PROJECT REPORT

On

"PARTICLE SWARM OPTIMIZATION USING FOR THE PREDICTION OF BREAST CANCER"

Submitted in Partial Fulfilment of VI semester Bachelor of Computer Application for the ACADEMIC YEAR 2022

SUBMITTED BY

KEERTHANA K R 19mLACB1027

Under the Guidance of

Prof. Bharathi D S B.E, MTech
Assistant Professor

Prof. Deeksha Holla BCA, MCA
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE

MAHARANI LAKSHMI AMMANNI COLLEGE FOR WOMEN AUTONOMOUS

Science P.O, Malleshwaram 18th Cross, Bengaluru, Karnataka 560012

MAHARANI LAKSHMI AMMANNI COLLEGE FOR WOMEN AUTONOMOUS

Science P.O, Malleshwaram, Bengaluru, Karnataka 560012

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that project work entitled ***"PARTICLE SWARM OPTIMIZATION USING FOR THE PREDICTION OF BREAST CANCER"*** *has been successfully submitted by*

KEERTHANA K R ***19mLACB1027***

a bonafide student of ***MAHARANI LAKSHMI AMMANNI COLLEGE FOR WOMEN AUTONOMOUS*** *in partial fulfilment of the requirement for VI semester BCA prescribed by BCU during the academic year 2022.*

Flt.Lt. Harish H
Head of Department,
Department of Computer Science,
mLAC

INTERNAL GUIDES

1. **Prof. Bharathi D S**
2. **Prof. Deeksha Holla**

EXAMINERS

1.
2.

ACKNOWLEDGEMENT

I thank the management of our institution Maharani Lakshmi Ammanni College for Women Autonomous, for providing us with all the facilities required to complete this project.

I deeply indebted to the principal **Dr. NAGALAXMI B N** for her support in providing all the facilities that was necessary to complete this project successfully.

I would like to take this opportunity to thank our **HOD Flt. Lt Harish H,** Department of computer science for his support and guidance throughout the process of project design and development.

I take this privileged to thank our project guides **Prof. Bharathi D S** and **Prof. Deeksha Holla** for their constant support in making this project successful. At last, we also wish to thank teaching, non-teaching staff, family and friends for their continuous support, motivation and valuable guidance.

I take great pleasure in expressing our gratitude to all the people who have directly or indirectly helped us to accomplish this project successfully on time.

KEERTHANA K R

ABSTRACT

Cancer is the second cause of death in the world. 10 million patients died due to cancer in 2021. Breast cancer is the leading cause of death among women. Breast cancer affects one out of eight females worldwide. It is diagnosed by detecting the malignancy of the cells of breast tissue. Modern medical image processing techniques work on histopathology images captured by a microscope, and then analyse them by using different algorithms and methods. Machine learning algorithms are now being used for processing medical imagery and pathological tools. In deep learning, this is often done by extracting features through a convolutional neural network (CNN) and then classifying employing a fully connected network. In this project, Convolutional Neural Network (CNN) is used to extract features, Particle Swarm Optimization (PSO) to select features and obtained prediction accuracy using Support Vector Machine (SVM).

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

The breast is made up of different tissue, ranging from very fatty tissue to very dense tissue. Within this tissue is a network of lobes. Each lobe is made up of tiny, tube-like structures called lobules that contain milk glands. Tiny ducts connect the glands, lobules, and lobes, carrying milk from the lobes to the nipple. The nipple is located in the middle of the areola, which is the darker area that surrounds the nipple. Blood and lymph vessels also run throughout the breast. Blood nourishes the cells. The lymph system drains bodily waste products. The lymph vessels connect to lymph nodes, the small, bean-shaped organs that help fight infection. Groups of lymph nodes are located in different areas throughout the body, such as in the neck, groin, and abdomen. Regional lymph nodes of the breast are those nearby the breast, such as the lymph nodes under the arm.

Cancer begins when healthy cells in the breast change and grow out of control, forming a mass or sheet of cells called a tumor. A tumor can be cancerous or benign. A cancerous tumor is malignant, meaning it can grow and spread to other parts of the body. A benign tumor means the tumor can grow but will not spread.

Breast cancer spreads when the cancer grows into adjacent organs or other parts of the body or when breast cancer cells move to other parts of the body through the blood vessels and/or lymph vessels. This is called a metastasis.

This guide covers both non-invasive (stage 0) as well as early-stage and locally advanced invasive breast cancer, which includes stages I, II, and III. The stage of breast cancer describes how much the cancer has grown, and if or where it has spread.

Although breast cancer most commonly spreads to nearby lymph nodes, it can also spread further through the body to areas such as the bones, lungs, liver, and brain. This is called metastatic or stage IV breast cancer.

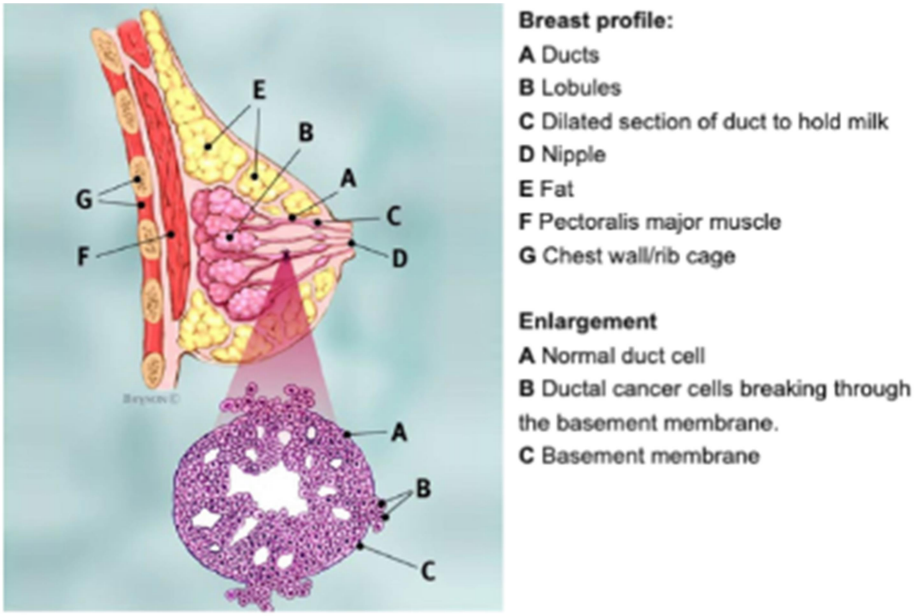


Figure 1.1 Invasive Dutual Carcinoma (IDC)

In case of IDC, the cancer would break through the duct cell's membrane and invade or spread to the nearby tissues. In this case the cancer would start in ductal tissue of the breast, duct is the tube that connects the lobules to the nipple. Carcinoma is the type of cancer that starts in the tissue that covers the internal organs, like breast cell in this case. If not detected early, invasive ductal carcinoma can invade the other tissues within breast or even other parts of the body.

Even though invasive ductal carcinoma is most common in older women, it can still affect younger women as well as men. Early detection of the breast cancer will help in increasing the survival rates of the patients. Currently the histopathologists diagnose the tissues extracted from the suspicious tumors and provide information related to type of cancer and its grade when the tumor tests to be malignant. Data

science can help in evaluating the suspected tumor tissues through an automated computer vision workflow and provide the diagnosis along with grade there by saving time as well increasing the accuracy of the diagnosis.

The treatment and diagnosis of breast cancer in an early stage is necessary for the reduction of death rates and to prevent the progression of the disease. The pathological images serve as effective standards for doctors and research scholars for the diagnosis of disease. The variations in morphological features of nuclei images are the source for tumor detection and diagnosis. So, the effective detection of nuclei in images is basic requirement for efficient diagnosis of breast cancer processes.

The biopsy samples of tissue or cells are kept on the glass slide for the process of staining and frequent microscopic examination. The pathologists view the glass slides under the microscope for analyzing various components of tissue that is dyed with more than one stain to examine the cellular level of the tumor. Hematoxylin-Eosin (H-E) staining is the most utilized staining method by pathologists. The Nuclei are stained by purple and blue color with Hematoxylin stain.

In the background structure, the stroma and cytoplasm are stained by color pink by using Eosin stain. The size of the nucleus, shape, ratio of cytoplasm, and chromatin patterns are considered in the tissue affected by breast cancer.

The detection of cell nuclei is the core operation in CAD for cancer detection. The CAD serves as the basis for counting the cells and for the study of subcellular morphology such as the investigation of shape, texture, and size of the cellular components. The detection of cell nuclei is difficult during the analysis of histopathological images such as target cells that are in various states. The smaller cells will be surrounded by the background clutters made up of histopathological structures such

as collagen, capillaries, etc., along with the irrelevant visual aspects and artefacts which occur during the acquisition of images.

The challenges faced during the automatic analysis of breast cancer from the histopathological images were mainly the complex and diverse structures. The difference in the appearance of H-E stained images was due to the various concentrations of the applied stains and inconsistent scanning of the images which significantly degrades the performance of subsequent processes of nuclei detection.

The normalization of stain is carried out to enhance the quality and consistency of appearance in the images. Another issue faced was less availability of training data, a larger size of biopsy images, and resizing the images, all of which resulted in the loss of detailed information.

The existing deep learning methods used the contour information for accurate detection of breast cancer. But, the existing method showed lesser performance by extracting only the semantic information from the former layer and not extracted the details from the shallow layer which is required for effective breast cancer detection

1.1 OBJECTIVE

The objective of the whole project is to detect the breast cancer using machine learning algorithms based on image processing.

1.2 PROBLEM STATEMENT

- Detection of breast cancer using machine learning and deep learning algorithms

- To understand the effectiveness of used algorithms for detecting the breast cancer.

1.3 JUSTIFICATION

Rapid growth in advanced digital technology in the field of breast cancer research as led to wide increase in understanding various clinical conditions, treatment protocols, to increase the survival rate of the patients. Therefore, this topic has chosen to identify, analyse, and expand the research study for betterment of patient's wellbeing.

1.4 SCOPE OF WORK

The purpose of this project is to detect breast cancer using particle swarm optimization and measure the accuracy.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

Rajesh Saturi *et al*., [1] have used the histopathology image dataset. This dataset undergoes preprocessing and then features were extracted using Convolutional Neural Network (CNN) pre-trained VGG-19 model, Histogram of Gradients (HoG) and Local Ternary Pattern (LTP). Also they have proposed the multi-objective feature selection method using Ant Colony Optimization (ACO) with Particle Swarm Optimisation (PSO) by selecting optimal features. Classification is done by Long Short-Term Memory Network (LSTM). By doing this proposed model they classified the dataset into 8 classes such as Ductal –Carcinoma, Lobular-Carcinoma, Mucinuo-Carinoma, Papilary-fibrodenoma, carcinoma, adenosis, phyllodes tumor and tubular-adenoma. They have secured the accuracy of 95.72%.

Rizki Habibi *et al*., [2] have done two methods to find the best algorithm. They have done one classification using Support Vector Machine (SVM) without optimisation and other classification with Support Vector Machine (SVM) and with optimization using Particle Swarm Optimisation (PSO) using matlab on the Wisconsin dataset. They have concluded that combination of SVM and PSO is quite efficient in finding the best parameter value. They have secured the accuracy of 78.91%.

Rachida Touami *et al,*. [3] have done image segmentation using region growing on 110 mammogram images taken from DDSM database. Later they have given segmented images as input for classification, which is done by Probabilistic Neural Network (PNN) trained by Particle Swarm Optimisation (PSO) to estimate optimal value of the parameter. They have proposed this method on mammogram images. They have secured accuracy rate of 96% using this method.

Badriya Al Maqbali *et al.,* [4] have done preprocessing on mammogram images using a median filtering model and Contrast Limited Adaptive Histogram Equalization (CLAHE). After preprocessing, the preprocessed has sent to segmentation which is done region growing algorithm. Later, they have done feature extraction to extract features like gradient features, geometric features and textures. Feature selection has been done using Hybrid Wolf Pack Algorithm And Particle Swarm Optimization, this novel optimization is called as hybrid WPA-PSO (Wolf Pack Algorithm – Particle Swarm Optimization). Lastly, they have done classification using NN classifier. They have secured the accuracy of 83.83%.

Jesutofunmi Onaope Afolayan [5] have performed breast cancer detection on wiscosin dataset. They have done optimization using particle swarm optimization and performed classification using decision tree algorithm and secured the accuracy of 92.26%.

Nashat Alrefat *et al.,* [6] have performed feature selection using particle swarm optimization and classification using ensemble learning. The accuracy secured is 86.36%.

Anusha Papasani *et al.,* [7] have performed classification using particle swarm optimization. The classification using machine learning classifiers, such as logistic regression (LR), Naive Bayes (N-Bayes), decision tree learning (DTL), and K-nearest neighbor (KNN) and compared the results secured.

Yuhua Ma *et al.,* [8] have performed classification on sample data collected from patients with cervical cancer, CIN (cervical intraepithelial neoplasia) I, CIN II, CIN III and hysteromyoma using FT-IR (Fourier-transform infrared spectroscopy) technology. They have performed classification using PSO-CNN model and accuracy secured is 87.2%.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATIONS

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

- Machine name: HP
- Processor: Intel core i5, 2.50 GHz
- RAM: 8 GB

3.2 SOFTWARE REQUIREMENTS

- Operating system: Windows 10
- Front end tools: Python, Data Analytics
- Web browser: Google Chrome
- Software: Jupyter Notebook

3.3 TECHNOLOGIES USED

3.3.1 PYTHON

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. It is,

- Free and open-source - You can freely use and distribute Python, even for commercial use.
- Easy to learn - Python has a simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like C++, Java, and C #.
- Portable - You can move Python programs from one platform to another, and run it without any changes.
- Python has a lot of applications. It's used for developing web applications, data science, rapid application development, and so on.
- Python allows you to write programs in fewer lines of code than most of the programming languages.
- The popularity of Python is growing rapidly. Now it's one of the most popular programming languages.

3.3.2 JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output. Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects. Main features of the web application:-

- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
- The ability to execute code from the browser, with the results of computations attached to the code which generated them.
- Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc. For example, publication-quality figures rendered by the matplotlib library, can be included inline.
- In-browser editing for rich text using the Markdown markup language, which can provide commentary for the code, is not limited to plain text.
- The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax.

CHAPTER 4

SYSTEM ANALYSIS

CHAPTER 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

In existing system, breast cancer classification was done using Convolutional Neural Network. This was done by creating model and called it as “cancernet”. This was done to know that whether it is the better algorithm to classify the data accurately. The dataset used was breast histopathology images. The proposed model was classifying the dataset into malignant and benign.

4.2 PROPOSED SYSTEM

In order to assess and identify diseases in medical images, machine learning techniques were applied. Many machine learning and deep learning approaches have been widely employed in medical image processing in recent years to detect and evaluate items in medical images.

This is image processing project. It is based on detection of breast cancer using machine learning algorithms. In this project, Kaggle dataset is used which is breast histopathology images, feature extraction is performed using Convolutional Neural Network (CNN) on 1,033 50X50 RGB images. After feature extraction, feature selection is done using Particle Swarm Optimization (PSO). After feature selection, accuracy is predicted using Support Vector Machine (SVM).

CHAPTER 5

ALGORITHMS AND PYTHON LIBRARIES USED

CHAPTER 5

ALGORITHMS

5.1 CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks are a type of Deep Learning Algorithm that take the image as an input and learn the various features of the image through filters. This allows them to learn the important objects present in the image, allowing them to discern one image from the other. For example, the convolutional network will learn the specific features of cats that differentiate from the dogs so that when we provide input of cats and dogs, it can easily differentiate between the two.

One important feature of Convolutional Neural Network that sets it apart from other Machine Learning algorithms is its ability to pre-process the data by itself. Thus, you may not spend a lot of resources in data pre-processing. During cold-start, the filters may require hand engineering but with progress in training, they are able to adapt to the learned features and develop filters of their own. Therefore, CNN is continuously evolving with growth in the data.

A CNN model includes 4 types of layers that are implemented as follows. The input preprocessed images are passed into convolutional layers and estimated the number of neurons connected with the regions of input images. Every neuron is estimated by dot product among smaller weights and regions connected to the volume of input images. Then, the activation function is performed to verify the neurons are correct or not by ReLU layers it will not change the dimension of the input images. Further, the pooling layer is utilized decrease the effect of noise among the extracted features. Finally, the higher level features are obtained by using Fully-Connected (FC) layer. In the proposed method, the pre-trained deep CNN models like VGG19 are utilized for feature

extraction. The VGG19 network includes 16 convolutional layers, 19 learnable weight layers and 3 FC layer with the soft-max function. The VGG19 model includes convolutional, pooling, ReLU, normalization, and FC layer. The convolutional layer extracts the local features from the images as shown in Eq. (1).

$$g_i^L = b_i^L + \sum_{j=1}^{m_1(L-1)} \psi_{i,j}^L \times h_j^{L-1} \quad (1)$$

where, g_i^L is the output layer of L, b_i^L is the base value, $\psi_{i.j}^l$ is the filter connection with i^{th} , j^{th} feature map and h_j is the output layer of $L-1$. The pooling layer extracts maximum responses from the lesser convolutional layer to reduce unwanted features and it solves the problem of over fitting which is explained through Eqs. (2-4).

$$m_i^L = m_1^{L-1} \quad (2)$$

$$m_2^L = \frac{m_2^{L-1} - F(L)}{S^L} + 1 \quad (3)$$

$$m_3^L = \frac{m_3^{L-1} - F(L)}{S^L} + 1 \quad (4)$$

where, S^L are the strides which are parameters of neural network that changes the movement of images, m_1^L , m_2^L and m_3^L are the filter of feature maps, the other layers like ReLU and FC are explained in Eqs. (5, 6).

$$Re_i^l = \max(h, h_i^{l-1} \quad (5)$$

$$FC_i^L = f\left(z_i^l\right) with\ z_i^l = \sum_{j=1}^{m_i(l-1)} \sum_{r=1}^{m_2^{l-1}} \sum_{s=1}^{m_3^{l-1}} w_{i,j,r,s}^l \left(FC_i^{i-1}\right)_{r,s} \quad (6)$$

where, Re_i^l is the ReLU layer, h is the output layer, FC_i^L L is the FC layer that follows pooling and convolutional layer that performs activation FC layer for deeper feature extractions.

5.2 PARTICAL SWARM OPTIMISATION

PSO, developed by Eberhart and Kennedy, is a swarm intelligence method for solving optimization problems]. PSO is inspired by the ability of flocks of birds, schools of fish, and herds of animals to adapt to their environment, find rich sources of food, and avoid predators by implementing "information sharing" approaches, hence, developing an evolutionary advantage. PSO is initialized with randomly generated population of particles (initial swarm) and a random velocity is assigned to each particle that propagates the particle in search space toward optima over a number of iterations. Each particle has a memory remembering best position attained by it in the past, which is called personal best position (P_{best}). Each particle has its P_{best} and the particle with the best value of fitness is called global best particle (G_{best}). Suppose that the search space is D dimensional, the ith particle of the population can be represented by a D-dimensional vector $(x_i^1, x_i^2 \ldots x_i^D)^T$. The velocity of this particle can be represented by another D-dimensional vector $(V_i^1, V_i^2 \ldots V_i^D)^T$. The previously best visited position of ith particle is denoted by P_i and the best particle in the swarm is denoted by P_g. The update of the particle's position is accomplished by the following two equations. Eq. (1) calculates a new velocity for each particle based on its previous velocity and Eq. (2) updates each particle's position in search space.

$$\text{Vidk+1=wVidk+c1r1pidkt−xidt+c2r2pgkt−xidkt} \quad (1)$$

$$\text{xidk+1t+1=xidkt+vidk+1t+1} \quad (2)$$

where k = iteration number, d = 1,2,3, …, D; i = 1,2,3, …, N; N = swarm size, w = inertia weight, which controls the momentum of

particle by weighing the contribution of previous velocity; c_1 and c_2 are positive constants called acceleration coefficients; r_1 and r_2 are random numbers uniformly distributed between [0,1].

5.3 SUPPORT VECTOR MACHINE

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Support Vector Machine (SVM) is an algorithm that works with nonlinear mapping which functions to transform initial training data into new, higher dimensions. In this new dimension, SVM will find the optimum linear hyperplane. By doing nonlinear mapping to higher dimensions, data from two classes will always be separated by a hyperplane. This method will find the hyperplane using support vectors and margins (Han et al., 2011). This method was first presented in 1992 by Vapnik, Boser, and Guyon at a Workshop on Computational Learning. SVM theory introduces a new strategy by finding the best hyperplane in the input space.

The principle of SVM was originally a linear classifier, but SVM was later developed to be able to work on non-linear problems by including the kernel. The development of this SVM stimulates research

interest in the field of pattern recognition in developing the potential capabilities of the SVM method both from a theoretical and application perspective.

Recently, SVM has been successfully applied in solving practical problems. The concept of SVM can be explained simply as an effort to find the best hyperplane which functions as a separator of two classes in the input space.

Classification problems can be interpreted as trying to find the line that separates the two groups. The best separator hyperplane between the two classes can be found by measuring the hyperplane margin and finding its maximum point. Margin is the distance between the hyperplane and the closest pattern from each class. This closest pattern is called a support vector. The effort to find the location of this hyperplane is at the core of the SVM learning process. In what is assumed it is given a set S of points $x_i \in R^n$ where i = 1, 2..... N. Each point xi belongs to one of the two classes and is thus labelled $Y_i \in \{1, -1\}$. Its purpose is to define a hyperplane equation that divides S leaving all points of the same class on the same side while maximizing the minimum distance between one of the two classes and the hyperplane. For this purpose some preliminary definitions are required (Pontil & Verri 1997).

First, the set S can be separated linearly if there is $w \in R^n$ and $b \in R$ such that:

$$x_i.w + b \geq +1 \text{ if } y_i = +1 \quad (1)$$

$$x_i.w + b \leq -1 \text{ if } y_i = -1 \quad (2)$$

w = weight vector perpendicular to the hyperplane (normal plane)

b = the position of the plane relative to the center of the coordinates

In simpler notation, the two inequalities above can be rewritten:

$$Y_i(w.x_i + b) \geq 1 \quad (3)$$

For i=1,2,…., N. The pair (w, b) represents the hyperplane of the following equation:

$$w.x + b = 0 \quad (4)$$

This is called the separating hyperplane. If denoted by which means w, the distance marked at the point of the separating hyperplane (w, b) is given by:

$$d_i = \frac{w.x_i + b}{\omega} \quad (5)$$

Combination of the above inequalities and equations for all $x_i \in S$, then

$$y_i d_i \geq \frac{1}{\omega} \quad (6)$$

Therefore, $1/\omega$ is the lower bound on the distance between the point and the separator hyperplane (w, b).

Second, given the separating hyperplane (w,b) for the linear separable set S, the Canonical Representation of the separating hyperplane is obtained by changing the size of the pair (w,b) into pairs (w',b') such that the distance of the closest point equals 1/w'. Through this definition is obtained

$$min_{x_i \epsilon S}\{y_i(w'.x_i + b)\} \quad (7)$$

Some of the commonly used Kernel functions are (Liu, 2018):

1. Linear kernel

$$K(x_i, x) = x_i^T.x \quad (8)$$

2. Polynomial kernel

$$K(x_i, x) = (\gamma.x_i^T.x + r)^d, \gamma > 0 \quad (9)$$

3. Radial Basic Function

$$K(x_i, x) = \exp\left(-\gamma |{x_i}^T - x|^2\right), \gamma > 0 \quad (10)$$

4. Sigmoid kernel

$$K(x_i, x) = \tanh(\gamma . {x_i}^T . x + r)^d, \gamma > 0 \quad (11)$$

In this case γ, r, and d are kernel parameters, and the C parameter is a penalty due to errors in the classification for each kernel.

5.4 PYTHON LIBRARIES

5.4.1 Keras

Keras is an open- source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the Tensorflow library.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. Keras has support for convolutional neural networks.

5.4.2 Tensorflow

Tensor Flow is a free and open- source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Tensor Flow serve as the core platform and library for machine learning. Tensor Flow's API'S use Keras to allow users to make their own machine learning models. In addition to build and to train our model, Tensor Flow can also help load the data to train the model.

5.4.3 ResNet-50

ResNet-50 is a convolutional neural network that is 50 layers deep. We can also load a pretrained version of the network trained on more than a million images from the imageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil and many animals.

5.4.4 KNeighborsClassifier

The K in the name of this classifier represents the k nearest neighbors, where k is an integer value specified by the user. Hence as the name suggests, this classifier implements learning based on the k nearest neighbors. The choice of the value of k is dependent on data.

5.4.5 NumPy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray; it provides a lot of supporting functions that make working with ndarray very easy.

5.4.6 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

5.4.7 OS

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The 'os' and 'os.path' modules include many functions to interact with the file system.

5.4.8 Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools

for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

5.4.9 Pandas

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

CHAPTER 6

IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION

6.1 WORKING PRINCIPLE OF THE

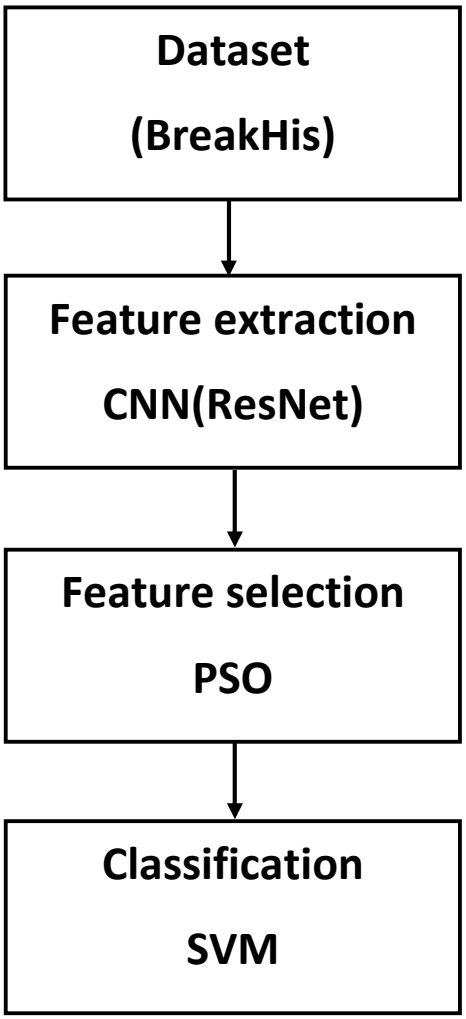


Figure 6.1 Working principle of the Particle Swarm Optimization using for the prediction of Breast Cancer

6.2 DATASET USED

The dataset consists of 2,77,524 50X50 pixel RGB digital image patches that were derived from 162 H&E-stained breast histopathology samples. These images are small patches that were extracted from digital images of breast tissue samples. The breast tissue contains many cells but only some of them are cancerous. Patches that are labelled “0” are benign (non-cancerous) cells and labelled “1” is malignant (cancerous) cells. In a malignant tumor, it includes 4 classes as Ductal Carcinoma, Lobular Carcinoma, Mucinous Carcinoma and Papillary Carcinoma. In benign tumor, it includes Adenosis, Fibroadenoma, Phyllodestumor, and Tubular-adenoma.

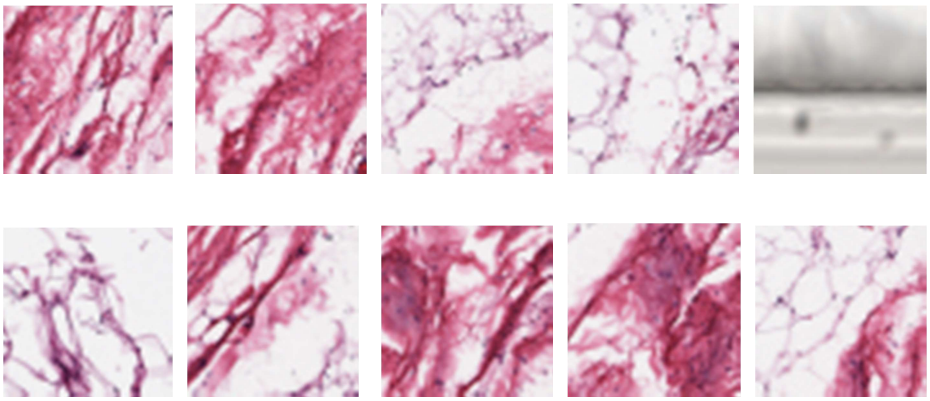


Figure 6.2 Benign images

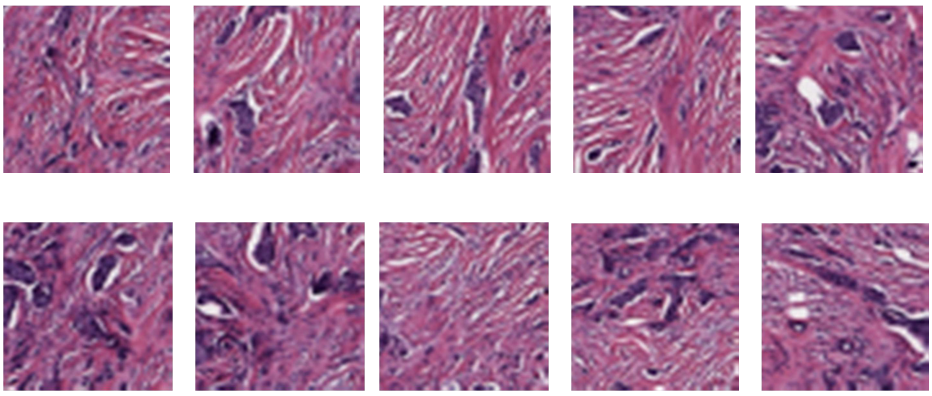


Figure 6.3 Malignant images

6.3 IMAGE PREPROCESSING

Most of the pixels in the image are redundant and do not contribute substantially to the intrinsic information of an image. This can be achieved by compression techniques. This project begins with the implementation by processing the images in the dataset. This is achieved with the help of the keras and tensorflow libraries in Python.

There are many other modules that can be used in this step e.g. MATLAB or other image processing libraries or software. This is necessary to remove redundancy from the input data which only contributes to the computational complexity of the network without providing any significant improvements in the result. The aspect ratio of the original slide is preserved since both the dimensions are reduced by a factor of 2, giving an image which is 1/4th in area that is of dimension 50×50 pixels.

6.4 FEATURE EXTRACTION

Feature extraction is a part of the dimensionality reduction process, in which, an initial set of the raw data is divided and reduced to more manageable groups. So when you want to process it will be easier. The most important characteristic of these large data sets is that they have a large number of variables. These variables require a lot of computing resources to process. So Feature extraction helps to get the best feature from those big data sets by selecting and combining variables into features, thus, effectively reducing the amount of data. These features are easy to process, but still able to describe the actual data set with accuracy and originality.

A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Feature extraction is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be

processed, while still accurately and completely describing the original data set.

For features extraction, Convolutional Neural Network (CNN) ResNet-50 architecture is used. It is fully connected layers from each model to allow the networks to consume images of an arbitrary size. In ResNet-50, the last convolutional layer consisting of 2048 channels is converted via GlobalAveragePooling into a one-dimensional feature vector with a length of 2048.

6.5 FEATURE SELECTION

Feature Selection is the method of reducing the input variable to the model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant features for the machine learning model based on the type of problem we are trying to solve. This can be done by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data.

For feature selection Particle Swarm Optimization is used. This method minimizes the number of features and reduces the error rate of classification by selecting optimal features.

6.6 CLASSIFICATION

Classification is the process of categorizing and labelling groups of pixels or vectors within an image based on specific rules. The categorization law can be devised using one or more spectral or textural characteristics. Two general methods of classification are ‘supervised’ and ‘unsupervised’.

In this project, the classifier is prepared to train on 80% of a breast cancer histopathology image dataset and 10% of the data for validation. Classification is done by Support Vector Machine (SVM).

CHAPTER 7

CODING

CHAPTER 7

CODING

In [1]:

```
# import the necessary packages

import os

# initialize the path to the *original* input directory of images

ORIG_INPUT_DATASET = "histopathology dataset"

# initialize the base path to the *new* directory that will contain

# our images after computing the training and testing split

BASE_PATH = "split data"

# derive the training, validation, and testing directories

TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])

VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])

TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])

# define the amount of data that will be used training

TRAIN_SPLIT = 0.8

# the amount of validation data will be a percentage of the

# *training* data

VAL_SPLIT = 0.1
```

In[2]:

```
# import the necessary packages
from imutils import paths
import random
import shutil
import os
# grab the paths to all input images in the original input directory
# and shuffle them
imagePaths = list(paths.list_images(ORIG_INPUT_DATASET))
random.seed(42)
random.shuffle(imagePaths)
# compute the training and testing split
i = int(len(imagePaths) * TRAIN_SPLIT)
trainPaths = imagePaths[:i]
testPaths = imagePaths[i:]
# we'll be using part of the training data for validation
i = int(len(trainPaths) * VAL_SPLIT)
valPaths = trainPaths[:i]
trainPaths = trainPaths[i:]
# define the datasets that we'll be building
datasets = [
        ("training", trainPaths, TRAIN_PATH),
        ("validation", valPaths, VAL_PATH),
        ("testing", testPaths, TEST_PATH)
```

```
]
# loop over the datasets
for (dType, imagePaths, baseOutput) in datasets:
        # show which data split we are creating
        print("[INFO] building '{}' split".format(dType))
        # if the output base output directory does not exist, create it
        if not os.path.exists(baseOutput):
                print("[INFO] 'creating {}' directory".format(baseOutput))
                os.makedirs(baseOutput)
        # loop over the input image paths
        for inputPath in imagePaths:
                # extract the filename of the input image and extract the
                # class label ("0" for "negative" and "1" for "positive")
                filename = inputPath.split(os.path.sep)[-1]
                label = filename[-5:-4]
                # build the path to the label directory
                labelPath = os.path.sep.join([baseOutput, label])
                # if the label output directory does not exist, create it
```

```
                    if not os.path.exists(labelPath):
                        print("[INFO] 'creating {}' directory".format(labelPath))
                        os.makedirs(labelPath)
                    # construct the path to the destination image and then copy
                    # the image itself
                    p = os.path.sep.join([labelPath, filename])
                    shutil.copy2(inputPath, p)
```

Out[2]:

```
[INFO] building 'training' split
[INFO] 'creating split data\training\0' directory
[INFO] 'creating split data\training\1' directory
[INFO] building 'validation' split
[INFO] 'creating split data\validation\1' directory
[INFO] 'creating split data\validation\0' directory
[INFO] building 'testing' split
[INFO] 'creating split data\testing\0' directory
[INFO] 'creating split data\testing\1' directory
```

FEATURE EXTRACTION

In [3]:

```
# Import the necessary packages
import os
# Initialize the path to the *original* input directory of images
ORIG_INPUT_DATASET = "dataset"
# Initialize the base path to the *new* directory that will contain
# Our images after computing the training and testing split
BASE_PATH = "TTdataset"
# Define the names of the training, testing, and validation
# Directories
TRAIN = "training"
TEST = "testing"
VAL = "validation"
# Initialize the list of class label names
CLASSES = ["0", "1"]
# Set the batch size
BATCH_SIZE = 32
# Initialize the label encoder file path and the output directory to
# Where the extracted features (in CSV file format) will be stored
LE_PATH = os.path.sep.join(["output", "le.cpickle"])
BASE_CSV_PATH = "output"
```

In [4]:

```
# Import the necessary packages
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from imutils import paths
import numpy as np
import pickle
import random
import os
# Load the ResNet50 network and initialize the label encoder
print("[INFO] loading network...")
model = ResNet50(weights="imagenet", include_top=False)
le = None
# Loop over the data splits
for split in (TRAIN, TEST, VAL):
        # Grab all image paths in the current split
        print("[INFO] processing '{} split'...".format(split))
        p = os.path.sep.join([BASE_PATH, split])
        imagePaths = list(paths.list_images(p))
```

```
# Randomly shuffle the image paths and then extract the
class
# Labels from the file paths
random.shuffle(imagePaths)
labels = [p.split(os.path.sep)[-2] for p in imagePaths]
# If the label encoder is None, create it
if le is None:
le = LabelEncoder()
le.fit(labels)
# Open the output CSV file for writing
csvPath = os.path.sep.join([BASE_CSV_PATH,
"{}.csv".format(split)])
csv = open(csvPath, "w")
  # loop over the images in batches
for (b, i) in enumerate(range(0, len(imagePaths),
BATCH_SIZE)):
# extract the batch of images and labels,
then initialize the
# List of actual images that will be
passed through the network
# For feature extraction
print("[INFO] processing batch
{}/{}".format(b + 1,
```

```
int(np.ceil(len(imagePaths) / float(BATCH_SIZE)))))
                batchPaths = imagePaths[i:i +
BATCH_SIZE]
                batchLabels = le.transform(labels[i:i +
BATCH_SIZE])
                batchImages = [ ]
                # loop over the images and labels in the
current batch
                for imagePath in batchPaths:
                        # load the input
image using the Keras helper utility
                        # while ensuring the
image is resized to 224x224 pixels
                        image =
load_img(imagePath, target_size=(224, 224))
                        image =
img_to_array(image)
                        # Preprocess the
image by (1) expanding the dimensions and
                        # (2) subtracting the
mean RGB pixel intensity from the
                        # ImageNet dataset
```

```
                                        image =
np.expand_dims(image, axis=0)
                                        image =
preprocess_input(image)
                                        # add the image to the
batch

            batchImages.append(image)
            # pass the images through the network and use the outputs as
                                # Our actual features, then reshape the
features into a
                                # Flattened volume
                                batchImages = np.vstack(batchImages)
                                features = model.predict(batchImages,
batch_size=BATCH_SIZE)
                                features =
features.reshape((features.shape[0], 7 * 7 * 2048))
                                # Loop over the class labels and extracted
features
                                for (label, vec) in zip(batchLabels,
features):
                                        # Construct a row that
exists of the class label and
                                        # Extracted features
```

```
                                        vec = ",".join([str(v)
for v in vec])

            csv.write("{},{}\n".format(label, vec))
            # close the CSV file
            csv.close()
# Serialize the label encoder to disk
f = open(LE_PATH, "wb")
f.write(pickle.dumps(le))
f.close()
```

Out [4]:

```
[INFO] loading network...
[INFO] processing 'training split'...
[INFO] processing batch 1/26
1/1    [==============================]    -    9s
9s/step
[INFO] processing batch 2/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 3/26
1/1    [==============================]    -    8s
8s/step
```

```
[INFO] processing batch 4/26
1/1    [==============================]    -    7s
7s/step
[INFO] processing batch 5/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 6/26
1/1    [==============================]    -    7s
7s/step
[INFO] processing batch 7/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 8/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 9/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 10/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 11/26
1/1    [==============================]    -    8s
8s/step
```

```
[INFO] processing batch 12/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 13/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 14/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 15/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 16/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 17/26
1/1    [==============================]    -    7s
7s/step
[INFO] processing batch 18/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 19/26
1/1    [==============================]    -    7s
7s/step
```

```
[INFO] processing batch 20/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 21/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 22/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 23/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 24/26
1/1    [==============================]    -    7s
7s/step
[INFO] processing batch 25/26
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 26/26
1/1    [==============================]    -    5s
5s/step
```

```
[INFO] processing 'testing split'...
[INFO] processing batch 1/8
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 2/8
1/1    [==============================]    -    7s
7s/step
[INFO] processing batch 3/8
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 4/8
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 5/8
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 6/8
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 7/8
1/1    [==============================]    -    8s
8s/step
```

```
[INFO] processing batch 8/8
1/1    [==============================]    -    1s
792ms/step
[INFO] processing 'validation split'...
[INFO] processing batch 1/3
1/1    [==============================]    -    7s
7s/step
[INFO] processing batch 2/3
1/1    [==============================]    -    8s
8s/step
[INFO] processing batch 3/3
1/1    [==============================]    -    6s
6s/step
```

FEATURE SELECTION

In [5]:

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier


# error rate
def error_rate(xtrain, ytrain, x, opts):
```

```
# parameters
k     = opts['k']
fold  = opts['fold']
xt    = fold['xt']
yt    = fold['yt']
xv    = fold['xv']
yv    = fold['yv']

# Number of instances
num_train = np.size(xt, 0)
num_valid = np.size(xv, 0)
# Define selected features
xtrain  = xt[:, x == 1]
ytrain  = yt.reshape(num_train)  # Solve bug
xvalid  = xv[:, x == 1]
yvalid  = yv.reshape(num_valid)  # Solve bug
# Training
mdl     = KNeighborsClassifier(n_neighbors = k)
mdl.fit(xtrain, ytrain)
# Prediction
ypred   = mdl.predict(xvalid)
acc     = np.sum(yvalid == ypred) / num_valid
error   = 1 - acc
```

```
    return error


# Error rate & Feature size
def Fun(xtrain, ytrain, x, opts):
    # Parameters
    alpha    = 0.99
    beta     = 1 - alpha
    # Original feature size
    max_feat = len(x)
    # Number of selected features
    num_feat = np.sum(x == 1)
    # Solve if no feature selected
    if num_feat == 0:
        cost  = 1
    else:
        # Get error rate
        error = error_rate(xtrain, ytrain, x, opts)
        # Objective function
        cost  = alpha * error + beta * (num_feat / max_feat)


    return cost
```

In [6]:

```
import numpy as np
from numpy.random import rand

def init_position(lb, ub, N, dim):
    X = np.zeros([N, dim], dtype='float')
    for i in range(N):
        for d in range(dim):
            X[i,d] = lb[0,d] + (ub[0,d] - lb[0,d]) * rand()

    return X


def init_velocity(lb, ub, N, dim):
    V    = np.zeros([N, dim], dtype='float')
    Vmax = np.zeros([1, dim], dtype='float')
    Vmin = np.zeros([1, dim], dtype='float')
    # Maximum & minimum velocity
    for d in range(dim):
        Vmax[0,d] = (ub[0,d] - lb[0,d]) / 2
        Vmin[0,d] = -Vmax[0,d]

    for i in range(N):
```

```
        for d in range(dim):
            V[i,d] = Vmin[0,d] + (Vmax[0,d] - Vmin[0,d]) * rand()

    return V, Vmax, Vmin


def binary_conversion(X, thres, N, dim):
    Xbin = np.zeros([N, dim], dtype='int')
    for i in range(N):
        for d in range(dim):
            if X[i,d] > thres:
                Xbin[i,d] = 1
            else:
                Xbin[i,d] = 0

    return Xbin


def boundary(x, lb, ub):
    if x < lb:
        x = lb
    if x > ub:
        x = ub
```

```
return x

def jfs(xtrain, ytrain, opts):

# Parameters

ub    = 1

lb    = 0

thres = 0.5

w     = 0.9    # inertia weight

c1    = 2      # acceleration factor

c2    = 2      # acceleration factor


N       = opts['N']

max_iter = opts['T']

if 'w' in opts:

    w    = opts['w']

if 'c1' in opts:

    c1   = opts['c1']

if 'c2' in opts:

    c2   = opts['c2']


# Dimension

dim = np.size(xtrain, 1)

if np.size(lb) == 1:

    ub = ub * np.ones([1, dim], dtype='float')

    lb = lb * np.ones([1, dim], dtype='float')
```

```
# Initialize position & velocity
X            = init_position(lb, ub, N, dim)
V, Vmax, Vmin = init_velocity(lb, ub, N, dim)

# Pre
fit   = np.zeros([N, 1], dtype='float')
Xgb   = np.zeros([1, dim], dtype='float')
fitG  = float('inf')
Xpb   = np.zeros([N, dim], dtype='float')
fitP  = float('inf') * np.ones([N, 1], dtype='float')
curve = np.zeros([1, max_iter], dtype='float')
t     = 0

while t < max_iter:
    # Binary conversion
    Xbin = binary_conversion(X, thres, N, dim)

    # Fitness
    for i in range(N):
        fit[i,0] = Fun(xtrain, ytrain, Xbin[i,:], opts)
        if fit[i,0] < fitP[i,0]:
            Xpb[i,:]  = X[i,:]
            fitP[i,0] = fit[i,0]
```

```
        if fitP[i,0] < fitG:
            Xgb[0,:]  = Xpb[i,:]
            fitG      = fitP[i,0]

    # Store result
    curve[0,t] = fitG.copy()
    print("Iteration:", t + 1)
    print("Best (PSO):", curve[0,t])
    t += 1

    for i in range(N):
        for d in range(dim):
            # Update velocity
            r1     = rand()
            r2     = rand()
            V[i,d] = w * V[i,d] + c1 * r1 * (Xpb[i,d] - X[i,d]) + c2 * r2 *
(Xgb[0,d] - X[i,d])
            # Boundary
            V[i,d] = boundary(V[i,d], Vmin[0,d], Vmax[0,d])
            # Update position
            X[i,d] = X[i,d] + V[i,d]
            # Boundary
            X[i,d] = boundary(X[i,d], lb[0,d], ub[0,d])
```

```
    # Best feature subset
    Gbin       = binary_conversion(Xgb, thres, 1, dim)
    Gbin       = Gbin.reshape(dim)
    pos        = np.asarray(range(0, dim))
    sel_index  = pos[Gbin == 1]
    num_feat   = len(sel_index)
    # Create dictionary
    pso_data = {'sf': sel_index, 'c': curve, 'nf': num_feat}

    return pso_data
```

In [7]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# load data
data  = pd.read_csv('E:/PROJECT/split csv/training.csv')
data  = data.values
feat  = np.asarray(data[:, 1:-1])
label = np.asarray(data[:, 0])
```

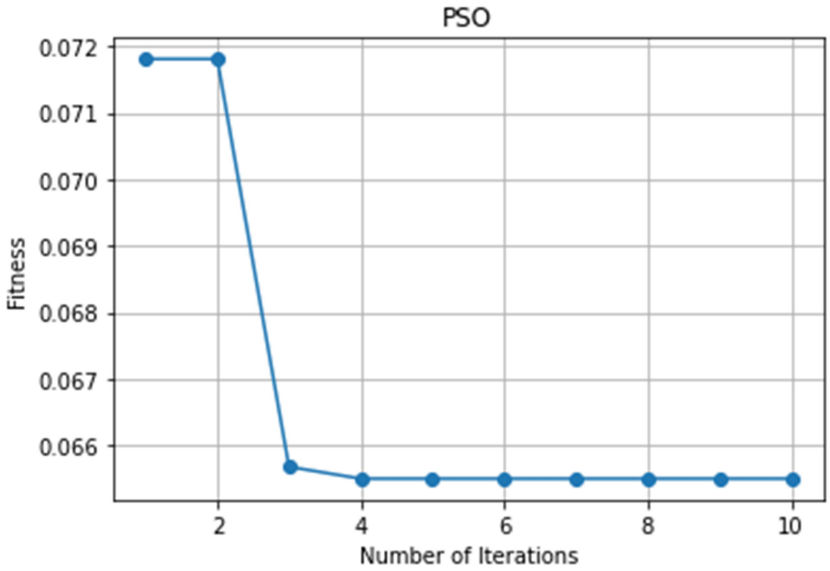
```
# split data into train & validation 80 --20)
xtrain, xtest, ytrain, ytest = train_test_split(feat, label, test_size=0.2,
stratify=label)
fold = {'xt':xtrain, 'yt':ytrain, 'xv':xtest, 'yv':ytest}

# parameter
k    = 5    # k-value in KNN
N    = 10    # number of particles
T    = 10   # maximum number of iterations
opts = {'k':k, 'fold':fold, 'N':N, 'T':T}

# perform feature selection
fmdl = jfs(feat, label, opts)
sf   = fmdl['sf']

# model with selected features
num_train = np.size(xtrain, 0)
num_valid = np.size(xtest, 0)
x_train   = xtrain[:, sf]
y_train   = ytrain.reshape(num_train)  # Solve bug
x_valid   = xtest[:, sf]
y_valid   = ytest.reshape(num_valid)  # Solve bug
```

```
# number of selected features
num_feat = fmdl['nf']
print("The number of features selected :", num_feat)


# plot convergence
curve   = fmdl['c']
curve   = curve.reshape(np.size(curve,1))
x       = np.arange(0, opts['T'], 1.0) + 1.0
fig, ax = plt.subplots()
ax.plot(x, curve, 'o-')
ax.set_xlabel('Number of Iterations')
ax.set_ylabel('Fitness')
ax.set_title('PSO')
ax.grid()
plt.show()
```

Out [7]:

```
Iteration: 1
Best (PSO): 0.07180777312125243
Iteration: 2
Best (PSO): 0.07180777312125243
Iteration: 3
Best (PSO): 0.0656798441152536
```

```
Iteration: 4

Best (PSO): 0.06550166950812462

Iteration: 5

Best (PSO): 0.06550166950812462

Iteration: 6

Best (PSO): 0.06550166950812462

Iteration: 7

Best (PSO): 0.06550166950812462

Iteration: 8

Best (PSO): 0.06550166950812462

Iteration: 9

Best (PSO): 0.06550166950812462

Iteration: 10

Best (PSO): 0.06550166950812462

The number of features selected: 47822
```



In [8]:

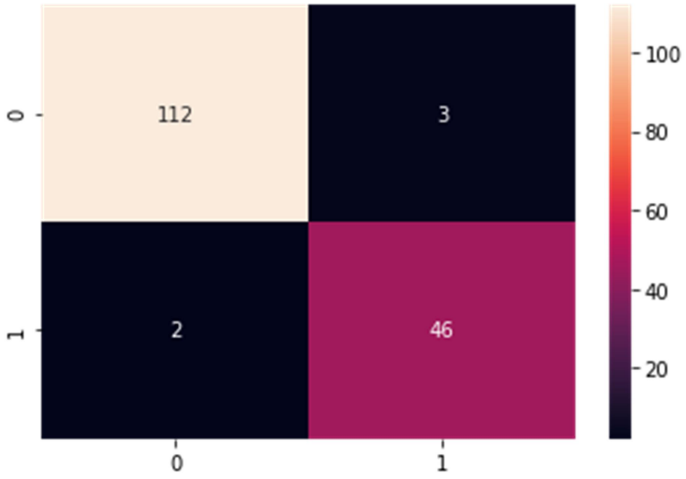
```
from sklearn import svm
from sklearn.metrics import confusion_matrix,precision_score, f1_score,
recall_score,plot_confusion_matrix
import seaborn as sns
from sklearn.pipeline import Pipeline

# model with selected features
num_train = np.size(xtrain, 0)
num_valid = np.size(xtest, 0)
x_train   = xtrain[:, sf]
y_train   = ytrain.reshape(num_train)  # Solve bug
x_valid   = xtest[:, sf]
y_valid   = ytest.reshape(num_valid)  # Solve bug

mdl = svm.SVC(kernel='linear')
mdl.fit(x_train, y_train)

# accuracy
y_pred    = mdl.predict(x_valid)
Acc= (np.sum(y_valid==y_pred) / num_valid)*100
print("Accuracy : %.3f" % Acc)
```

```
Precision=precision_score(y_valid,y_pred)*100
print("Precision : %.3f" % Precision)

Recall=recall_score(y_valid,y_pred)*100
print("Recall : %.3f" % Recall)

F1=f1_score(y_valid,y_pred)*100
print("F1-score : %.3f" % F1)

cm = confusion_matrix(y_valid, y_pred)
f = sns.heatmap(cm, annot=True, fmt='d')
total1=sum(sum(cm))

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])*100
print('Sensitivity : ', sensitivity )

specificity = cm[1,1]/(cm[1,0]+cm[1,1])*100
print('Specificity : ', specificity)
```

Out [8]:

```
Accuracy : 96.933
Precision : 93.878
Recall : 95.833
```

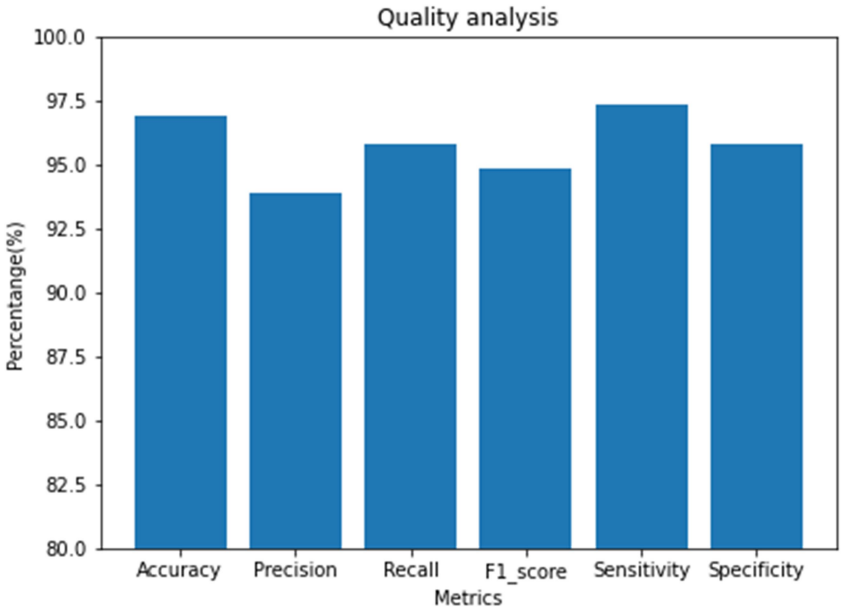
```
F1-score : 94.845

Sensitivity :  97.3913043478261

Specificity :  95.83333333333334
```



In [9]

```
x = ['Accuracy', 'Precision', 'Recall', 'F1_score','Sensitivity','Specificity']

y= [Acc,Precision,Recall,F1,sensitivity,specificity]

fig=plt.figure(figsize=(7,5))

plt.ylim(80,100)

plt.bar(x, y)

plt.title('Quality analysis')

plt.xlabel('Metrics')

plt.ylabel('Percentange(%)')

# Displaying the bar plot

plt.show()
```

Out[9]:


Quality analysis
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
Percentange(%)
Accuracy
Precision
Recall
F1_score
Sensitivity
Specificity
Metrics

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

Breast cancer causes great damage in women. Hence it is very important to prevent or take care, detect and get rid of the disease. In this paper, the histology images of the IDC patient tissues are used them to extract features using Convolutional Neural Network (CNN) and then feature selection is done using Particle Swarm Optimization (PSO) and then classification using Support Vector Machine (SVM). We have done a good job in predicting the classification of histology images with 97.56% accuracy.

The main limitation of this study is to use the secondary database like Kaggle, and future study should be done based on primary data for more accuracy of the results related to breast cancer identification.

8.2 FUTURE SCOPE

In future, we may combine various imaging technologies such as MRI, CT scan, ultrasound, and mammographic images, and determine their collective results. This technique is known as multimodel fusion. Problems stated above can again readily be solved by deep learning, and can be used to perform high quality research that might provide even better results.

REFERENCES

REFERENCES

DATASET LINK

https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images

RESEARCH PAPER

[1] Rachida Touami, Nacera Benamrane, University of Science and Technology of Oran Mohammed Boudiaf, Algeria "Microcalcification Detection in Mammograms Using Particle Swarm Optimization and Probabilistic Neural Network" – 2021

[2] Rizki Habibi, Medicom Academy of Informatics and Computers, Indonesia, "Svm Performance Optimization using PSO for Breast Cancer Classification" – 2021

[3] Rajesh Saturi, Parvataneni Premchand, University College of Engineering, Osmania University, Hyderabad, India "Multi-Objective Feature Selection Method by Using ACO and PSO Algorithm for Breast Cancer Detection" – 2021

[4] Badriya Al Maqbali, Department of Process Engineering, International Maritime College, Oman, "Hybrid Wolf Pack Algorithm and Particle Swarm Optimization Algorithm for Breast Cancer Diagnosis" - 2021

[5] Jesutofunmi Onaope Afolayan, Marion Olubunmi Adebiyi, Micheal Olaolu Arowolo & Ayodele Ariyo Adebiyi, Department of Computer Science, Landmark University, Omu-Aran, Nigeria, Chinmay Chakraborty, Department of Electronics and Communication Engineering, Birla Institute of Technology, Jharkhand, India "Breast Cancer Detection Using Particle Swarm Optimization and Decision Tree Machine Learning Technique." – 2022

[6] Nashat Alrefai & Othman Ibrahim, School of Computing, Faculty of Engineering, University Technology Malaysia (UTM), 81310, Johor Bahru, Johor, Malaysia, Othman Ibrahimm, Azman Hashim International Business School, University Technology Malaysia (UTM), 81310, Skudai, Johor, Malaysia, "Optimized feature selection method using particle swarm intelligence with ensemble learning for cancer classification based on microarray datasets." – 2022

[7] Anusha Papasani, Nagaraju Devarakonda & N. Bhagya Lakshmi, School of Computer Science and Engineering, VIT-AP University, Amaravati, AP, India, Zdzislaw Polkowski, Technical Sciences, Jan Wyzykowski University, Polkowice, Poland, Madhavi Thotakura Department of Computer Science and Engineering, SVECW, W.G. District, Kovvada, AP, India, "Feature Selection Using PSO Optimized-Framework with Machine Learning Classification System via Breast Cancer Survival Data" – 2022

[8] Yuhua Ma, Department of Oncology, Shanghai East Hospital, Tongji University School of Medicine, Shanghai 200120, China, Fei Liang, Department of Pathology, Karamay central Hosptial of XinJiang Karamay, Karamay, Xinjiang Uygur Autonomous Region 834000, China, Min Zhu, Chen Chen, Xiaoyi Lv, College of Software, Xinjiang University, No. 448, Northwest Road, Shayibake District, Urumqi, Xinjiang 830046, China,Chen Chen, College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China," FT-IR combined with PSO-CNN algorithm for rapid screening of cervical tumors" - 2022

REFERENCE LINK

- http://oaji.net/pdf.html?n=2021/3603-1629691308.pdf
- https://www.bircu-journal.com/index.php/birex/article/view/1499
- https://cys.cic.ipn.mx/ojs/index.php/CyS/article/view/3429

- https://link.springer.com/chapter/10.1007/978-981-16-9573-5_38
- https://link.springer.com/chapter/10.1007/978-981-16-8150-9_4
- https://www.igi-global.com/article/features-selection-study-for-breast-cancer-diagnosis-using-thermographic-images-genetic-algorithms-and-particle-swarm-optimization/277431
- https://link.springer.com/chapter/10.1007/978-981-15-4992-2_35
- https://publisher.resbee.org/admin/index.php/mr/article/view/29
- https://www.sciencedirect.com/science/article/pii/S2666827021000281
- https://aeuso.org/includes/files/articles/Vol12_Iss43_5101-5105_Breast_Cancer_Detection_using_Modif.pdf