Chapter 4

Regularized Deep Neural Network with hybrid approach of Independent Component Analysis

---

Deep learning is a sub-field of machine learning, which is an important step on the path for creating artificial intelligence. There are complex non-linear functions that are beyond the capabilities of architectural representation. ANN contains very few hidden layers and consequently very few nonlinear transformations which are not capable of representing the complex non-linear functions. Which is the most significant disadvantage of ANN. As a direct consequence of this, "Deep Neural networks," so-called due to the complexity of its internal structure, have seen a rise in their level of utilization. This chapter includes the predictive models constructed using ANN and DNN architectures and trained with a variety of optimization algorithms for breast cancer classification. The model's validity is established by computing its predicted results using different performance measures. Overview of ANN and DNN is covered in Section 4.1. The DNN learning process and optimization strategies are explained in Section 4.2 and Section 4.3. The method for minimizing features using Independent Component Analysis is described in Section 4.4. Section 4.5 discusses the experiments and their outcomes for different cases of proposed Regularized Deep Neural Network (R-DNN) and standard DNN, while Section 4.6 includes a conclusion.

4.1 Introduction

As neural networks were revived in the 2000s, Deep learning emerged as a new research frontier that paved the way for the development of contemporary machine learning. Before this algorithm was known as an ANN. However, DNN encompasses many more domain of interconnected device then ANN. DNN is sub part of AI that enables computer systems to enhance automatically via the use of experience and new or existing data.

Neurons in the human brain are connected into complex networks. The goal of ANN is to establish a simulation of these networks and program computers to behave like interconnected brain cells. This will allow the computers to learn and make judgments in a manner that is more similar to how humans do so. The human brain is organized hierarchically or in layers, such that distinct areas of the brain are responsible for processing different types of information. In this manner, as information is received by the brain, each level of neurons analyses the information gains insight from the processing and then passes the knowledge to the next layer and gives the conclusion. The foundation behind ANNs is to mathematically simulate how the human brain processes information. Perceptron are the building blocks of ANNs which are designed to mimic the organization or architecture of the human brain. Classical Neural networks have the drawback of being overly simplistic as they are unable to accurately represent certain complex non-linear function due to shallow architecture since then, there has been an effort to develop neural network with deep architecture.

Deep Neural Network (DNN) are exactly the same as classical Neural Network apart from that they have more hidden layers, which makes them "Deep". Deep learning is a form of computer learning in which, unlike traditional methods of ML, which include humans instructing computers how to process and learn from data. Deep neural Network is widely used in medical field to diagnose diseases. ANN is a mathematically developed computational network which is modelled after biological neural networks that from the structure of the human brain. In the same way that neurons in a human brain are connected to each other, neurons in ANN are also inter connected to each other. These neurons are called nodes. It produces an output pattern for a given input pattern. It is a study of a network consisting of nodes connected by changeable weights. Neurons perform as summing and non-linear mapping junctions. The nodes store experiment at knowledge through a process of learning. The nodes of the brain are changeable. They gain knowledge by change

in weights.

A human brain consists of approximately 100 billion of processing units which is called neuron. They are communicated through a connection network of axon and synapses. They have approximate 1014 density synapses per neuron. Neurons are the building block of our brains. Dendrites of neighboring neurons transmit input signal to a neuron. These signals are first processed in the cell body of the neuron and then they are transfer via the axon to another neuron which is known as output terminal. In ANN, Dendrites represent as input cell nucleus represent as Nodes, Synapse represent as weights and Axon represents as output. Relationship or terminology of biological neural network and ANN is as shown below in table 4.1.

Table 4.1: Associated terminology of BNN and ANN

Biological Neural Network (BNN)	Artificial Neural Network (ANN)
Dendrites	Inputs
Cell Nucleus	Nodes
Synapse	Weights
Axon	Output

In the late 1800s, scientists began studying the functioning of the human brain, potentially establishing the beginning of the neural network. In 1890, William James published the first book of brain activity patterns. In 1943, Warren McCulloch and Walter Pitts introduced the first mathematical model of single idealized biological neuron which is known as McCulloch – Pitts neuron model which is shown in fig. 4.1. This model is quite simple and divided into two parts because it does not required learning. In 1958, Rosenblatt introduced first learning algorithm to classify the linearly separable data known as perceptron. It is a single layer network consists of only input and output layer. But after 1986 ANN become more popular because having multi-layer network which contain input layer, output layer and one more hidden layer with back propagation algorithm. Back propagation algorithm was discovered by Rumelhart, Hinton and Williams in 1986. ANN Models mimic this process by considering input signals as input variables which stores the information about the pattern. To mimic the behaviors of dendrites, these input variables are weighted based on their relative relevance. Comparison of BNN and ANN is depicted in fig. 4.2 Different activation functions like Sigmoid, Tanh, ReLU, etc. are applied to these weighted signals in hidden layer [102].

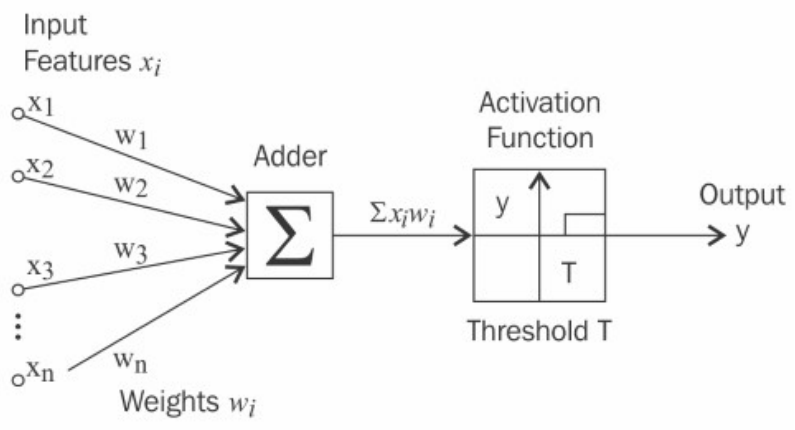

Figure 4.1: Architecture of McCulloch Pitts

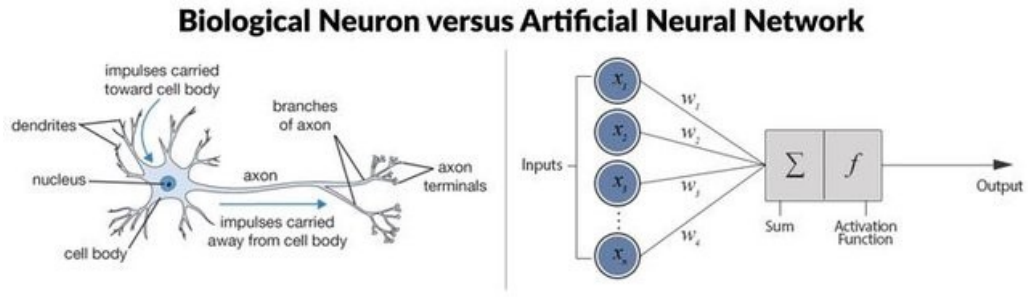

Figure 4.2: Biological Neuron versus Artificial Neural Network [85]

I. Maglogians et. Al. compared ANN and Bayesian classifiers with SVM for the diagnosis of breast cancer [74]. NN with L-BFGS optimization algorithm on breast cancer data set, SPECT heart problem , Australian Credit approval problem least problem and Escherichia coil problem were discussed by M.S. Apostolopoulou et. al. [11]. Q.V. Le et. al. used deep learning for MNIST data set using SGD and L-BFGS optimization algorithms with line search [68]. G.I.Salama et. al. employed multi classifier for diagnosis of breast cancer. Where they used MLP, Naive Bayes, SMU and Instance based for KNN [105]. R. Fakoor et al. studied about how deep learning can be used to improve cancer detection by implementing unsupervised feature learning [102]. G. Zoriluoglu and M. Agagla achieved 94.44% and 96.77% classification accuracy for ensemble approach and respectively [42]. A. Mart et. al. investigated about feature reduction technique ICA with KNN, ANN, and SVM [77].

A. Esteva et. Al used DNN for diagnosis of skin cancer and achieved 72.1% classification accuracy [36]. Z. Han et. Al used deep learning in classification of breast cancer and obtained 93.2% accuracy [44]. A.F. Agarap attained 99.04% accuracy using ML algorithms [6]. Authors employes SVM, ANN, KNN, DT and NB with feature reduction technique in classification of breast cancer and obtained with different

optimization algorithms near by 98% of accuracy [32] [130] [89]. 17) M. Tiwari et. Al. proposed various modals such as ANN, SVM, MLP etc. in diagnosis of breast cancer and obtained 97.3% classification accuracy [121]. S. Aalaei et. Al. employed ANN with GA. Based algorithm for feature selection and used PSO algorithm for optimization in diagnosis of 3 different breast cancer dataset namely WBC, WBDC and WPBC [1]. A.M. Abdel-Zaher and A.M. Eldeib employed computer aided based Deep Belif network with recursive feature elimination method for classification of Breast Cancer [3]. L. Abdel-Ilah et. al. used WBC dataset in classification of breast cancer using ANN and achieved 98% accuracy [2]. H. Tike Thesin , K, Kaushik and C, Prasetgo, utomo et. Al. employed ANN, in identification of breast cancer for WBC, WBDC dataset and achevieed 98% of classification accuracy [120] [59] [97]. S. Karthik et. al. proposed DNN with REF feature selection technique and evaluated on WBC dataset andobtained 98.62% accuracy [58]. 10-11) ANN Cascade – forward NN, RF, KNN, NB, SVM were employed in diagnosis of breast cancer for WBC and WDBC dataset by Ms. M. Gayatheri, B. Sahi et. al. [104] [72]. D.A. omondiagbe et. Al. diagnosed breast cancer by employing SVM, ANN and NB using Linear Discriminant Analysis which is feature reduction technique [89]. SVM and ANN employed with correlate feature selection technique by R. Aiyami et. al. and achieved 97.14% and 96.71% classification accuracy [9]. I.Maglogiannis et. al and M.A. Gokhan Zorluogul employs different classifier in diagnosis of breast cancer [74] [42].

In this chapter Regularized Deep Neural Network (R-DNN) is proposed in classification of breast cancer data. Proposed R-DNN is compared with ANNs and results are obtained. Also Independent Component Analysis (ICA) is employed for feature reduction in proposed R-DNN model, which is validated on WBC and WDBC data sets. Optimization techniques and ICA are explained in depth in the next section.

4.2 Learning using DNN and ANN

ANN is consisting of single hidden layer and then it is extended to multi hidden layers which is known as Deep Neural Network with advanced optimization techniques. It has 3 layers namely input layer, hidden layer, output layer. Each layers consists of nodes. Each neuron in neural network is connected to at least one other neuron. Moreover, each connection in neural network is associated with a weight. These weights are multiplied by the input value which receives from the previous layer. It determines the significance of this relationship. Each neuron possesses an activation

function which determines the neurons output. The activation function is utilized to introduce non-linearity into the network's modelling capabilities. Let network has m training samples where each sample consist of n inputs say $X = \{x_1, x_2, ..., x_n\}; X \in \Re^n$. The layer which receives these input signal is known as hidden layer. Let $Z = \{z_1, z_2, ..., z_j\}$ be the outputs of the J neurons in the hidden layer [134]. Then this Z is an input for the output layer. Let Y be the L- dimensional $Y = \{y_1, y_2, ..., y_l\}$ which is output vector corresponding to n dimensional input vector. Each neuron in the hidden layer receives the n inputs through weighted links. Each neuron in the output layer receives the J input from the hidden layer as input through weighted links. When the network is fully trained, the output vector Y should be identical, i.e. it should be very close to desired output vector d associated with the input vector X. Deep Neural Network is divides into two phase : i) forward propagation ii) Backward propagation. Training algorithm is applied to the multi-layer feed-forward networks. Once the loss for the trained network is calculated, this error propagated backward. Consider each neuron in the hidden layer uses an activation function f_h and each neuron in the output layer uses an activation function f_o. Choice of activation function depends on the application and range of output values. Fig. 4.3 shows the difference between Simple Neural Network and Deep Neural Network.

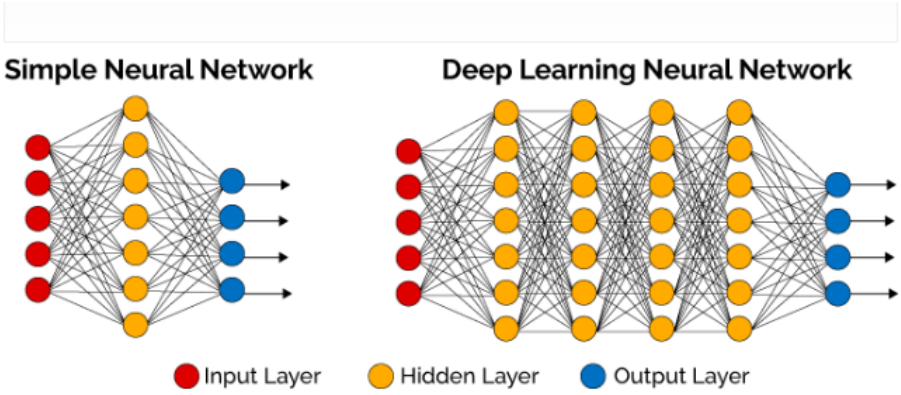

Figure 4.3: Shallow Artificial Neural Network vs Deep Neural Network [124]

4.2.1 Methodology

Let w_{ji} be the weights of the link connecting the j^{th} hidden neuron z_j to the i^{th} input x_i. Also let w_{lj} be the weights connecting the l^{th} output neuron y_l to the j^{th} hidden neuron z_j. After passing the input data through the network with all internal calculations in forward phase, the final layer will calculate the result of the input samples and give whether the bad or good prediction. Then calculate loss using loss function to estimate the loss and compare network's predicted output with

original output. Once the loss is calculated to reduce the error, loss is propagated backwards. To reduce the error, weights of inter connections of the neurons will be updated using weight optimization technique in back propagation phase. It starts from output layer to hidden layer where loss information propagates to all neuron which takes part directly to calculate output of the network. Then further it is propagated between hidden to input layer. This procedure is repeated, one layer at a time, until all of the neurons in the network have received a loss signal which represents their mutual contribution to overall loss. The main aim is to minimize the error between network output and desired output by optimizing weights i.e. by finding weight which can reduce the error. Network's error is given by eq. 4.1.

$$E = \sum_{j=1}^{l} e_{ij}^2 = \frac{1}{2} \sum_{j=1}^{l} \left(y_{ij} - y_{ij}^*\right)^2; i = 1, 2, .., m \tag{4.1}$$

Procedure of the forward and backward phase is as follows:

1. Consider random weights $\{w_{j1}, w_{j2}, ..., w_{ji}\}, j = 1, 2, ..., J$, from each input node to j^{th} hidden neuron. Where, n is the number of input nodes and j is the number of neurons at the hidden layers. The input of the j^{th} hidden node is calculated as follows:

$$net_j = \sum_{i=1}^{n} w_{ji} x_i; j = 1, 2, ..., J$$

2. The output of j^{th} hidden node is given by $z_j = f_h\left(\sum_{i=1}^{n} w_{ji} x_i\right); j = 1, 2, ..., J$, where, f_h is activation function used at the hidden layer.

3. Network output is calculated as: $y^* = f_o\left(\sum_{j=1}^{J} w_{lj} z_j\right); l = 1, 2, ...L$, Where f_o is the activation function used at the output layer and z_j is the output of the j^{th} hidden node.

4. After calculating output of the each node at the output layer in forward propagation phase, calculate network's total output. Compare network's output with original output and calculate the error using eq.4.1.

5. Initialize the tolerance. If the network error E is more than tolerance then using back-propagation update the weights of the network to minimize the error E.

6. We have used different optimization algorithm like Stochastic Gradient Descent (SGD), L-BFGS and Adam to minimize the network's error E by updating weights. Also we have used different activation functions like logistic, tanh and ReLU.

4.3 Optimization algorithm

An optimization algorithm is a process that is carried out in an iterative manner by comparing various potential solutions by changing the hyper-parameters until the optimal solution is determined, to create an accurate model with less error. Any DNN model employs an optimization algorithm to generalize the data in order to predict new data. During the training of DNN model, optimization algorithms optimize each epoch's weights or parameters and also it minimize or maximize the cost/loss function to reduce the error whom mapping inputs to outputs. Such optimizers have a significant impact on the precision of the DNN model. The model's speed and learning capabilities are also impacted by these factors. By adjusting parameters like weights and learning rate, an optimizer can improve the performance of DNN. As a result, the overall loss is reduced and efficiency of DNN is enhanced.

Before, proceeding with various optimization algorithms following few terms are important to familiar with.

1. Batch: Indicates the group of samples that should be taken in order to update the model's parameter.

2. Epoch: Interactions over the entire training data set that the algorithm is performed.

3. Sample :- A single row of data set.

4. Learning Rate:- It is a step size or α which is used to find the minimum of cost function. Consider small values such as 0.1, 0.001 or 0.0001. Usually, if a small value which is examined and modified according to how the cost function is behaving. With small learning rate, any optimization algorithms converge to its optimal solution or it may hit the maximum number of iterations. The algorithm may diverge or fail to converge if the learning rate is two high.

5. Cost/Loss function:- It is a function that evaluate how well a model works with the data that has been provided. It used to evaluate the error between

actual output and network's predicted output. This enhances the model's performance by giving it the information if needs to fine-tune its parameters in order to reduce error and determine the local or global minimum. A cost function is minimized by repeatedly iterating in the negative direction of gradient until the function's value approaches or equals to zero. The learning process will come to an end for the model once find optimum solution.

6. Weights:- The parameters of a model that can be learned, which regulate the signal that is sent between two neurons.

Various optimization algorithms are describe as follows.

4.3.1 Stochastic Gradient Descent

Gradient Descent (GD) is a first order iterative optimization algorithm which is very popular in DNN for finding minima or maxima of a differentiable and convex function. It is used to find optimized parameters of the learning model by minimizing cost or loss function to approach the desired output. There are three types of GD learning algorithms namely, i) Batch GD ii) Stochastic GD iii) Mini-Batch GD. In this study, Stochastic Gradient Descent (SGD) algorithm is used to optimize the weights of the network.

Minimizing a given cost / loss function is the objective of the GD algorithm. For this purpose, it repeatedly carry out the following two steps:

1. Determine the gradient of the cost function at the given point.

2. Take a step in the direction which is opposite to the gradient direction by multiplying the gradient α times from the current point.

Iterative calculating the next point based on the gradient at the current position, scaling it by learning rate and then subtracting the resultant value from the current position is precisely what GD algorithm does. The value is subtracted to minimize the cos function. If the function is maximized then add the value [88].

Stochastic Gradient Descent

SGD is one of variant of GD which proceed with one training sample from data set for each iterations. As a consequence, the parameters continue to be modified even

after one iteration has passed and only one sample has been worked through. So, SGD is a lot quicker than batch GD. However, even when there are many training samples, the system only process one sample at a time. For SGD, algorithm 1 is written as follows:

Algorithm 1 An algorithm for SGD

Require: Training samples: $\{x_i, y_i\}_{i=1}^{n}$; where n is no. of samples
Require: Initialize weight Parameter: w
Ensure: Set as the limit of convergence as: ϵ
Ensure: Learning rate: α
  for $i = 1$ to n **do**
    for $t = 1$ to m **do**
  Compute network prediction:
      $y_i^* \leftarrow h(x_i)$
  Compute total Loss/Error function J**:**
      $L(w) \leftarrow \frac{2}{n} \sum_{i=1}^{n} (y_i - y_i^*)^2$
      while $|w_{j+1} - w_j| > \epsilon$ **do**
        $w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w}$;
      end while
    end for
  end for

4.3.2 Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) optimizer is improved algorithm of SGD which was introduced by Diederik Kingma and Jimmy Ba in 2015 in the paper entitle "Adam: A method for stochastic optimization" [61]. It is a first and second order gradient based optimization algorithm which is used to update network's weights for training data iteratively [61].

Adam combines the best characteristics of AdaGrad (Adaptive Gradient) and RMSProp (Root Mean Square Propagation) algorithms. In SGD throughout the training process, the learning rate α remains constant and applied to all weight updates. In Adam, each weight/parameter in the network is assigned a learning rate which is updated individually during training process. The algorithm evaluates the first and second Adapt moments of the gradients to determine individual learning rates for each weight/parameter of the network, that's why it is called Adaptive. Like RMSProp, it scales the learning rate based on the squared gradients. But like AdaGrad, it also takes advantages of momentum by calculating the moving average of the gradient, rather than gradient itself. Here, first moment means un-centered variance (i.e. without subtracting mean during variance calculation). By modifying the GD rate, Adam ensures the least amount of oscillation at the global minimum, while taking large step size to overcome the local minima. Above two strategies can be combined to find the global minimum effectively. Adam uses the average of the second moments of the gradients in addition to the average of the first moments to determine the learning rates for the parameters. Mathematically, it is define in eq. 4.2 and eq. 4.3:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.2}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4.3}$$

Here, g is the gradient of the cost function at time step t. m_t is the exponential average of the gradient along W and v_t is the exponential squared average of the gradient along W. The bias correction of moving averages is calculated using eq. 4.4.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{4.4}$$

Weights or parameters are updated based on the calculated moving averages with learning rate α using following eq. 4.5

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t \tag{4.5}$$

Where, $\hat{m}_t$ and $\hat{v}_t$ are moving averages and η is learning rate.

Adam relies on hyper-parameter β_1 and β_2 values, where β_1 is the exponential decay rate which is 0.9 for the first moment estimation and β_2 is the exponential decay rate which is 0.999 for second moment estimation. The vectors of moving averages are initialized with zero at the first iteration. Adam algorithm is descibe as follows in algorithm 2:

Algorithm 2 An algorithm for Adam

Require: Learning rate: α
Require: Exponential decay rates for the moment estimates: $\beta_1, \beta_2 \in [0, 1)$
Ensure: Set as the limit of convergence as: ϵ
Require: Objective function with parameter w: $f(w)$
Require: Initial weights vector: w_0
  $m_0 \leftarrow 0$ ▷ Initialize 1^{st} moment vector
  $v_0 \leftarrow 0$ ▷ Initialize 2^{nd} moment vector
  while $|w_{t+1} - w_t| > \epsilon$ **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_w f_t(w_{t-1})$ ▷ Find gradient w.r.t. objective function at time step t
    $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ ▷ Update biased first moment estimate
    $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ ▷ Update biased second row moment estimate
    $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ ▷ Compute bias-corrected first moment estimate
    $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$ ▷ Compute bias-corrected second row moment estimate
    $w_{t+1} \leftarrow w_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$ ▷ Update parameters
  end while
  Return: w_t ▷ Resulting parameters

4.3.3 L-BFGS

Finding the minimum or maximum value of any objective/cost/loss function is the goal of optimization problems. First-order derivative methods like GD and steep-

est descent and second order derivative methods like newton's method are both deterministic ways to deal with optimization problems. In order to determine the function's maximum and minimum values, the first order derivative method depends on the first derivative which is known as the gradient, in either a downward or an upward direction for optimal solution. In order to estimate more accurately the minima of the objective function, second-order derivative method uses the derivative of derivative (i.e. Hessian matrix - a matrix holding the second order derivatives) have been employed [40].

L-BFGS stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno. The L-BFGS algorithm is non-linear and iterative algorithm. It is extension of BFGS algorithm, which is second-order optimization algorithm that falls within the category of Quasi-Newton method. As a second order derivative, the Hessian matrix is utilized in Newton's approach. The size of the Hessian and its inverse depends on the number of parameters of the loss function. It can be challenging to manage the size of the hessian if problem is really Vast. The L-BFGS is able to overcome this problem by making the assumption that the preceding/last iterations inverse of the Hessian can be simplified. In contrast to BFGS, which takes into account the entire history of the gradient, L-BFGS only takes into account the most $'n'$ recent gradients. L-BFGS is desribe as in algorithm 3 [99].

Algorithm 3 Algorithm for L-BFGS [99]

Require: Training samples: $\{x_i, y_i\}_{i=1}^{n}$
Ensure: Tolerance of function value from previous iteration: ϵ_1
Ensure: Tolerance on gradient value: ϵ_2
Require: Initialize $n \times n$ positive definite symmetric matrix $[B]$ as the identity matrix

Step 1: Compute gradient vector $\nabla f_1 = \nabla f(x_1)$ and set the iteration no. as $i = 1$.

Step 2: Compute $f(x_i)$ and gradient of function $\nabla f(x_i)$ at point x_i and set $S_i = -[B_i]\nabla f_i$ (Search direction).

Step 3: Find the optimal step α_i^* in the direction S_i and update $x_{i+1} = x_i + \alpha_i^* S_i$.

Step 4: For optimality, test the point x_{i+1}. If $||\nabla f_{i+1}|| \leq \epsilon_2$ or $|f_{i+2} - f_{i+1}| \leq \epsilon_2$. Where, ϵ_1 and ϵ_2 is a small quantity. Take $x^* = x_{i+1}$ and stop the process otherwise go to step 5.

Step 5: Update the Hessian matrix as,

$$\big[B_{i+1}\big] = \big[B_i\big] + \left(1 + \frac{g_i^T\big[B_i\big]g_i}{d_i^T g_i}\right)\frac{d_i d_i^T}{d_i^T g_i} - \frac{d_i g_i^T\big[B\big]}{d_i^T g_i} - \frac{\big[B_i\big]g_i d_i^T}{d_i^T g_i}$$

Where, $d_i = x_{i+1} - x_i = {\alpha_i}^*$ and $g_i = \nabla f_{i+1} - \nabla f_i$

Step 6: Set the new iteration as $i = i + 1$ and repeat the procedure from step 2.

4.3.4 Activation functions

It is used to determine whether neuron should be activated or not. It is also known as Transfer function. Activation function regulates the results of neural network in various ways. These activation function can be linear or non-linear. A neural network without an activation function is just a linear regression model. The activation function applies a nonlinear transformation to the input, enabling the system to learn and execute more complex tasks. In our study, we used following activation functions.

1. Sigmoid (Logistic) function : $f(x) = \frac{1}{1+e^{-x}}$
2. Tanh: $f(x) = \frac{2}{1+e^{-2x}} - 1$
3. ReLU:$f(x) = max(0, x)$

4.4 Independent Component Analysis

J. Herault, C. Jutten and B. Aur., introduced an iterative real-time Independent Component Analysis (ICA) technique in their work in 1980s [119]. The objective of ICA is to derive meaningful information from the data. This information can be images, data or audio files. As a result, ICA is utilized for the purpose of extracting source signals in a wide variety of applications, including but not limited to medical signals [129] [25], biological assays If ICA can eliminate or keep only one source, it is considered a dimensionality reduction algorithm. ICA is often seen as a more advanced form of PCA [118] [29]. While ICA optimize higher order statistics like kurtosis, PCA maximize the data's covariance matrix which represents second-order statics. As a result, PCA identifies uncorrelated components while ICA finds independent component [50][118]. This means that ICA can be used to extract independent sources from mixture data when the higher-order correlations are minimal (or) negligible or insignificant [50].

The fundamental structure of ICA is as follows. Assume that, observed signal is a linear combination of two independent sources. The observed signal can be expressed as: $x = As$; where, s is source signal, A is unknown mixing matrix consists of constant elements or Identity matrix and x is observed values. Using ICA, unknown mixing matrix A is calculated. The separating matrix W is calculated by $W = A^{-1}$. Aim is to find mixing matrix A and original source signal s. The mixture of sources s can be retrieved by multiplying the observed signal x with the inverse of mixing matrix $W = A^{-1}$. i.e. The original signal can be found using: $\hat{s} = Wx$ [78] [118] [50]. Mixing and unmixing signals, centering and whiting of the data is the first step in computing the ICs which is shown in the following figures 4.4 and 4.5.

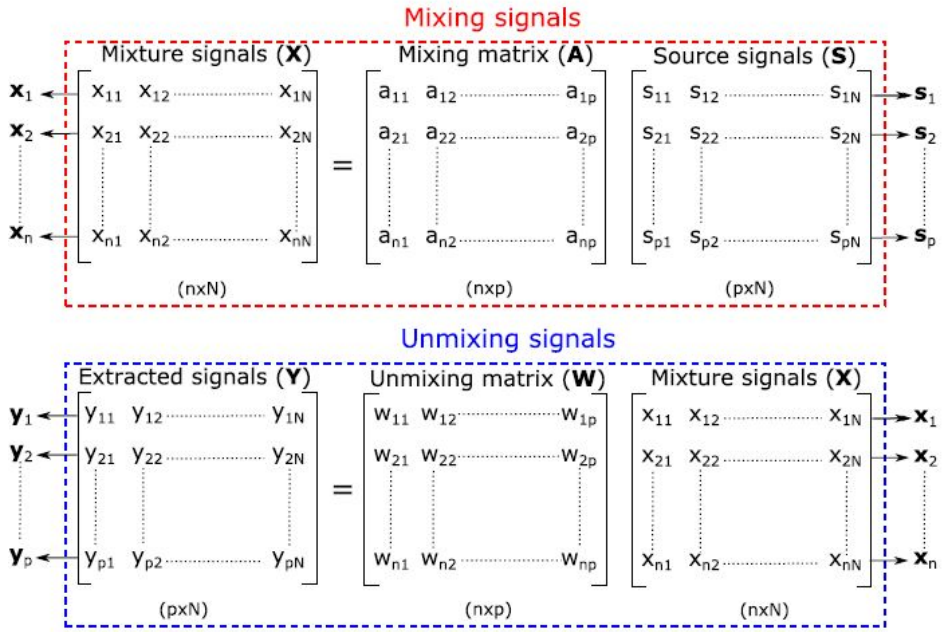

Figure 4.4: Block diagram of ICA mixing steps
[119]

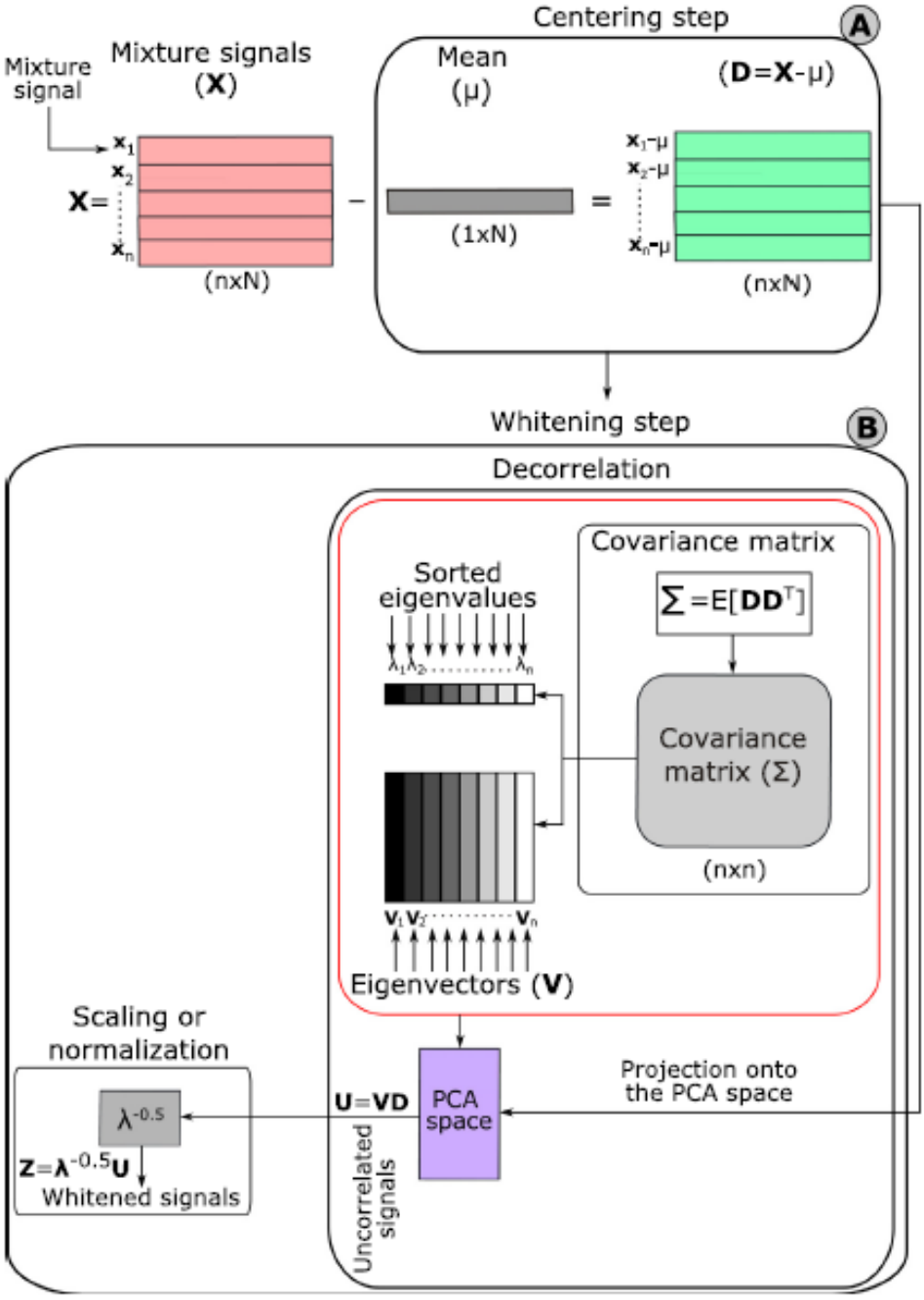

Figure 4.5: Visualization for pre-processing steps in ICA [119]

4.5 Experiments and results

R-DNN with ICA is the foundation of the suggested model. The proposed model incorporates four major steps for training R-DNN, which are depicted in fig. 4.6. There are five parts in this process: data collection, data pre-processing, feature reduction, separating the data into train and test sets and finally evaluating the model's performance by defined optimization techniques and stated activation functions. After loading the WBC and WDBC data set into the network, we normalize data set using a pre-processing technique to ensure that no attributes were missing or that the data was consistent. The following eq. 4.6 is used to normalize the data.

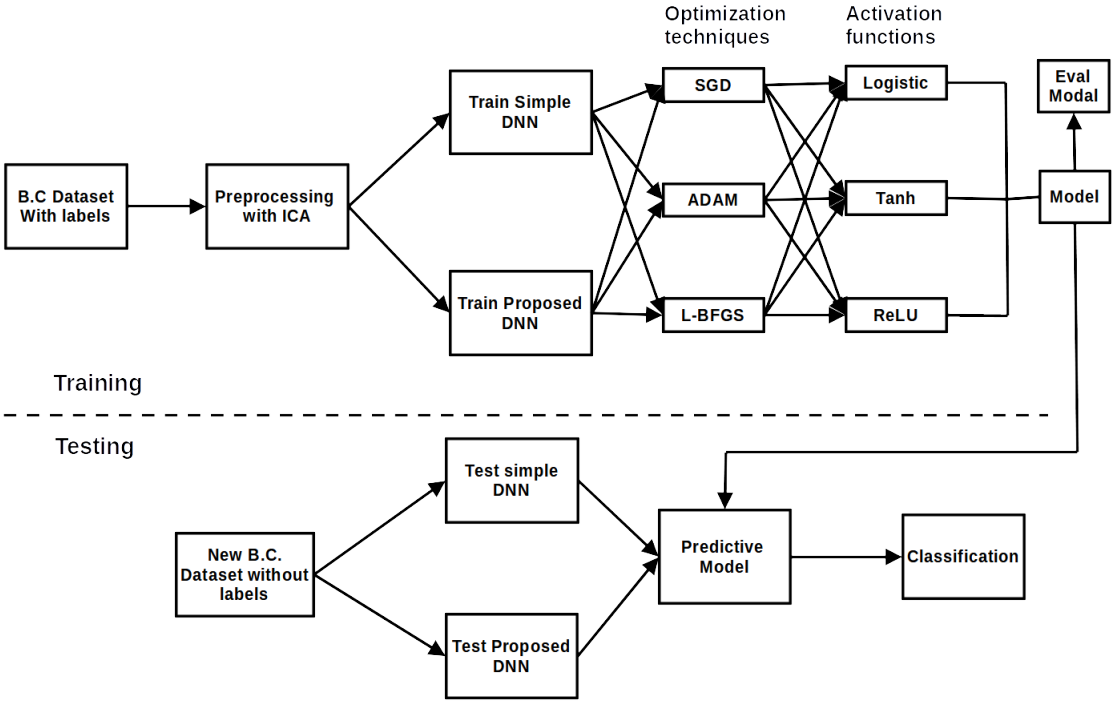

Figure 4.6: Flowchart of proposed R-DNN with ICA

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.6}$$

In the third stage, we employed a feature reduction strategy to decrease multicollinearity from the model and increase its overall performance and accuracy. We utilize ICA to minimize the number of features in the WBC data set. The number of features was decreased to three with no loss of generality and all of the original information from the data was kept. Following that, a 10-fold CV was applied to the data to separate it into a train-test set. With this technique, data sets are divided

into 10-folds (i.e. groups), with each group serving as both a training and testing set to determine the model's overall effectiveness in general. Working steps of proposed R-DNN is as follows:

1. Input data set i.e. feature vector of WBC and WDBC i.e. inputs $X = \{x_1, x_2, ..., x_9\}$ are pass through in the first layer. Then the data are sent to hidden layers with weights assigned. There can be as many as hidden layers.

2. All calculation is done in the hidden layer after the inputs have been passed on. i.e. all input vectors are multiplied by weight vector W and added to the bias b: $y = WX + b$. Weights are randomly initialized in forward propagation.

3. The activation function is then applied to linear equation y in the step ii. as shown in eq. 4.7. The activation function is a nonlinear transformation applied to the input before it is sent to the next layer of neurons. The activation function's significance is that it introduces non-linearity into the model.

$$y == \varphi\Big(W^T X + b\Big) \tag{4.7}$$

   Where, W is weight vectors, X is the input vectors, b is the bias and φ is the non-linear activation function.

4. Each hidden layer goes through the entire process stated in step iii. After we have passed through all of the hidden layers, we go to the final layer, which is our output layer, which provides us with the final output.

5. The error is calculated after receiving the output layer's predictions, which is the difference between the actual and predicted output. If the error is large, actions are taken to minimize the error using Back-propagation. The error or loss is calculated using log loss error function as shown in eq. 4.8:

$$loss = -\frac{1}{N}\sum_{j=1}^{n} y_{ij} ln(p(y_{ij})) + (1 - y_{ij}) ln(1 - p(y_{ij})) \tag{4.8}$$

$$; i = 1, 2, ..., m$$

   Where, y_{ij} is the actual class and $ln(p(y_{ij}))$ is probability of actual class.

6. The weights are updated with different optimizers methods using equation from algorithm 1, eq. 4.5.

The suggested DNN was trained and evaluated using the optimizers like SGD, Adam, and L-BFGS algorithms, which used activation functions such as Logistic, Tanh, and ReLU to train and test the network. We used the log-loss function, also known as the binary cross-entropy function, to calculate the difference in error between the network output and the desired output and to compare the two results.

We include an additional $L2$ Regularization term in the log-loss error function as shown in eq. 4.9 to make it more robust. To reduce error by fitting a function adequately on the provided training set and avoiding overfitting, the regularization term is employed. This regularization technique helps to reduce variance in our model by penalizing for complexity. By adding $L2$ regularization to our model, we're effectively giving over some of our model's capacity to fit the training data well in exchange for the ability to generalize the model to data which hasn't been seen before i.e. on test data. Large weights are penalized by adding $L2$ regularization into log loss error function.

$$loss + \sum_{i=1}^{n} \|W_{ji}\|^2 \frac{\lambda}{2m} \tag{4.9}$$

Where, n be the number of layers, w_j be the weight matrix for the j_{th} layer, m is the number of input and λ regularization parameter.

We evaluated our proposed model for various λ values such as 0.1,0.01,0.001,0.0001 and 0.00001 and achieved 100% classification accuracy for $\lambda = 0.001$.

Following the training of the Deep Neural Network, test data is collected in order to evaluate the model's performance in the classification of breast cancer as benign or malignant, respectively. Model performance and efficiency are determined by the following parameters: accuracy, sensitivity, specificity, precision, and F-score from the confusion matrix.

In order to determine whether a cancer model is capable of appropriately classifying cancer as malignant or benign, the Receiver Operating characteristic (ROC) curve is utilized [18]. A True Positive Rate (TPR) is calculated by comparing the rate of correctly classified cases (True Positive Rate) against the rate of wrongly classified instances (False Positive Rate). The True Positive Rate (TPR) is between 0 and 1. A different trade-off between a correctly diagnosed tumor being classified as benign or malignant is represented by each dot on the curve.

4.5.1 Experiments and results of WBC data set

A data set of WBC has been used as an input in this experiment, with the desired result being either benign or malignant. Nine input neurons, two hidden layers, and one output neuron are used to construct the model. It is necessary to increase or decrease the number of neurons in hidden layers in order to attain the best accuracy. It is trained with two hidden layers, each of which has seven pairs of hidden neurons, such as 100-0, 100-100, 250-100, 250-250, 500-100, 500-500, and 1000-1000, respectively. We used learning rates of 0.01, 0.001, and 0.0001 to train our network in order to accelerate the convergence of our model. DNN without ICA and 10-fold CV were used in the tests, as were DNN with ICA and 10-fold CV in the case of DNN with ICA.

Case 1: Results for DNN without ICA and 10-fold CV (Simple DNN model)

In this scenario, the data set is divided into two parts: a train set and a test set. In this case, 80% of data are taken as training set and 20% of data are taken as testing test. There are no approaches utilized in this case because DNN is a simple model.

The log loss error is used to assess performance. It depicts error vs. epoch for the training data set, as seen in fig. 4.7, fig. 4.8 and fig. 4.9. Although the error decreases with additional training epochs, the error on the validation data set may begin to climb when the network becomes over-fitted to the training data as a result of overfitting. For e.g., in the default configuration, the training is terminated after six consecutive increases in validation error, and the highest performance is obtained from the epoch with the lowest validation error. In this particular instance, the error is assessed in the cross-entropy function.

The fig. 4.7, fig. 4.8 and fig. 4.9 depicts the varied learning rates for the training error. After training, the version of the network that performed the best on the validation set was used.

On the basis of train-test data, Fig. 4.10 depicts the confusion matrix for DNN without ICA and without a 10-fold CV for training data. When using a trained network, the Fig. 6 reveals that 97.5% of samples are accurately detected. For the first and second rows of the diagonal block, it is shown that 33.47% of the samples, or 160 samples, are accurately classified as benign and 64.02%, or 306 samples, are correctly classified as malignant. Both the benign and malignant instances

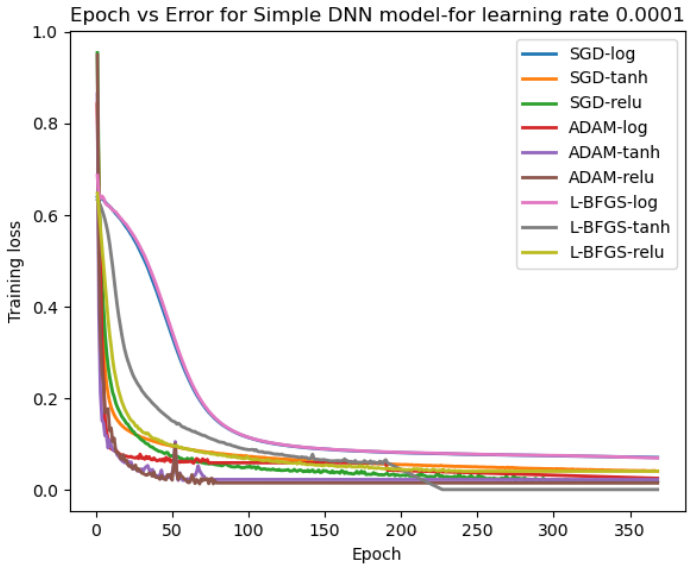

Figure 4.7: Epoch vs Error for Simple DNN model-for learning rate 0.0001

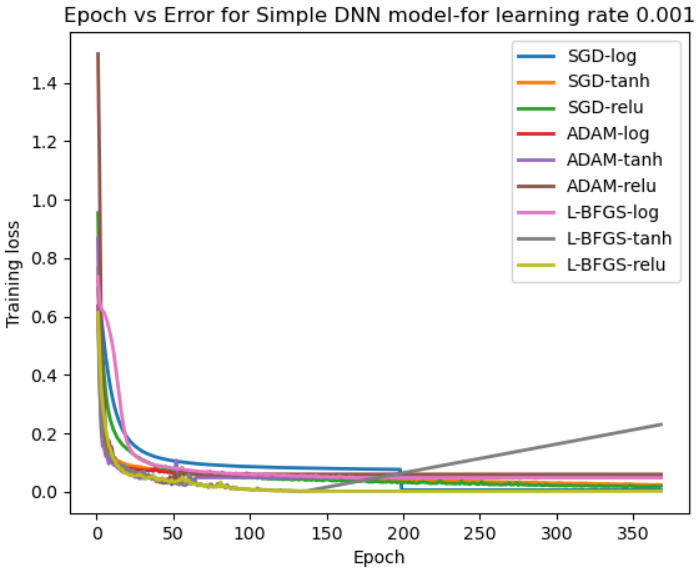

Figure 4.8: . Epoch vs Error for Simple DNN model-for learning rate 0.001

represented by the other two blocks were misdiagnosed. A total of 0.21% i.e. only one malignant sample is mistaken as a benign sample and 2.30% i.e. 11 benign samples is mistaken as malignant samples.

Fig. 4.11 depicts the confusion matrix for testing data. 97.1% success rate for correctly diagnosing tumors is shown in fig. 4.11 for the network under consideration. According to the first two diagonal blocks, 36.59%, or 75 samples, are accurately classified as benign, whereas 60.49%, or 124 samples, are appropriately classified as malignant.

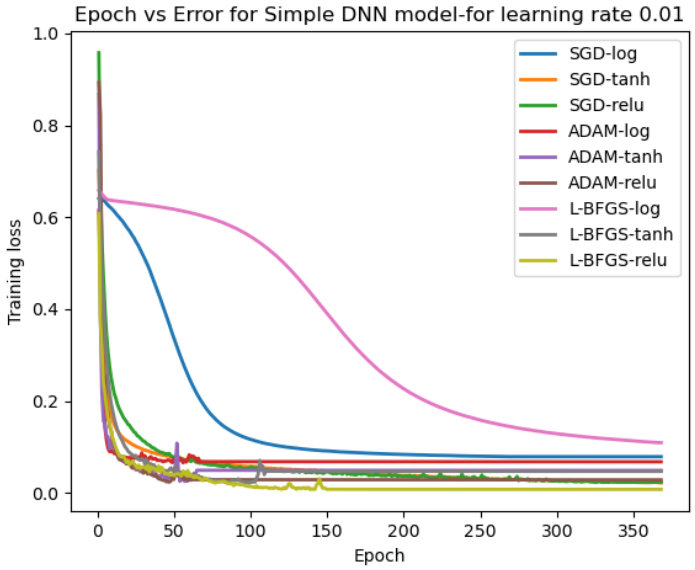

Figure 4.9: Epoch vs Error for Simple DNN model-for learning rate 0.01

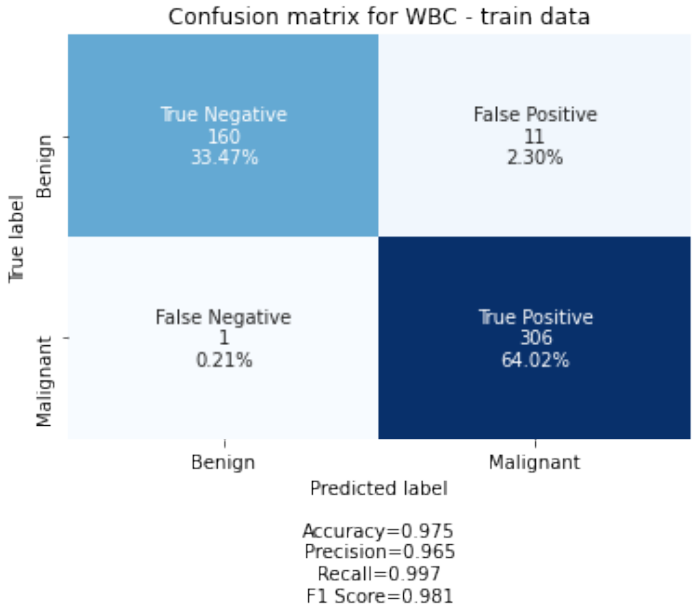

Figure 4.10: Confusion matrix for train data set-Case 1

Comparative analysis of different optimizers with different activation functions is shown in table 4.2. DNN obtained the greatest accuracy of 97.1% with 97.6% precision and 97.6% recall for Adam algorithm with logistic activation function in 1.47 seconds with 100-100 neurons at two hidden layers.

Precision-Recall curve and ROC curve for a simple model are shown in fig. 4.12 and fig. 4.13, which are used to assess the efficiency and ability of the model and to determine if it is great, good, or awful in working with test data. Due to the fact that it is not closer to the top left corner of the ROC curve, this curve is not the

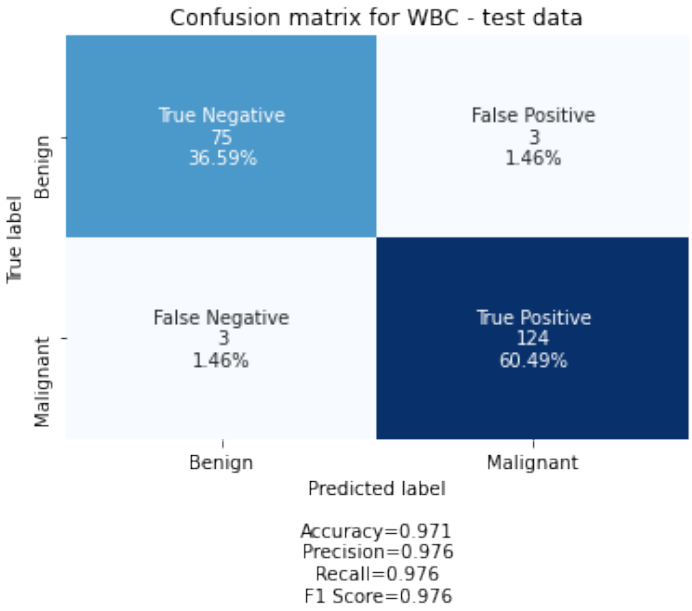

Figure 4.11: Confusion matrix for test data set-Case 1

Table 4.2: DNN without ICA and 10-fold CV - Case 1

Optimization algorithm	Activation function	Accuracy (%)	Time (seconds)	Iterations
SGD	Logistic	95.61	0.87	197
	Tanh	96.07	42.65	192
	ReLU	96.59	35.89	191
Adam	Logistic	97.10	1.47	154
	Tanh	96.09	1.309	258
	ReLU	97.07	2.84	47
L-BFGS	Logistic	97.07	54.02	177
	Tanh	95.61	2.26	60
	ReLU	95.61	2.89	122

best for classification. The attainment of a desirable outcome of network prediction is represented by the precision-recall curves.

Case 2: Results for R-DNN with ICA and 10-fold CV (Proposed R-DNN model)

In this scenario, data is separated into 10 folds for the purposes of training and testing the model. Each fold is treated as a training and testing set. The fig. 4.14, fig. 4.15 and fig. 4.16 demonstrates the varied learning rates for the training error. After training, the version of the network that performed the best on the validation set was used.

This is the case in which the 10-fold CV and ICA are applied on WBC data. Ta-

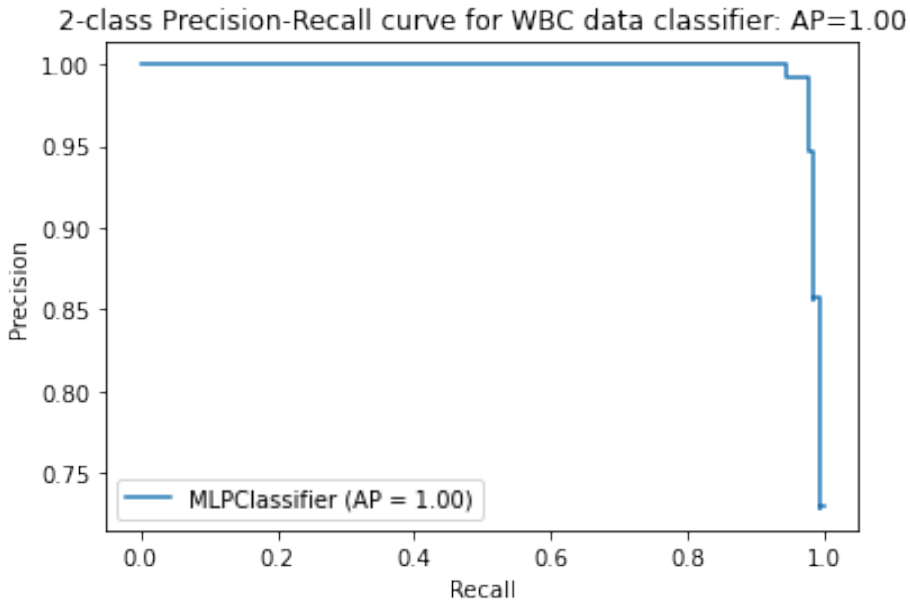

Figure 4.12: 2-class Precision-Recall Curve - Case 1

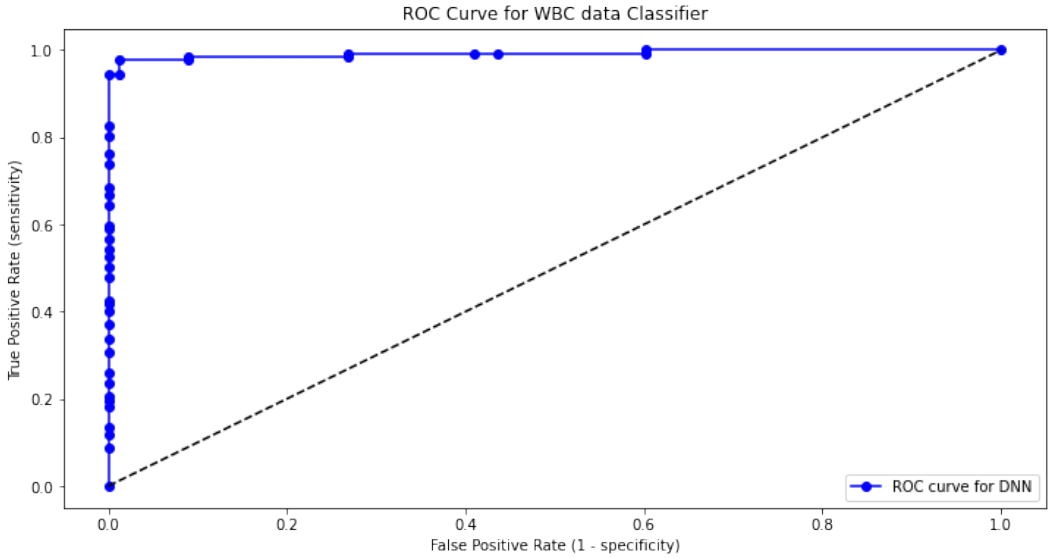

Figure 4.13: ROC Curve - Case 1

ble 4.3 compares the performance of various optimization algorithms with varying activation functions. With Logistic, Tanh and ReLU. L-BFGS optimizers fluctuate a lot and also didn't reach to optimal solution and leads to divergence. Such that L-BFGS mislead to the solution. As it can be observed in fig. 4.14, fig. 4.15 and fig. 4.16. With Logistic, Tanh, and ReLU activation functions, SGD and Adam optimizer provide 100% accurate predication. Adam optimizer with ReLU activation function provides the maximum accuracy for 100-100 neurons at each hidden layers, 0.001 learning rate with reduced training time, and 163 numbers of iterations, which is superior to the accuracy produced by other optimizers. Out of all other trials with

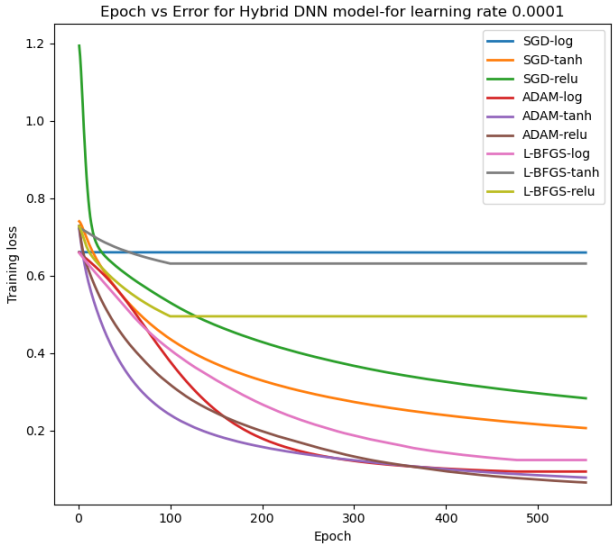

Figure 4.14: Epoch vs Error for Proposed R-DNN model-for learning rate 0.0001

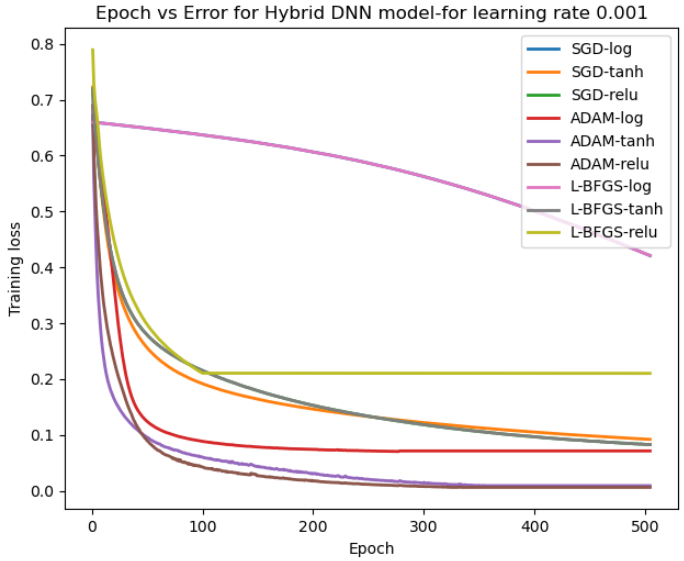

Figure 4.15: Epoch vs Error for Proposed R-DNN model-for learning rate 0.001

varied hidden neurons, regularisation parameter, and learning rate, we received the best results when the regularization value was set to 0.0001 and the learning rate was set to 0.001.

Confusion matrix for WBC train-test data is shown in fig. 4.17 and fig. 4.18. The first diagonal block in the Fig. 13 represents the trained network's correct benign and malignant diagnosis rate. The first two diagonal blocks indicate that out of 699 samples, 35.40%, or 218 samples, are accurately classified as benign, while 61.30%, or 377 samples, are correctly classified as malignant. 1.95%, or 12 instances, receive

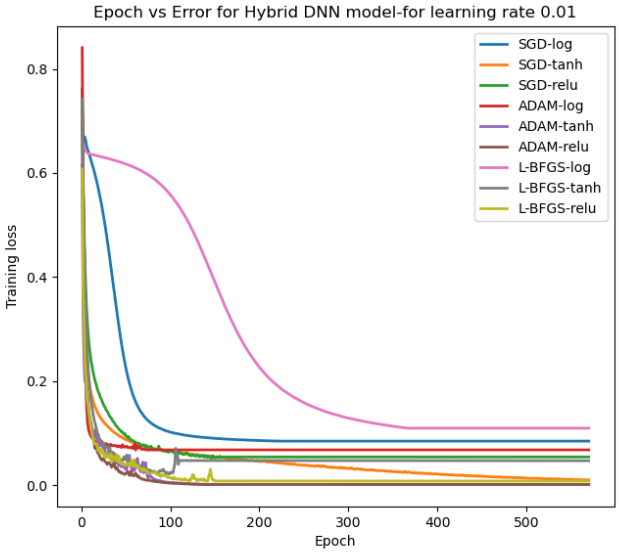

Figure 4.16: Epoch vs Error for Proposed R-DNN model-for learning rate 0.01

Table 4.3: R-DNN with ICA and with 10-fold CV - Case 2

Optimization algorithm	Activation function	Accuracy (%)	Time (seconds)	Iterations
SGD	Logistic	100.00	4.17	1449
	Tanh	100.00	3.38	505
	ReLU	100.00	9.31	605
Adam	Logistic	100.00	2.03	149
	Tanh	100.00	2.4	136
	ReLU	100.00	1.81	163
L-BFGS	Logistic	98.51	1.39	120
	Tanh	98.51	0.98	78
	ReLU	97.06	1.79	121

an incorrect diagnosis of malignancy, while 1.30%, or 8 samples, receive an incorrect diagnosis of benign. Similarly, the confusion matrix for the WBC test data set is illustrated in the Fig. 14. 55 samples, or 80.88% of 699 total cases, are accurately classified as benign, while 13 samples are appropriately classified as malignant (4).

The remainder of the block represents the number of benign and malignant tumours misdiagnosed. No specimens are misdiagnosed in benign or malignant. Hence, we claim that the proposed model worked exceptionally well, achieving a perfect accuracy rate of 100%.

Overall, 100% of predictions were properly classified, and 100% of precision, recall, and F-score were attained for the diagnosis of a cancerous tumour. The Precision-Recall curve for a 2-class system is depicted in fig. 4.19.

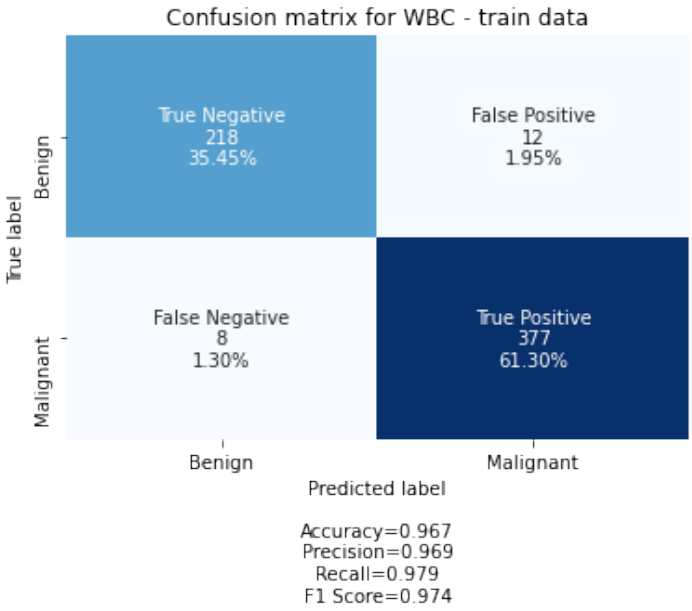

Figure 4.17: Confusion matrix for train data set-Case 2

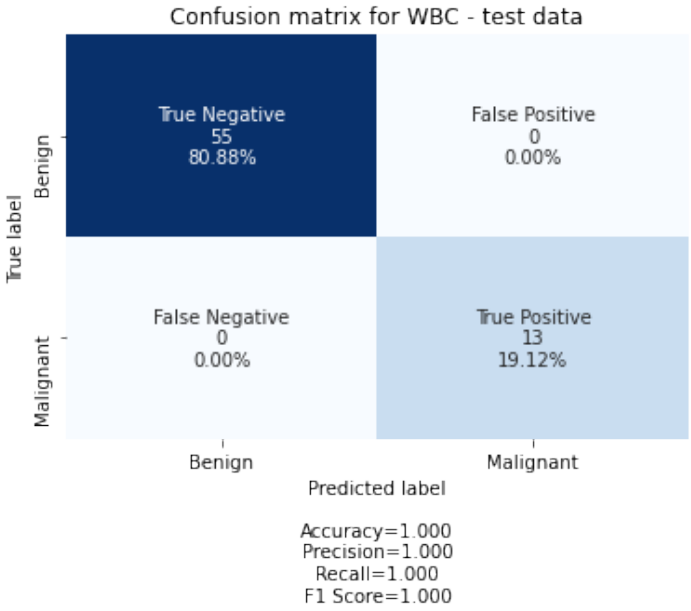

Figure 4.18: Confusion matrix for test data set-Case 2

Fig. 4.20 depicts a Receiver Operating Characteristic curve (ROC curve), which indicates that the model can accurately and perfectly diagnose WBC data. In this case, AUC = 1 indicates that all test data results in the proper classification of benign and malignant tumours for the provided model.

A comparison of categorization approaches with the work of other authors is shown in table 4.4. In the illustration, you can see a graphical depiction in fig. 4.21 of table 4.4.

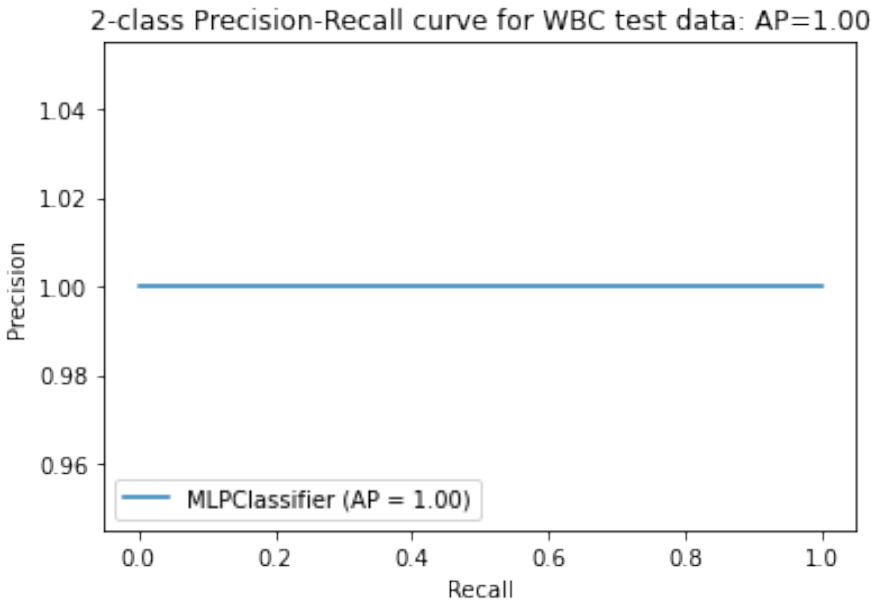

Figure 4.19: 2-class Precision-Recall Curve - Case 2

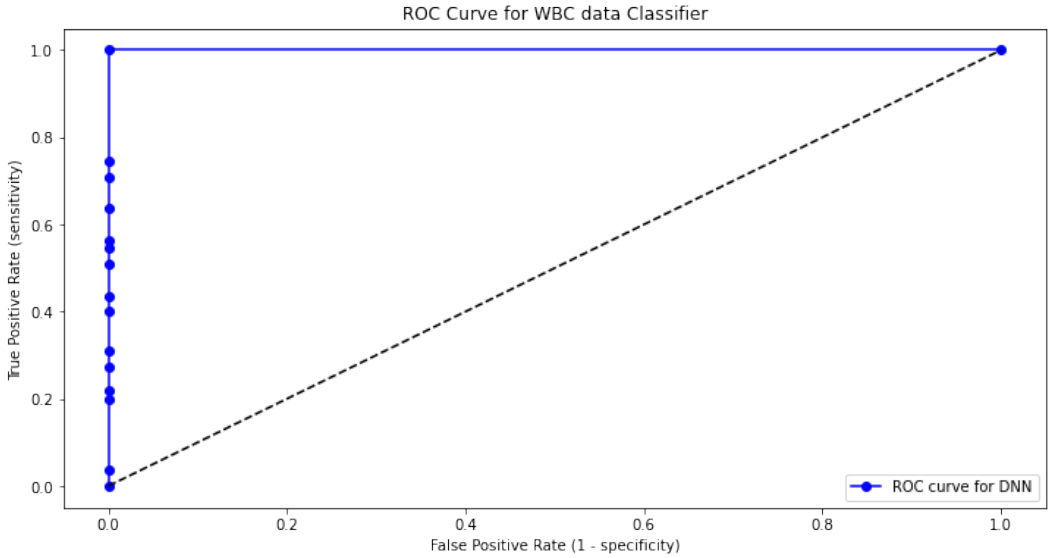

Figure 4.20: ROC Curve - Case 2

4.5.2 Experiments and results of WDBC data set

To perform the experiment on WDBC data set, we use the following architecture. The network consists of 30 input nodes at input layer, 2 hidden layers and 1 neuron at output layer. Different pairs of neurons at hidden layers are chosen like 25-10, 50-25, 100-120, 100-100, 250-100, 250-250, 500-100, 500-500, 1000-1000 to perform the experiments and network is trained and tested with different learning rates as 0.01, 0.001, 0.0001. The best results under each scenario are listed in the tables. We perform the experiments using two different approaches.

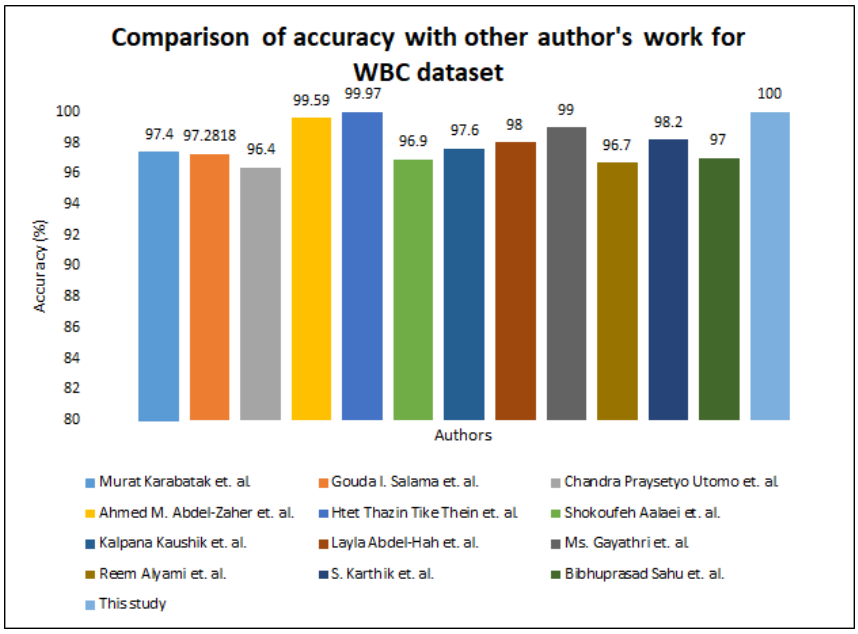

Figure 4.21: Performance analysis of the proposed R-DNN+ICA model with others authors

Table 4.4: Performance analysis of the proposed R-DNN+ICA model

Authors	Year	Methods	Accuracy (%)
Murat Karabatak et. al.	2009	ANN+AR1 (Association Rule)	97.40
Gouda I. Salama et. al.	2012	Fusion of SMO+IBK+NB+J48	97.28
Chandra Praysetyo Utomo et. al.	2014	ELM+ANN	96.40
Ahmed M. AbdelZaher et. al.	2015	DBN-NN(Conjugate gradient BP) RIW-BPNN(Conjugate gradient BP) DBN-NN (Levenberg Marquardt) RIW-BPNN (Levenberg Marquardt)	99.59 98.86 99.68 99.03
Htet Thazin Tike Thein et. al.	2015	ANN	99.97
Shokoufeh Aalaei et. al.	2016	ANN PS-classifier GA-classifier	96.70 96.90 96.60
Kalpana Kaushik et. al.	2016	ANN	97.60
Layla Abdel-Hah et. al.	2017	ANN	98.00
Ms. Gayathri et. al.	2017	Cascade-forward BP	99.00
Reem Alyami et. al.	2017	ANN SVM	96.70 97.14
S. Karthik et. al.	2018	DNN+RFF	98.20
Bibhuprasad Sahu et. al.	2019	ANN	97.00
This study	2021	R-DNN+ICA	100.00

Case 1: Results for DNN without ICA and 10-fold CV (Simple DNN model)

In this case, the data set is split into 80-20% train-test set. Table 4.5 shows the comparison of accuracy with different optimizers and different activation functions.

We see that highest accuracy of 97.10% is achieved for Adam optimizer with Tanh activation function in just 3.78 seconds. This accuracy is obtained by choice of both the regularization parameter as 0.01 and learning rate as 0.0001 and number of neurons at hidden layers as 25-10.

Table 4.5: DNN without ICA and 10-fold CV - Case 1

Optimization algorithm	Activation function	Accuracy (%)	Time (seconds)	Iterations
SGD	Logistic	94.15	6.24	2150
	Tanh	95.10	63.23	465
	ReLU	95.91	3.72	68
Adam	Logistic	97.07	11.33	1823
	Tanh	97.10	3.78	2028
	ReLU	96.49	3.35	1620
L-BFGS	Logistic	95.91	36.20	11197
	Tanh	95.32	23.85	3576
	ReLU	69.84	0.16	4

Figure 4.22 and fig. 4.23 shows the confusion matrix for train-test data set.

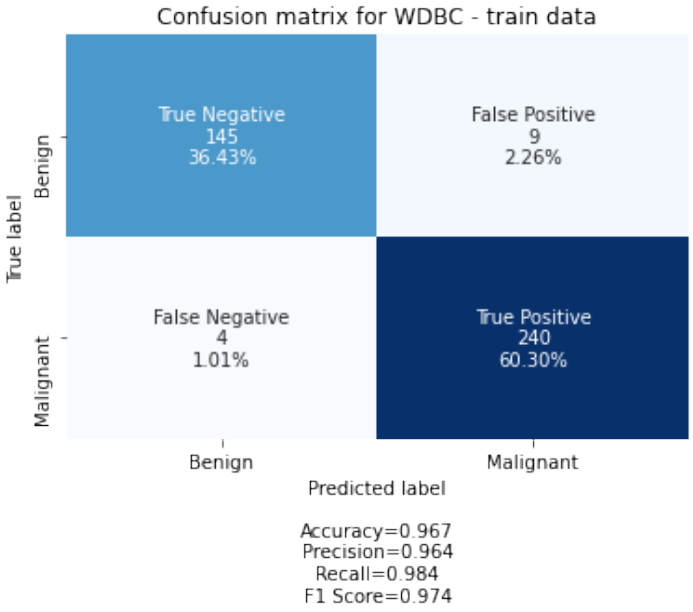

Figure 4.22: Confusion matrix for train data set - Case 1

Figure 4.24 represents the ROC curve and fig. 4.25 represents 2-class Precision-Recall curve for WDBC classifier. These tools are used to measure the efficiency and ability of model. This ROC curve is not best for classification as it is not closer to the top left corner of ROC. Precision-Recall curve represents the achievement of favorable outcome of prediction of network. Adam optimizer gives best result with 97.2% of precision and 98.1% of recall.

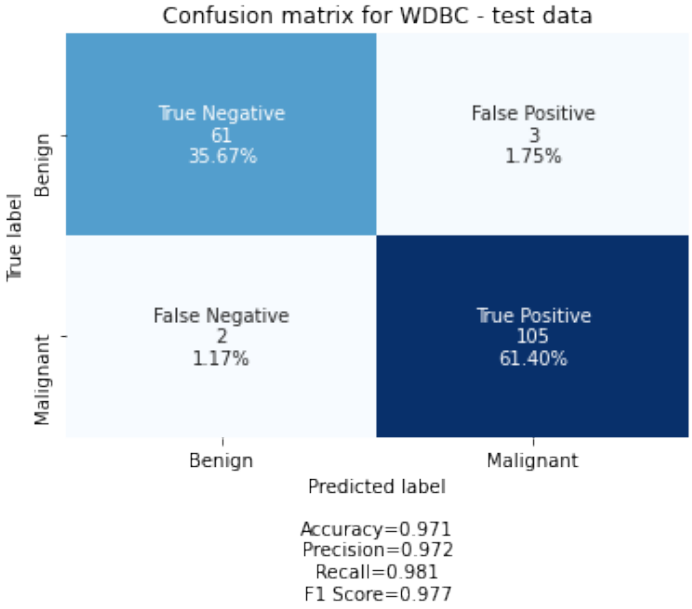

Figure 4.23: Confusion matrix for test data set - Case 1

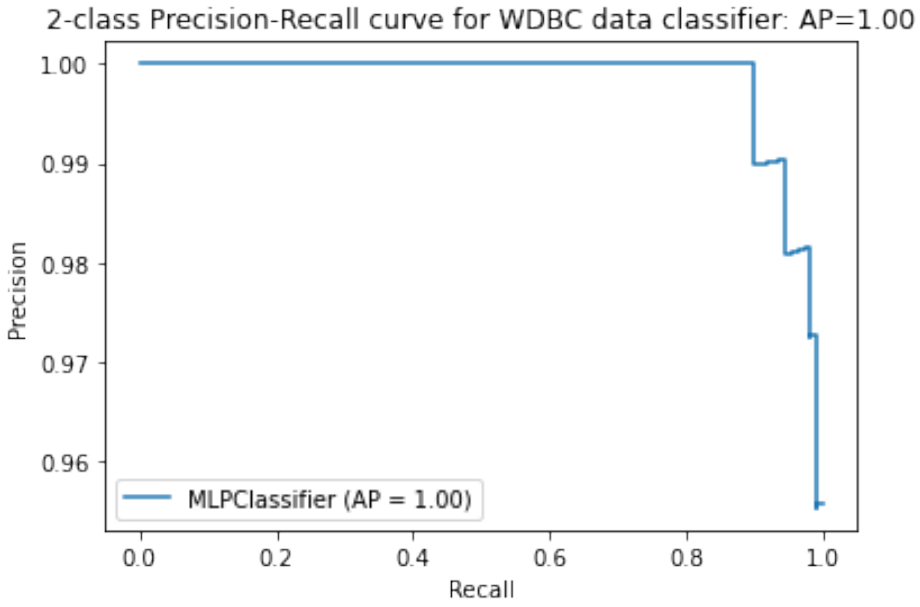

Figure 4.24: 2-class Precision-Recall Curve - Case 2

Case 2: Results for R-DNN with ICA and 10-fold CV (Proposed R-DNN model)

In this experiment, we use ICA for feature selection and 10-fold CV to train and test the DNN. On choosing learning rate as 0.0001, regularization term as 0.5 and pair of two hidden layers having 500 neurons each. We obtain best accuracy under different scenarios but time taken is very large. But on choosing 25-10 neurons at two hidden layers, we obtain 100% accuracy with Adam optimizer and ReLU

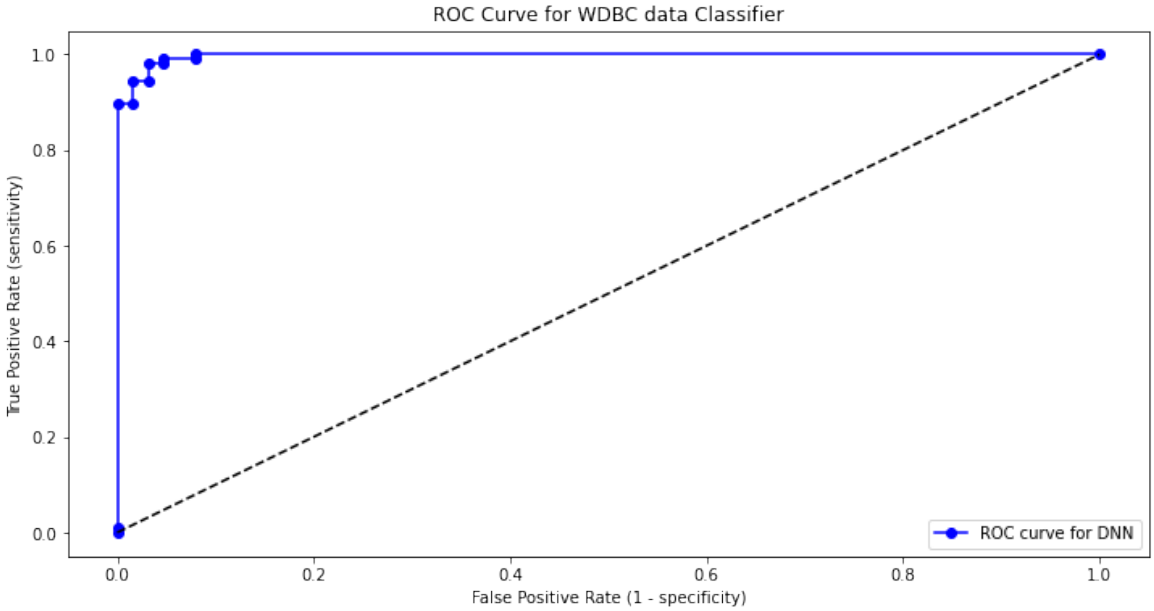

Figure 4.25: ROC Curve - Case 2

activation function within 2 seconds. Table 4.6 shows the comparison of accuracy with different optimizers and different activation functions.

Table 4.6: R-DNN with ICA and with 10-fold CV - Case 2

Optimization algorithm	Activation function	Accuracy (%)	Time (seconds)	Iterations
SGD	Logistic	92.86	3.32	580
	Tanh	94.64	3.05	779
	ReLU	98.21	2.70	707
Adam	Logistic	98.25	3.32	580
	Tanh	94.64	3.04	779
	ReLU	100.00	1.80	643
L-BFGS	Logistic	100.00	305.75	3001
	Tanh	100.00	249.69	3001
	ReLU	23.21	1.19	3

Figure 4.26 and fig. 4.27 shows the confusion matrix of train-test data set for this experiment. In both trained network and tested network, we achieved 100% accuracy with 100% precision, recall and $F1$-score.

Figure 4.28 shows the ROC curve for WDBC classifier. This plot represents area under the ROC curve which is 1 and it proves the perfect test of classifier.

Figure 4.29 represents the 2-class Precision-Recall curve which shows that diagnosis of tumor is 100% correct.

We have also compared our results with the classification accuracy obtained by other

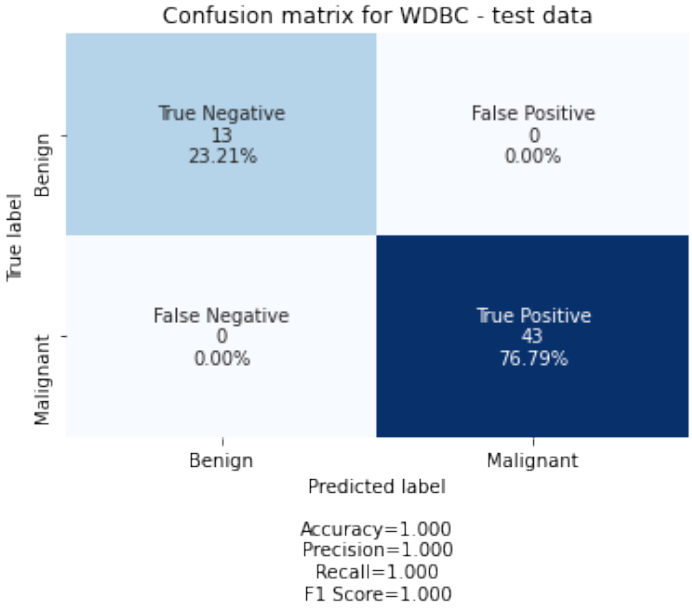

Figure 4.26: Confusion matrix for train data set - Case 2

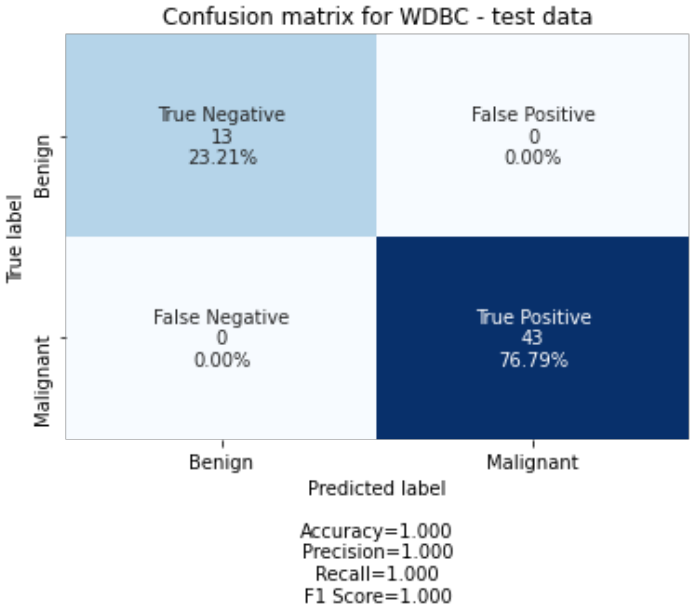

Figure 4.27: Confusion matrix for test data set - Case 2

authors who worked for the same WDBC data set in their research using different tools. These results are listed in table 4.7.

The detailed information of the both data set are given in the Appendix. Also, above all computation is carried out using Python programming and it is given in the Appendix.

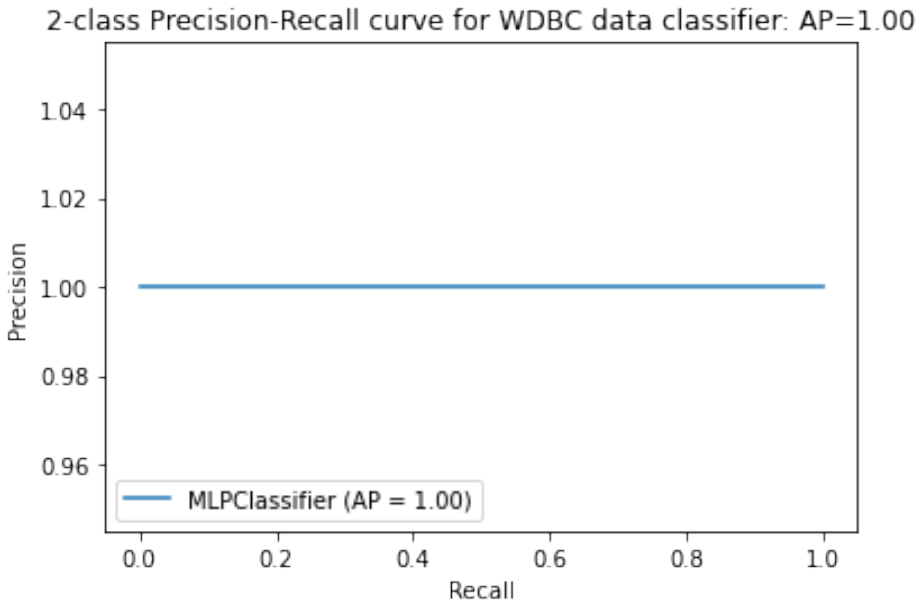

Figure 4.28: 2-class Precision-Recall Curve - Case 2

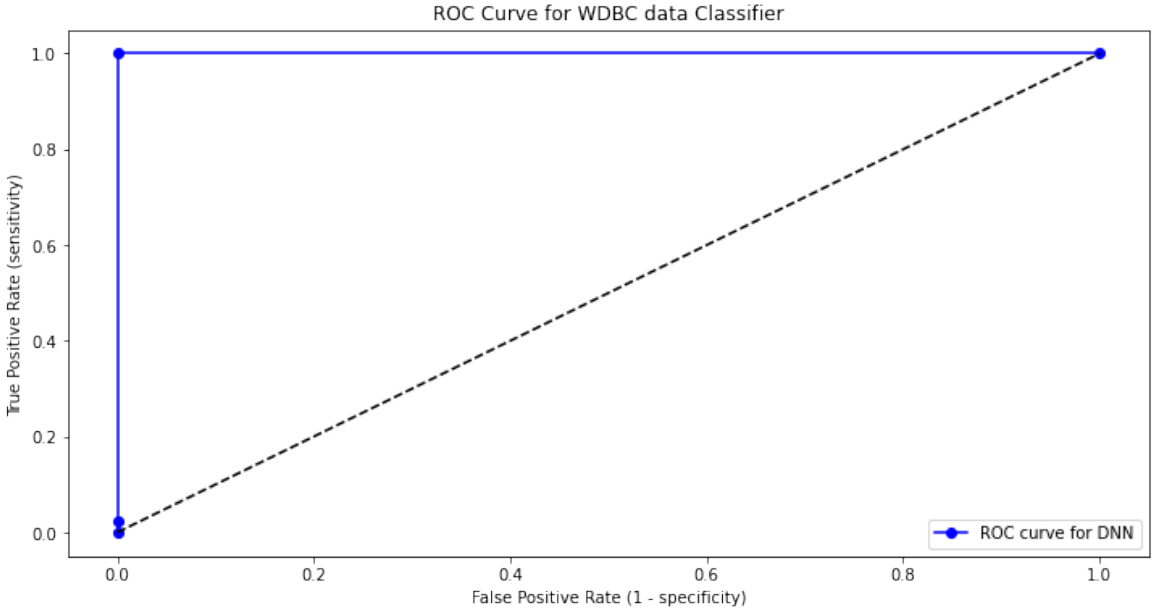

Figure 4.29: ROC Curve - Case 2

Table 4.7: Performance analysis of the proposed R-DNN+ICA model

Authors	Year	Methods	Accuracy (%)
I. Maglogiannis, et al.	2007	ANN	97.90
G. I. Salama, et al.	2012	Fusion of SMO + IBK + NB + J48	97.01
G. Zorluoglu, et al.	2015	ANN	97.54
		SVM	98.07
		Ensemble	98.77
		C5.0	98.07
A. Mart, et al.	2015	RBFNN+ICA	69.63
		SVM	97.47
S. Aalael, et al.	2016	ANN	97.30
		PS-classifier	97.20
		GA-classifier	96.60
A. F. M. Agarap, et al.	2018	MLP	99.04
D. A. Omondiagbe, et al.	2019	ANN+LDA	98.82
		ANN+PCA	97.65
M. Tiwari, et al.	2020	ANN	99.30
		CNN(DL)	97.30
This study	2020	DNN+ICA	100.00

4.6 Conclusion

In this study, we investigated and observed the influence of ICA and 10-fold CV in the diagnosis of breast cancer. We compared the results to a simple model with proposed hybrid R-DNN model. Using the WBC data set, we discovered that R-DNN with ICA and 10-fold CV provide the best accuracy rate, as well as 100% precision and recall. We observed DNN with ICA and 10-fold CV for WDBC data set gives highest accuracy rate with 100% of precision and recall. With ICA, dimension is significantly reduced without loss of generality and which made the faster classification. The tools of ROC curve and 2-class Precision-Recall curve are also used to measure the efficiency and ability of the model. By introducing additional term L2 regularization, the error is controlled so that the coefficients do not take on extreme values when the function is excessively fluctuating. By modifying the weights of the penalty term, the hyper parameter $\lambda = 0.001$ adjusts the trade-off between how well the data fits and how complex the model is. If λ is increased, the model complexity will contribute more to the cost. Because the minimal cost hypothesis has been chosen. This means that a higher λ will favour the model with the lowest complexity. Dimensions were decreased to only three features without sacrificing generality and classification accuracy attained to 100%. In proposed model, the sensitivity and specificity are 100%, resulting in the ROC curve of 1, indicating that there is no erroneous prediction in either the benign or malignant cases studied. The results obtained in this study are compared with the results of other researchers working in the same area and we observed that our model is the best one which gives 100% accuracy.