in lower time.

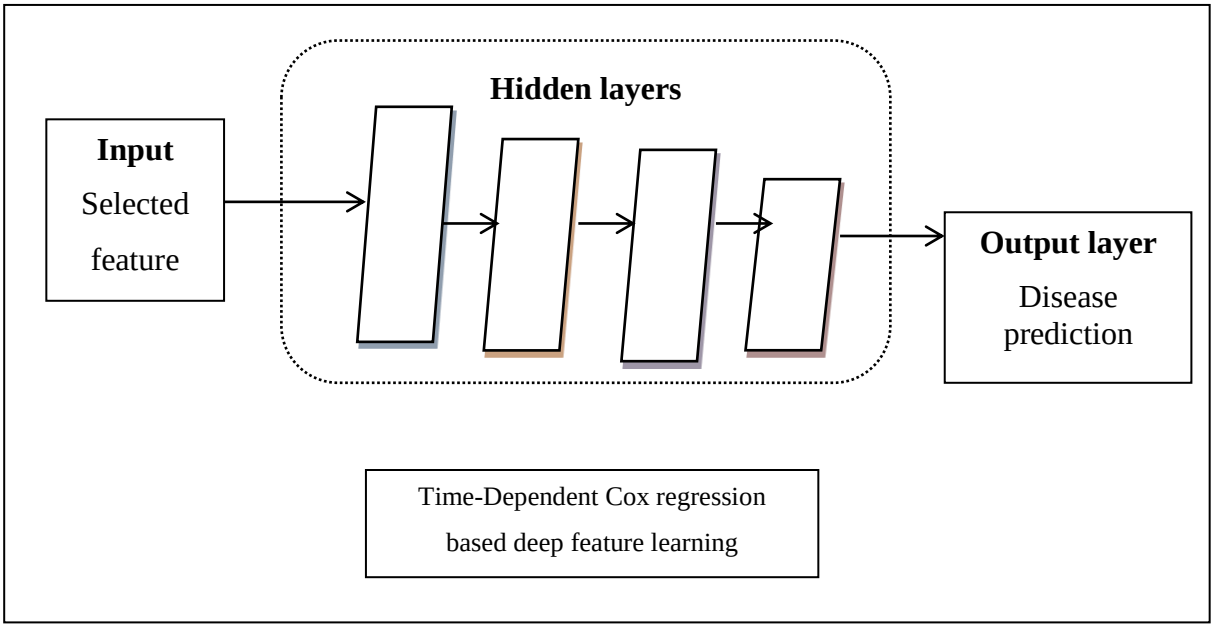

Figure 6.3 Structure of Time-dependent Cox regressive Levenberg-Marquart Convolutional Neural Learning Classifier (TCLMCNL)

Figure 6.3 illustrates the structure of TCLMCNL classifier technique, which accurately classifies patient data by extracting patterns or features for disease prediction. The designed classifier includes three kinds of layers. The input layer of the DL classifier receives input and passes it to the scheme for additional processing in the successive layers of artificial neurons. The input layer is located in the start of the DL network. The output layer is completely associated with alternate consecutive layers in a feed-forward way through the corresponding set of weights to generate the whole network.

The input layer comprises chosen aspects and patient data as input, and these aspects are observed through hidden layers. According to the regression results, efficient classification is performed. In addition, Huber loss is computed (to decrease the error rate) in each time step for predicted and actual classification results. Levenberg–Marquardt algorithm is applied in the proposed TCLMCNL classifier to decrease the error rate during the classification process. Later, forecast outcomes are obtained at the output layer.

Consider the input layer, it obtains selected features $f_1, f_2, f_3, \ldots f_k$ and patient information as input. Then, activity of neuron at the input layer '$z(t)$' is formulated as follows.

$$z(t) = D + \left[\sum_{i=1}^{k} f_i(t) * q_{input}\right] \quad (6.5)$$

In the above equation (6.5), 'D' denotes the bias whose value is '1', '$f_i(t)$' refers to input features, 'q_{input}' refers weight. Later, input is sent to the hidden layers, where feature learning is performed using correspondance correlated Cox regressive. In hidden layer, CNL classifier comprises max-pooling layer that reduces dimensions of information via integrating input at single layer and transforming to subsequent layer.

Time-dependent cox regression method is ML method which finds an association between time-to-event outcome Y and set of explanatory variables. Outcome of disease forecast is obtained applying time-dependent cox regression model, which accounts for the passage of time.

$$Y\,(t) = g_o(t).\exp\,(\rho_c.R) \quad (6.6)$$

Where '$Y\,(t)$' refers to hazards function at times, '$g_o(t)$' refers covariate vector, 'ρ_c' refers regression coefficient, 'R' refers Cramér's phi correlation function. Cramér's phi correlation is applied to calculate the association between two variables, as it is given below.

$$R = \left[\frac{\sum_{i=1}^{k}\sum_{j=1}^{m}|f_k - f_j|^2}{(n-1)+(m-1)}\right] \quad (6.7)$$

Where 'R' symbolizes Cramér's phi test results, aspects 'f_k'symbolizes chosen aspect, 'f_j' symbolizes testing disease patterns, and 'n, m' are sample sizes. The results of Cramér's phi test ranges as 0 to 1. If Cramér's phi test gives an output value as '0', then the features have no association. Otherwise, the Cramér's phi test provides '1' as output, then the absolute association between the features is identified.

$$R = \begin{cases} 1\ ; & f_k \quad isassociated \quad with \quad f_j \\ 0\ ; & no\ association \quad between \quad f_k and\ f_j \end{cases} \quad (6.8)$$

Where output of R gives '1' refers to which aspects are comparable and the patient information is correctly categorized as diseased or normal. In contrast, the value of '0' refers that the patient data are incorrectly classified. As a result, accurate classification is attained with minimum time complexity. Lastly, the results of hidden layer are formulated as below.

$$P\ (t) = \left[\sum_{i=1}^{k} f_i(t) * q_{input}\right] + [q_{\text{hidden}} * P_{h-1}] \tag{6.9}$$

Where '$P(t)$' denotes hidden layer output, 'R_i' refers to weight, 'R_{h-1}' refers previous hidden layer, '$*$' indicates convolutional operator. Lastly, output is sent to hidden layer as well as customized Huber loss function is employed to decrease error rate as follows.

$$L_h = \frac{1}{2}[A_o - P_o]^2 \tag{6.10}$$

In the above equation (6.10), 'L_h' refers a Loss, 'A_o' denotes actual results, 'P_o' denotes predicted classification outputs. TCLMCNL classifier utilizes Levenberg–Marquardt method for detecting local minimum as below.

$$F = arg\min L_h \tag{6.11}$$

Where '$argmin$' indicates argument of minimum function, and 'L_h' denotes loss. When lesser error is achieved, outcomes are exposed in output layer as given below.

$$Z = \sum_{i=1}^{n} P(t) * q_{\text{hidden out}} \tag{6.12}$$

From the above Equation (6.12), '$P(t)$' indicates output of classification outcome, '$P(t)$' indicates hidden layer output, '$q_{\text{hidden out}}$' refers a weight among hidden and output layers. Hence, precise classification is acquired at output layer.

//Algorithm 6.2: Time-Dependent Cox regressive Levenberg–Marquardt Convolutional neural learning classifier-based disease prediction

Input: selected features '$f_1, f_2, f_3, \ldots . f_k$', Patient data samples $D = D_1, D_2, \ldots, D_n$

Output: Increase prediction accuracy

Begin

step 1. **Collect the relevant** features $f_1, f_2, f_3, \ldots . f_k$ in input layer

step 2: **For** each testing feature 'f_i' --**hidden layer**

step 3: **For** each training feature 'f_k'

step 4: Apply Time-Dependent Cox regression

step 5: **If** $(R = 1)$**then**

step 6: patient Data sample is classified as a diseased

step 7: **else**

step 8: patient Data sample is classified as a normal

step 9: **End if**

step 10: **End for**

step 11: **End for**

step 12: **For each time step 't'**

step 13: Measure Huber loss 'L_h'

step 14: Apply –the Levenberg Marquardt algorithm to find the minimum loss

step 15: $\boldsymbol{if}$(arg min L_h) **then**

step 16: Obtain accurate classification results at the output layer

step 17: **else**

step 18: Repeat step 2

step 19: **End if**

step 20: **End for**

End

Algorithm 6.2 describes processes of disease detection using TCLMCNL classifier technique. Initially, several pertinent important aspects is provided as input. By using chosen training feature values, categorization is carried out by examining both testing disease patterns

and training information. According to the time-dependent Cox regression, precise categorization is carried out using Cramér's phi correlation. Additionally, the Huber loss is calculated for each classified results and observed results. In addition, the Levenberg–Marquardt method is employed to determine the minimum error classification results. Finally, predicted outcomes are obtained, thereby enhance the accuracy of disease prediction.

6.4 PROPOSED MEMETIC OPTIMIZED U-NET DEEP LEARNING CLASSIFIER

A novel and efficient technique, called the Memetic Optimised U-Net Deep Learning (MO-UNetDL) classifier, is proposed for patient data classification in disease prediction. The U-Net is an enhanced version of the deep convolutional network, where the architecture is dependent on the training data to achieve more accurate classification results than the existing convolutional network. The major advantage of U-Net is that it is computationally efficient to train a huge dataset.

Figure 6.4 depicts the Memetic optimized U-Net deep learning classifier for patient data sample classification. The proposed MO-UNetDL classifier utilises the input features and their corresponding data for the prediction process. Initially, the K-fold Cross-Validation method is applied to choose the testing and training data from the dataset. Upon spitting the dataset, the training of the deep U-Net learning model is initiated, and patient information is classified as normal or diseased. During the classification process, the hyperparameter of the DL method is optimised using the Memetic algorithm to achieve higher classification accuracy with lower error. Lastly, accurate patient data classification outcomes are obtained.

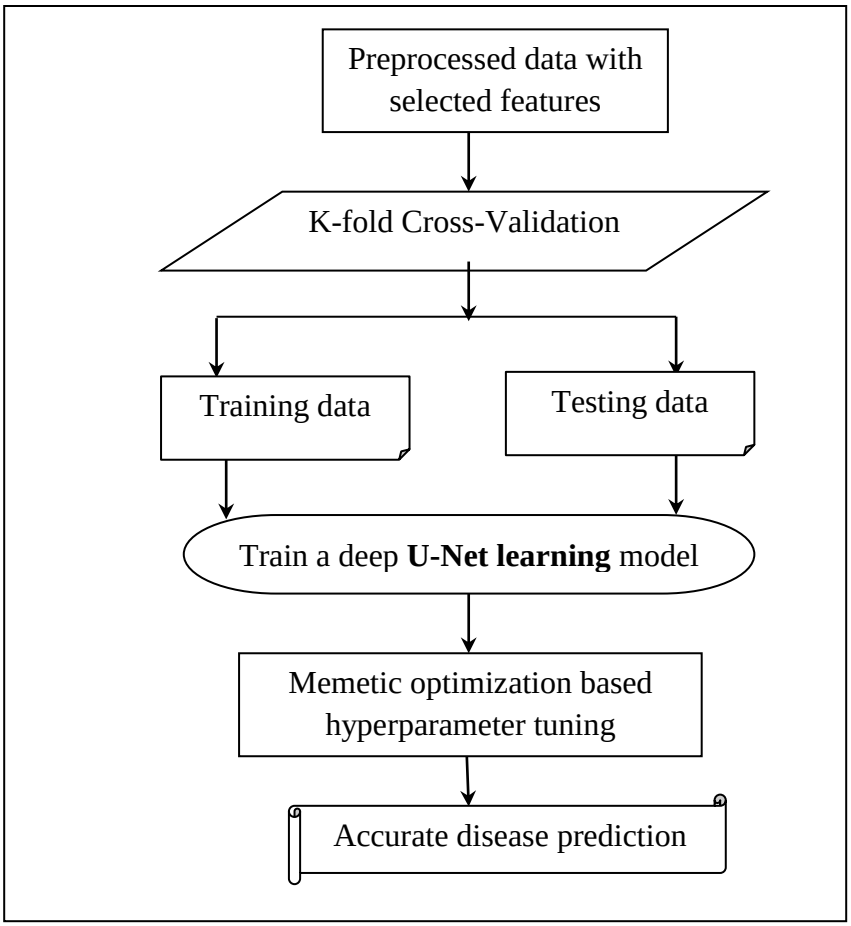

Figure 6.4 Flow Process of Memetic Optimized U-Net Deep Learning Classifier for Disease Prediction

6.4.1 K-fold Cross-Validation

It is a statistical model which computes DL model capacity. Generally, it is employed at DL to compare and select the training and testing data for a given predictive problem, as it is simple to carry out data classification.

Consider number of patient data samples $D_1, D_2, \ldots, D_n$. This process has a single factor referred to as 'k' that denotes a number of partitioned groups of given patient data samples. Thus, it is represented as k-fold cross-validation. Primarily, database is partitioned into 'k' number of groups (iterations i.e. k =10). For each different group, use test data as well as other

data as a training data set. Next, a deep U-Net learning method is employed on training set and evaluated on test set for getting certain classification outcomes.

6.4.2 Memetic Optimized U-Net Deep Learning Model

Upon completing the cross-validation**,** classification is performed based on the training and testing databases. The proposed MO-UNetDL classifier employs the deep U-Net learning method to categorise sample information based on selected features. A deep U-Net learning method comprises many convolutional layers and fully connected layers, as well as weights, max pooling layers, and activation functions.

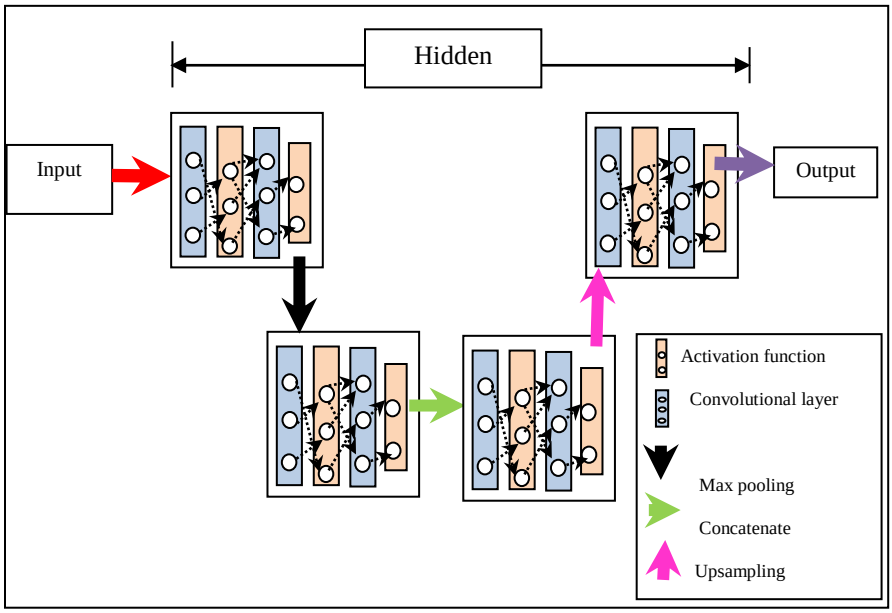

Figure 6.5 Schematic Structure of Deep U-Net Learning Model

Figure 6.5 depicts a schematic structural diagram of a deep U-Net learning classifier with various kinds of layers. The proposed deep neural network (NN) includes neuron-like nodes. Here, the output of one layer is completely associated with the succeeding layers. Figure 6.5 looks like a 'U' shape, which describes its U-Net learning classifier. The architecture consists of two sections: a contraction section (i.e., the left side) and an expansion section (i.e., the right side). The contraction section is prepared using a stack of many convolutional layers. Then, the soft step activation functions are used to provide non-linearity to the method, and max-pooling layers are employed for down sampling. The expansion section (i.e., the right side) is also

prepared using a stack of many convolutional layers, followed by soft step activation functions for upsampling. The respective output from the contraction section is connected with the respective block in the expansion section. In the U-Net learning classifier model, the count of expansion segment is equivalent to the count of contraction blocks.

The contraction section has the input of number of training data samples $D = D_1, D_2, \dots, D_n$ and predictions are produced from the final output layer following the expansive path. Convolutional layers convolve input data samples as well as send its output to subsequent layer. This layer carry outs a mathematical operation referred as "convolution". Convolution is linear operation which contributes to the multiplication of set of weights through input data samples. Product is performed among array of source data as well as set of weights.

$$Y = \sum(w_j * D_i + bias) \tag{6.13}$$

Where 'Y' indicates a Convolutional layer output, 'w_j' symbolizes an array of weights, 'D_i' symbolizes the input data samples added with bias, '$*$' symbolizes a mathematical referred "convolution". After that, the input data samples are sent to the next layer. Here, the feature mapping is performed by employing a Wilcox index to a qualitative data. Wilcox's index coefficient is the computation of statistical dispersion in nominal data distribution. The coefficient is also employed to carry out the feature mapping between the training and testing data samples.

$$Q = 1 - \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} |D_i - D_j| \tag{6.14}$$

Where 'Q' refers a Wilcox's index coefficient, 'n' refers the count of training samples 'D_i', and 'D_j' refers a testing sample. The results of Wilcox's index coefficient are varied between 0 to 1. Soft step activation is applied to examine similarity value. The benefit of using activation function is to help network for learning complex training and testing data samples. This function outputs the best-normalized function in the form of 1 and 0, generating an accurate disease prediction. Also, the activation function gives a bounded absolute value. Thus, the introduced deep learning classifier utilizes the soft step activation function 'A' to achieve precise disease prediction.

$$A = \frac{1}{1+\exp(-Q)} \tag{6.15}$$

Where 'Q' refers the coefficient results. When the feature maps are generated using convolution layers, the dimensionality of the data needs to be minimized to attaining better data classification. This is addressed by using Max Pooling operation. Later, the correlation values are forwarded to the convolution pooling layer, where highest value of the activation function is chosen to perform predictive analysis, thereby reducing the dimensions of data samples. Thus, the process is referred as max pooling operations.

$$Z = \begin{cases} A > \delta\ ; & Selected \\ A < \delta; & Removed \end{cases} \tag{6.16}$$

Where 'Z' refers to the output of pooling layer, 'δ' refers a threshold, and 'A' refers a maximum value of activation function. Values higher than threshold are selected for further processing. Otherwise, the outputs are eliminated. According to the activation results, the disease is accurately determined.

$$A = \begin{cases} 1; Disease\ diagnosed \\ 0\ ; \qquad\qquad normal \end{cases} \tag{6.17}$$

Based on the similarity in both testing and training data samples, the sigmoid activation operation gives the result as '1', then the disease is diagnosed. Otherwise, the data samples are classified as normal.

Following this, the deep U-Net learning classifier performs the upsampling, which is an uppooling operation. It generates a larger feature map closer to the input, resulting in a more detailed output. The feature mapping process in the upsampling process is carried out using Wilcox's index coefficient, and then the outputs are applied to the activation function. It provides classification outcomes by analysing all Wilcoxon's index coefficient results, either minimum or maximum, compared to the threshold values. This, in turn, enables each data sample to be classified, allowing for accurate disease prediction.

With the classified results, the cross-entropy loss is estimated for evaluate the performance of a categorization output.

$$LL = -[\,Y\,ln\,Q + (1-Y)ln\;(1-Q)] \tag{6.18}$$

Where 'LL' symbolizes a log loss, 'Y' symbolizes actual classification results, 'Q' symbolizes predicted classification results, 'ln' indicates the natural logarithm of a number. Depending on the loss, the weight is updated as given below.

$$\nabla w = w * \eta\left(\frac{\partial LL}{\partial w}\right) \tag{6.19}$$

Where '∇w' refers an updated weight, 'w' refers a current weight, 'η' refers learning rate($\eta < 1$). Large value of learning rate allows models to study quicker than the smaller value, '$\frac{\partial LL}{\partial w}$' suggests a partial differential of the log loss 'LL' with corresponding to the present weight 'w'.

The hyperparameters of the U-Net learning classifier are optimised by an Evolutionary algorithm called memetic optimisation to minimise the loss of disease prediction. To address the optimisation problems, magnetic optimisation is applied to reduce the objective function. Memetic optimisation is a population-based approach to determining the optimal solution (i.e., epochs, learning rate, and weight) for minimising loss and improving prediction accuracy.

First, the population of hyperparameter values is initialised in the form of weight values, the count of epochs the learning rate value. After the initialization, memetic fitness is determined based on the loss .

$$F = \arg\min\; LL \tag{6.20}$$

Where 'F' indicates a fitness function, 'arg min' indicates argument of minimum function, and 'LL' refers to cross-entropy loss. Depended on the fitness, bio-inspired memetic operators in particular truncation selection, two-point crossover, and bit flip mutation, are performed to identify the optimal hyperparameters for decreasing the loss and increasing the disease prediction accuracy. These three processes of the memetic operators are described in the below sections.

Truncation Selection

It is the basic method in the memetic optimization algorithm to select parent chromosomes from initial population based on higher fitness by applying truncation selection to produce the new generation. First, the chromosomes are ranked based on current fitness (F). Once it is ranked, the truncation threshold is fixed to choose the best individuals. Individual's fitness above the threshold is chosen as parents to the next recombination process. The chromosome that is below the threshold is not able to carry out the recombination to generate offspring.

$$\mathrm{T} = \begin{cases} \mathrm{F} > \mathrm{F_{th}}\ ; & \text{selected as parent chromosomes} \\ \mathrm{F} < \mathrm{F_{th}}\ ; & \text{not selected as parent chromosomes} \end{cases} \tag{6.21}$$

From the above equation (6.21), the parent chromosomes truncation selection indicates as 'T' and 'F_{th}' denotes the threshold. Accordingly, recombination procedure is performed to choose greatest individual.

Two Point Crossover

Followed by the parent chromosome selection, the two-point crossover is carried out to produce the offspring. Figure 6.6 illustrates the example process of two-point crossover for offspring generation.

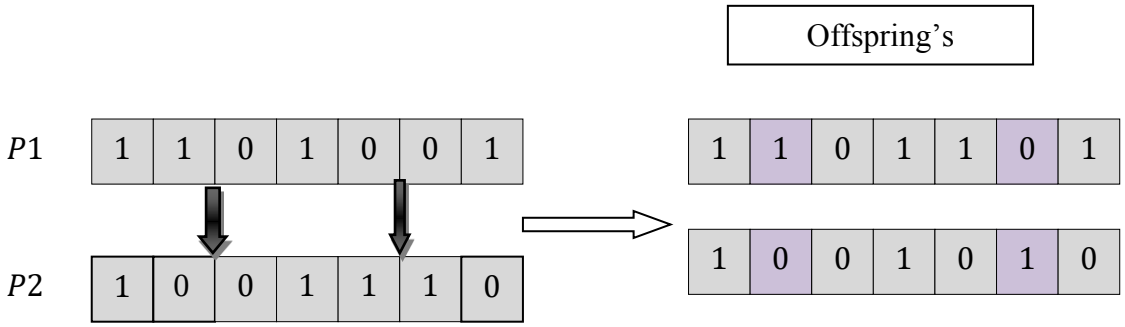

Figure 6.6 Two-Point Crossover

Two crossover points are randomly selected from parent chromosomes '$P1$,' and '$P2$' in the two-point crossover. The bits between the two points are changed between the parent organisms and create the two offspring which is shown in Figure 6.6.

Bit Flip Mutation

Lastly, the bit flip mutation is used to manage genetic diversity from one generation to the next. Genetic diversity denotes the inconsistency present within chromosomes. The chromosomes in the mutation process consider offspring generated from the crossover as input. The changes in the bit flips mutation are 0 to 1 and 1 to 0 in the chromosomes. Figure 6.7 shows the mutation of the bits in the chromosome.

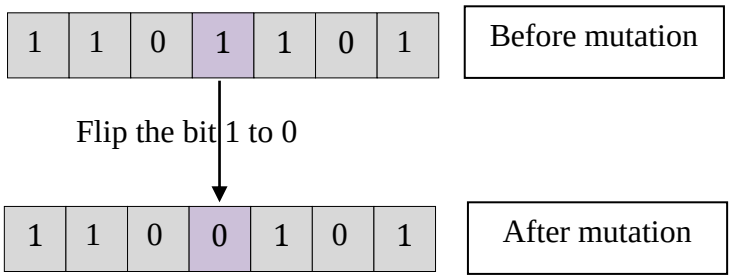

Figure 6.7 Bit Flip Mutation Process

Figure 6.7 demonstrates the Adaptive bit Flip Mutation of offspring value '1' is randomly changed by a precise chromosome string of '1'. Mutation includes random altering bits to produce new offspring via selecting the best optimal. Next, the previously chosen individual is replaced with the new optimal. The fitness criterion is verified again for the newly produced optimal solution. This procedure is repeated for all individuals in the population, detecting the optimal parameters that minimise the error rate. Lastly, accurate classification outcomes are acquired by the output layer with smaller errors. The algorithm for the Memetic-optimised U-Net deep learning classifier for disease identification is presented as follows.

// **Algorithm 6.3: Memetic optimized U-Net deep learning classifier**
Input: Selected relevant features $f_k = f_1, f_2, \dots, f_k$ and training data samples $D = D_1, D_2, \dots, D_n$ **Output:** Increase the disease prediction accuracy
Begin **step 1: Collect the selected features** $f_k = f_1, f_2, \dots, f_k$ with data samples

step 2: **Apply** k fold cross-validation for splitting the data sent into testing and training

step 3: Training data are given as input to the input layer of the U-Net deep learning classifier

step 4: **For each training data [hidden layer]**

step 5: **Perform the convolution process** with weight 'α_i' and add bias 'g'

step 6: **For** each training data with testing disease data

step 7: Measure the Wilcox's index coefficient 'Q'

step 8: Apply soft step activation function 'A'

step 9: Perform a max-pooling operation (6.16)

step 10: Select the best value and remove others (6.17)

step 11: **If** ($A = 1$) **then**

step 12: Correctly predicted as disease

step 13: **Else**

step 14: Correctly predicted as normal

step 15: **End if**

step 16: Perform upsampling process for predicting all the data samples

step 17: **For each classification results**

step 18: Measure the cross-entropy loss 'LL'

step 19: Update the weight '∇w'

step 20: Find minimum loss by tuning hyperparameters using memetic optimization

step 21: Initialize the populations of hyperparameters such as weight, epochs, and learning rate

step 22: **For each** individual in the population

step 23: Calculate the fitness 'F' (6.20)

step 24: **If** ($arg\min LL$) **then**

step 25: Selects an individual from the population

step 26: **End if**

step 27: Select the individual from the population

step 28: Generate new offspring using a two-point operator

step 29: Perform bit flip mutation for inverting the bit

step 30: Replace the old individual with a new one

step 31: Go to step 24

step 32: Terminate the algorithm until the optimal solution is obtained

step 33: **Return (Optimal hyperparameters)**

step 34: **End for**

step 35: **End for**

step 36: **End for**

step 37: **End for**

End

Algorithm 6.3 illustrates the process of the MO-UNetDL classifier for achieving disease prediction with superior accuracy and reduced time. The memetic-optimised U-Net deep learning classifier comprises several layers. First, k fold cross-validation is carried out to divide the dataset in to training, testing samples. The input layer gets selected features as input. In the convolution layer, weights are allocated for each input data sample to add the bias function. After that, Wilcox's index coefficient is measured to discover similarity among aspects. Based on similarity values, the soft step activation correctly distinguishes between diseased and normal data. Subsequently, the max-pooling operation is carried out to lessen the dimension of data samples. Lastly, classification results are acquired. For each classified result, cross-entropy loss is calculated depending on the actual and predicted output results. To minimise the loss of each classification, a memetic optimisation algorithm is used to tune the hyperparameters. At first, the numbers of individuals are initialized. Then the fitness is computed founded on the loss. The truncation selection is executed for selecting the finest individual out the population based on fitness. Then the exchanging of the two chromosomes is executed to create the new offspring using a two-point crossover. At last, the bit flip mutation is carried out to swap the input bit string randomly. After performing genetic operators, the fitness of that selected individual is calculated and verified the fitness. This process is continued until an optimal solution is selected. As a result, the memetic optimized U-Net deep learning classifier finds an optimal hyperparameter to minimize the error resulting in increases in the accuracy of disease prediction.

6.5 EXPERIMENTAL SETUP

The experimental assessment of the EPBC technique, TCLMCNL technique, and MO-UNetDL technique is executed in the Python language. To examine the results of three proposed classifier techniques over existing methods**,** a COVID-19 coronavirus India database and Pneumonia datasets are considered for the execution procedure. After collecting data, k-fold cross-validation is employed to separate data samples into different groups (iterations, i.e.). In this work, the 10 cross-validation is employed. Overall, the database is categorized as training and testing sets. For each iteration, split 80% of the training data, and the remaining 20% of the data samples build the testing set. The assessment process consists of three steps: designing a procedure, executing the procedure on the specific platform, and analysing the results. First, the proposed EPBC technique, TCLMCNL technique and MO-UNetDL technique are designed for disease prediction. Then, the algorithmic procedures of EPBC, TCLMCNL, and MO-UNetDL techniques are executed using the Python high-level programming language platform. The results of the three proposed classifiers were obtained after applying the preprocessing technique ZMFNE and the feature selection technique SCTPP-FS.

Table 6.1 Classification Accuracy – COVID-19 and Pneumonia Datasets

Data Samples		EPBC – Classifier (%)		TCLMCNL – Classifier (%)		MO-UNetDL – Classifier (%)	
COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset
10000	2000	97.4	97.56	98	98.46	99.3	99.5
20000	4000	97.16	97.8	97.6	97.82	98.23	98.65
30000	6000	96.94	97.89	97.53	97.76	98.11	98.46
40000	8000	96.76	98	97.33	97.14	98	98.25
50000	10000	95.83	97.58	96.71	98	97.93	98
60000	12000	95.72	97.89	96.52	97.14	97.83	97.93
70000	14000	95.93	96.78	96.43	96.58	97.62	97.82
80000	16000	95.81	96.45	96.34	96.52	97.49	97.52
90000	18000	95.46	97.59	96.05	96.69	97.31	97
100000	20000	94.82	97.65	95.82	95.25	97.22	96.89

Table 6.1 describes the classification accuracy of COVID-19 and Pneumonia datasets for EPBC, TCLMCNL and MO-UNetDL classifiers. To carry out the experiments, the number of data samples used ranges from 10000 to 100000 for COVID-19 and from 2000 to 20000 for the Pneumonia dataset. In comparison to other proposed classifiers, the MO-UNetDL classifier exhibits better accuracy.

Table 6.2 Results of Precision – COVID-19 and Pneumonia Datasets

Data Samples		EPBC – Classifier (%)		TCLMCNL – Classifier (%)		MO-UNetDL – Classifier (%)	
COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset
10000	2000	98.24	96	98.24	97	99.49	99
20000	4000	97.66	94.52	98.12	96.56	99.12	98.46
30000	6000	97.68	94.31	98.16	96.12	99.04	98.11
40000	8000	97.64	94	98.04	96	98.86	98.54
50000	10000	96.91	93.76	97.91	95.72	98.84	97.82
60000	12000	96.83	94.54	97.88	95.43	98.81	98.87
70000	14000	96.52	93.59	97.86	95	98.66	98.69
80000	16000	96.47	94	97.81	94.52	98.69	98.76
90000	18000	96.43	94.55	97.77	94	98.59	98.49
100000	20000	96.38	93.26	97.73	93.46	98.57	97.89

Table 6.2 describes the results of precision for COVID-19 and Pneumonia dataset for the EBPC, TCLMCNL and MO-UNetDL classifiers. The results shows the high precision value for MO-UNetDL classifier when compared to other proposed classifier.

Table 6.3 Results of Recall – COVID – 19 and Pneumonia Datasets

Data Samples		EPBC – Classifier (%)		TCLMCNL – Classifier (%)		MO-UNetDL – Classifier (%)	
COVID-	Pneumon	COVID-	Pneumonia	COVID-	Pneumonia	COVID-19	Pneumonia

19 Dataset	ia Dataset	19 Dataset	Dataset	19 Dataset	Dataset	Dataset	Dataset
10000	2000	99.06	95.00	99.27	97.00	99.79	99.00
20000	4000	98.00	94.86	98.86	96.52	99.32	98.82
30000	6000	97.76	94.63	98.81	96.47	99.28	98.41
40000	8000	97.56	94.34	98.77	96.00	99.26	98.00
50000	10000	97.44	93.41	98.71	95.75	99.25	97.82
60000	12000	97.31	93.00	98.67	95.48	99.23	97.54
70000	14000	97.28	92.71	98.62	95.23	99.19	97.12
80000	16000	97.25	92.51	97.86	95.00	98.69	97.00
90000	18000	97.23	92.00	97.84	94.82	98.76	96.86
100000	20000	97.19	91.64	97.81	94.64	98.77	96.00

Table 6.3 depicts the recall oucome for the Proposed classifiers, where the MO-UNet classifier outperformed well. The following Table 6.4 presents the specificity results of the three introduced classifiers. The MO-UNet classifier produced better results when compared to the EBPC and TCLMCNL classifiers.

6.4 Results of Specificity – COVID – 19 and Pneumonia Datasets

Data Samples		EPBC – Classifier (%)		TCLMCNL – Classifier (%)		MO-UNetDL – Classifier (%)	
COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset	COVID-19 Dataset	Pneumonia Dataset
10000	2000	64.19	69.00	68.18	72.00	72.72	75.00
20000	4000	55.25	67.00	59.25	70.63	62.63	73.25
30000	6000	48.41	66.52	50.34	68.62	54.61	71.36
40000	8000	49.36	64.00	51.25	67.00	55.67	69.63
50000	10000	48.14	63.21	50.62	65.52	54.68	68.25
60000	12000	47.64	60.00	49.36	63.00	53.91	65.41
70000	14000	47.25	59.64	49.11	61.36	52.21	64.21
80000	16000	49.25	56.00	51.35	59.24	54.69	63.14

90000	18000	48.55	55.63	50.31	58.31	53.31	62.00
100000	20000	47.25	54.00	49.22	57.14	51.26	60.34

The following Table 6.5 shows the results for the prediction time of the COVID -19 and Pneumonia datasets. From the results MO-UNet classifier uses minimal amount of time in the classification than the other proposed classifiers.

Table 6.5 Results of Prediction Time – COVID-19 and Pneumonia Datasets

Data Samples		**EPBC – Classifier (ms)**		**TCLMCNL – Classifier (ms)**		**MO-UNetDL – Classifier (ms)**	
COVID-19 Dataset	**Pneumonia Dataset**	**COVID-19 Dataset**	**Pneumonia Dataset**	**COVID-19 Dataset**	**Pneumonia Dataset**	**COVID-19 Dataset**	**Pneumonia Dataset**
10000	2000	2900	1200	2500	900	2000	800
20000	4000	3700	1400	3300	1000	2700	900
30000	6000	4400	1600	3900	1300	3400	1100
40000	8000	4900	1800	4300	1600	3900	1400
50000	10000	5300	2000	5100	1800	4600	1600
60000	12000	6400	2200	5900	1900	5500	1700
70000	14000	6800	2400	6300	2100	5800	1800
80000	16000	7000	2600	6500	2300	6100	2000
90000	18000	7300	2800	6900	2500	6400	2200
100000	20000	7700	3000	7400	2700	7000	2400

6.6 CHAPTER SUMMARY

Efficient classification techniques such as EPBC, TCLMCNL and MO-UNetDL techniques are proposed to categorise the patient data samples in support of disease prediction. Initially, the EPBC technique is proposed for accurately classifying information through superior accuracy and minimum time. In EPBC, the weak classifier is dependent on the perceptron function, which is modeled for classifying the data into diseased or non-diseased. After that, the

Emphasis boosting is created through the mean weighted sum, which partitions features through lower training error. With this, disease prediction is performed with lower time and higher accuracy. Second, TCLMCNL technique is proposed for accurate classification of patient data. During the classification process, time-dependent Cox regression is applied using Cramér's phi correlation to categorize the testing disease data and training data with selected features. Levenberg–Marquardt method is applied to decrease the error rate of classification process. It helps to improve disease prediction accuracy. Lastly, the MO-UNetDL technique is proposed for analysing similarity among training and testing disease information, and it provides prediction output with lower loss. The experimental evaluation is conducted using the EPBC, TCLMCNL, and MO-UNetDL techniques, along with other conventional techniques. With the increasing number of data samples, the proposed classification models yield better performance. It means that the proposed methods address the overfitting issues and provide accurate prediction results during sample data classification. This can be done by taking preprocessed data with selected features as input for disease prediction. By processing selected data, the overfitting and underfitting problems are handled in EPBC, TCLMCNL and MO-UNetDL techniques. Examined outcomes denote which result of the MO-UNetDL technique rise prediction accuracy, precision, recall, specificity, and lesser prediction time when compared to other proposed techniques. The next chapter presents an ablation study conducted on the proposed machine learning and deep learning models. This study systematically removes certain parts of specific components to evaluate their individual contributions to overall model performance.