CHAPTER 3
MACHINE VISION SYSTEM FOR THE CLASSIFICATION OF CHRONIC LIVER DISEASE USING TEXTURAL FEATURES AND TEXTURE SPECTRUM

Recent advances in digital imaging technology have greatly enhanced the interpretation of critical/pathology conditions from the 2-dimensional medical images. This has become realistic due to the existence of the computer aided diagnostic tool. This work enumerates on development of CAD tool for classification of chronic liver disease through the 2-D image acquired from ultrasonic device. The proposed work although can be applied to any medical images, the analysis has restricted with Ultrasonic images. Characterization of tissue through qualitative treatment leads to the detection of abnormality which is not feasible through qualitative visual inspection by the radiologist. Common liver diseases are the indicators of changes in tissue elasticity. A Feedforward neural network with backpropogation algorithm is applied for classifying the normal (vs) fatty and normal (vs) malignant yielding classification accuracy of 90%. Further multi-classification using Fuzzy Texture Feature Vectors is also performed which yields an overall classification accuracy of 95%. The system has been tested with several images for further validation.

3.1 INTRODUCTION

Diagnosis of any disease/disorder plays a crucial role for medical community. Due to the advent of information technology automated computer aided analysis (CAD) provides satisfactory solutions to the medical experts.
This work aims to develop a software engineering tool to detect and classify the various ultrasonic images of liver. Any abnormality in Ultrasonic images can be recognized / diagnosed by clinician. Often visual inspection of detecting the abnormality does not yield required diagnostic accuracy thereby leading to severe complications. Computer aided diagnosis provides practical solutions to such problems and in current practice; it is being accepted by clinician due to its inherent advantages such as automation efficiency in finding abnormal points. In CT-images of liver, disease such as carcinoma, hepatoma can be found. In ultrasound images disease such as fatty liver, liver cancer and hemangioma can be found.

3.2 OBJECTIVES

The main objectives of this work are as follows:

1. To acquire the ultrasound image for identifying liver diseases.
2. To apply suitable preprocessing technique for filtering the noise if present and to enhance the image for extracting the possible features.
3. To classify the normal and abnormal image using suitable classifier using textural features and Texture spectrum.

In this system, an automatic liver diseases diagnostic system for early detection of liver diseases, has classification accuracy of 95%. The advantage is its high accuracy and its computation simplicity. Ultrasound is a widely used medical imaging technique. Tissue characterization with ultrasound has become important topic since computer facilities have been available for the analysis of ultrasound signals. Automatic liver tissue characterizations from ultrasonic scans have been long the concern of many
researchers. Different techniques have been used ranging from processing the RF signals received by the transducer to using neural networks to analyze images based on image texture.

### 3.3 LITERATURE WORK


### 3.4 SYSTEM DESIGN, ANALYSIS AND INVESTIGATIONS

Classifying objects or regions is based on neural network analysis. Samples from ultrasound images are taken, grouped and numbered according to each disease case. The following image features are extracted from each image:

- Mean gray level
- Variance of gray level
- Skewness of gray level
- Texture Feature Vectors
- Fractal features

This work focuses on developing an automated software tool for classifying various ultrasound based liver images. The Ultrasonic Images of Liver are fed into the system which is automatically divided into blocks of
In the current system we divide the whole image into block by block instead of choosing the ROI. Hence full image is utilized in the proposed system.

3.5 ARTIFICIAL NEURAL NETWORKS

3.5.1 Introduction

Artificial Neural Networks (ANNS) are systems that are deliberately constructed to make use of some organizational principles resembling those of human brain. The key factors that distinguish Artificial Neural Networks from other computational techniques are

- ANNS are nonlinear; able to classify patterns and capture complex interactions among the input variables in the system.
- ANNS are adaptive: they can take data and learn from it. (i.e.) online training.
- ANNS can generalize: they can correctly process data that broadly resembles the data they were trained originally.
- ANN is a parallel-distributed information processing structure.

3.5.2 Artificial Neuron model

The model of an artificial neuron is shown in Figure 3.1. In this model the processing elements (neurons) compute the weighted sum of its inputs and outputs according to whether this weighted input sum is above or below a certain threshold \( \theta_k \). The externally applied bias has the effect of lowering the net input of the activation function

\[
y_k = f (u_k - \theta_k)
\]  

(3.1)
where \[ u_k = \sum w_{kj}x_j \] (3.2)

Here \( x_1, x_2, \ldots x_p \) are input signals and \( w_{k1}, w_{k2}, \ldots w_{kp} \) are interconnection weights of the neuron \( k \). \( u_k \) is the linearly combined output.

**Figure 3.1 Artificial Neuron Model**

Some commonly used activation functions are

i) **Sigmoid function:**

More than one functions are defined by this name. They differ in their output ranges. All have graphs similar to a stretched letter S. Hyperbolic tangent function is first of sigmoid function and has values in range \((-1,1)\). Second is a logistic function and has values in range \((0,1)\). One that best fits the required range has to be chosen. These two functions are respectively represented by

\[ S(x) = \text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \] (3.3)
b) \[ S(x) = \frac{1}{1 + e^{-x}} \]  \hspace{1cm} (3.4)

ii) **Step function**

Sometimes step function is used as threshold function. The function is 0 to start with and remains so to the left of some threshold value \( \phi \). A jump to 1 occurs for the value of function to the right of \( \phi \) and the function then remains at the level 1. It is shown in Figure 3.2.

![Step function](image)

**Figure 3.2 Step function**

It is mathematically represented as:

\[
S(x) = \begin{cases} 
0 & \text{for } x < \phi \\
1 & \text{for } x \geq \phi 
\end{cases}
\]  \hspace{1cm} (3.5)

As soon as the argument exceeds threshold value \( \phi \), the step function that gives a value from 0 to 1, in one step.

iii) **Ramp function**: Ramp function is described by the following equation:
\[ S(x) = 0 \quad \text{for } x < 0 \]
\[ S(x) = x \quad \text{for } x \geq 0 \]  

This is nothing but a step function that makes a jump from 0 to 1 not at once, but gradually gain in value along a straight line over a finite interval reaching from an initial 0 to final 1. Thus it is a piecewise linear approximation of a sigmoid.

Hard limiting transfer function is considered as special case of step function defined as

\[ S(x) = \begin{cases} 
0 & \text{if } x < 0 \\
1 & \text{if } x \geq 0 
\end{cases} \]  

A symmetrical hard limiting function, on the other hand is as

\[ S(x) = \begin{cases} 
-1 & \text{if } x < 0 \\
1 & \text{if } x \geq 0 
\end{cases} \]  

(iv) Linear function: Linear function is simply given by

\[ S(x) = \alpha x + \beta \]  

Where \( \alpha \) and \( \beta \) are constants. When \( \alpha = 1 \) the sum of inputs are simply added to bias (\( \beta \)). There are two types of linear function: (a) saturating linear and (b) symmetrical saturating linear function.

(a) Saturating linear transfer function is defined as

\[ S(x) = \begin{cases} 
0 & \text{if } x < 0 \\
\alpha x + \beta & \text{if } 0 < x \leq 1 \\
\beta & \text{if } x > 1 
\end{cases} \]
Symmetrical saturating linear function is defined as

\[ S(x) = \begin{cases} 
-\beta & \text{if } x < -1 \\
\alpha x + \beta & \text{if } -1 < x \leq 1 \\
\beta & \text{if } x > 1
\end{cases} \] (3.11)

Sometimes, a competitive transfer function has to be employed

### 3.6 NEURAL NETWORK CONNECTIONS

ANNS are weighted directed graphs in which neurons are nodes and directed edges (with weights) are connected between neuron outputs and neuron inputs.

Based on connection patterns ANNS are grouped as

- Feed-forward networks in which graphs have no loops.
- Recurrent or feedback networks in which loops occurs.

Feed-forward networks are static as they produce only one set of output values for a given input. These networks are memory-less in the sense that their response to an input is independent of the previous network state. In the simplest form of single layer feed-forward networks these is an input layer of source codes that project into an output layer of neurons. The simple layer feed forward networks are shown in Figure 3.3. The interconnection of several layers forms a multilayer feed-forward networks as shown in Figure 3.4. The layer between the input and output layer is called a hidden layer and its function is to intervene between the external input and output.
The hidden layer had no direct contact with the external environment. The radial basis function network is a special class of multilayer feed-forward networks. The hidden layer in this employs a radial basis function, such as a Gaussian kernel as the activation function.

Feedback networks that have closed loops are called recurrent networks. In single layer recurrent network the processing element output is feedback to itself or to other processing element or to both. In the multilayer recurrent network the neuron output can be directed back to the nodes in the preceding layers.
3.7 LEARNING IN ANN

Learning is not a unique process; there are different learning processes, each suitable to different species. In ANN the concepts of learning processes have been borrowed from the behaviorist’s lab and implemented in electronic circuitry.

Learning is a process by which neural network adapts itself to stimulus and eventually (after making the proper parameter adjustments to itself) it produces a desired response. Learning is also a continuous classification process of input stimuli. During the process of learning the network adjusts its parameters, the synaptic weights in response to and input stimulus so that its actual output response converges to the derived output response.

3.8 BACK PROPAGATION LEARNING ALGORITHM

Based on this algorithm, the networks learns a distributed associative map between the input and output layers. What makes this algorithm different than the others is the process by which the weights are calculated during the learning phase of the network. In general, difficulty with multilayer perceptions is calculating the weights of the hidden layers in an efficient way that results in the least (or zero) output error; the more hidden layers there are; the more difficult it becomes. To update the weights, one must calculate an error. At the output layer this error is easily measured; this is the difference between the actual and desired (target) outputs. At the hidden layers however, there is no direct observation of the error, hence some other technique must be used to calculate error, as this is the ultimate goal.

During the training session of the network, a pair of patterns is presented \((x_k, T_k)\), where \(x_k\) is the input pattern and \(d_k\) is the target or desired
pattern. The $x_k$ pattern causes output responses at each neuron in each layer and, hence actual output $O_k$ at the output layer. At the output layer, the difference between the actual and target outputs yields an error signal. This error signal depends on the values of the weights of the neurons in each layer. This error is minimized, and during this process new values for the weights are obtained. The speed and accuracy of the learning process (i.e., the process of updating the weights also depends on factor known as the learning rate.

The basis for this weight update algorithm is simply the gradient–descent method as used for simple perceptrons with differentiable units. For a given input–output pair ($x_k, T_k$) the back–propagation algorithm performs two phases of data flow. First, the input pattern ‘ax’ is propagated from the input layer to the output layer and, as a result of this forward flow of data, it produces an actual output $y_k$. Then the error signals resulting from the difference between $T_k$ and $O_k$ are back–propagated from the output layer to the previous layers for them to update their weights.

![Figure 3.4 Back propagation network](image-url)
Back propagation Learning

Consider the network as shown in the Figure 3.5 where the letters L, H, O denote input, hidden and output neurons.

Figure 3.5 Simplified multilayer feed forward back propagation network

Consider a problem in which an “nset” of l inputs and the corresponding “nset” of “n” output data is given as shown in the Table 3.1
Table 3.1 “nset” of input and output data

<table>
<thead>
<tr>
<th>No.</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$I_1$ $I_2$ .... $I_n$</td>
<td>$O_1$ $O_2$ .... $O_n$</td>
</tr>
<tr>
<td>2</td>
<td>0.3 0.4..... 0.8</td>
<td>0.1 0.56..... 0.82</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input Layer Computation

\[ I_{Hp} = V_{1p}o_{11} + V_{2p}o_{12} + ... + V_{1p}o_{11} \]  (3.12)

\((p=1, 2, 3, ....m)\)

Denoting weight matrix or connectivity matrix between input neurons and hidden neurons as \((v)\), we can get an input to the hidden neurons as:

\[ \{I\}_H = [v]^T \{o\}_I \]

Hidden Layer computation

Considering sigmoidal function or squashed-S function, the output of the \(P^{th}\) hidden neuron is given by

\[ O_{Hp} = \frac{1}{1 + e^{-e_{Hp}(I_{Hp} - e_{Hp})}} \]  (3.13)

Where \(O_{Hp}\) is the output of the \(P^{th}\) hidden neuron, \(I_{Hp}\) is the input of the \(P^{th}\) hidden neuron, and \(e_{Hp}\) is the threshold of the \(P^{th}\) neuron. A non-
Zero threshold neuron is computationally equivalent to an input that is always held at -1 and the non-zero threshold becomes the connecting weight values.

\[ O_{hi} = \begin{cases} 1 & \text{if } \sum W_{hi} - \theta_{hi} > 0 \\ 0 & \text{otherwise} \end{cases} \]

Figure 3.6 Treating threshold in hidden layer

Now output to the hidden neuron is given by

\[
\{O\}_H = \frac{1}{1 + e^{-\lambda(I_{HP} - \theta_{HP})}}
\]

Treating each component of the input of the hidden neuron separately, we get the output of the hidden neuron

The input to the output neuron is the weighted sum of the outputs of the hidden neurons to get \( I_{Oq} \) (i.e the input to the \( q \)th output neuron)

\[
I_{Oq} = W_{1q} O_{H1} + W_{2q} O_{H2} + \ldots + W_{mq} O_{Hm}
\]

\((q=1,2,3,\ldots,n)\)
Output layer computation

Considering sigmoidal function, the output of the $q^{th}$ output neuron is given by

$$O_{oq} = \frac{1}{1 + e^{-\phi(I_{oq} - \theta_{oq})}}$$  \hspace{1cm} (3.16)

Where $O_{pq}$ is a output of the $q^{th}$ output neuron, $I_{oq}$ is the input to the $q^{th}$ output neuron, and $\theta_{oq}$ is the threshold of the $q^{th}$ neuron. This threshold may also be tackled again by considering extra $O^{th}$ neuron in the hidden layer with output of -1 and the threshold value $\theta_{oq}$ becomes the connecting weight value.

![Figure 3.7 Treating threshold in output layer](image)

Figure 3.7 Treating threshold in output layer
Calculation of Error

Consider the Figure 3.6 and 3.7. Considering any $r^{th}$ output neuron and for the training example we have calculated the output ‘$o$’ for which the target output ‘$t$’ is given in Table 3.1.

Hence, the error norm in output for the $r^{th}$ output neuron is given by

$$E_r = \frac{1}{2} e_r^2 = \frac{1}{2} (T - O)^2$$

(3.17)

Where $E_r = 1/2$ second norm of the error in the $r^{th}$ neuron (‘$e_r$’), for the given training pattern. The square of the error is considered since irrespective of whether error is positive or negative, we consider only absolute values. The Euclidean norm of error $E_1$ for the first training pattern is given by

$$E_1 = \frac{1}{2} \sum_{j} (T_{or} - O_{or})^2$$

(3.18)

Equation gives the error function in one training pattern. If we use the same technique for the entire training pattern, we get

$$E(v,w) = \sum_{i} E(v,w_i)$$

(3.19)

where $E(v,w)$ is the error function depending on the m (1+n) weights of [w] and [v]. This is a classic type of optimization problem. For such problems, an objective function or cost function is usually defined to be maximized or minimized with respect to a set of parameters.
3.8.1 Training of neural network

The synaptic weighting and aggregation operations performed by the synapses and soma respectively, provide a ‘similarity measure’ between the input vector \( I \) and the synaptic weights \([v]\) and \([w]\) (accumulation knowledge base). When a new input pattern that is significantly different form the previously learned pattern is presented to the neural network, the similarity between this input and the existing knowledge base is small. As the neural network learns this new pattern, by changing the strengths of synaptic weights, the distance between the new information and accumulated knowledge decreases. In other words, the purpose of learning is to make “\( w \) and \( v \)” very similar to given pattern \( I \).

The neural network structures undergo ‘learning procedures’ during which synaptic weights \( W \) and \( V \) are adjusted. Algorithms for determining the connection strengths to ensure learning are called ‘learning rules’. The objective of learning rules depends upon applications. In classification and functional approximation problems, however, there has been no generalization as to how a neural network can be trained.

3.8.2 Back Propagation Algorithm

It is already explained the benefit of the middle- hidden layer in an artificial neural network. We understand that the hidden layer allows ANN to develop its own internal representation of this mapping. Such a rich and complex internal representation capability allows the hierarchical network to learn any mapping and not just linearly separable ones. Let us consider the three- layer network with input layer having \( l \) nodes. Hidden layer having \( m \) nodes, and an output layer with \( n \) nodes. We consider sigmoidal function for activation function for the hidden and output layer and linear
activation function for input layer. The number of neurons in the hidden layer may be chosen to lie between 1 and 21.

Algorithm (Back propagation learning Algorithm)

**Step 1:** Normalize the inputs and outputs with respect to their maximum values. It is proved that the neural networks work better if input and outputs lie between 0-1.

**Step 2** Assume the number of neurons in the hidden layer to lie between 1 < m < 21

**Step 3** \([v]\) represents the weights of synapses connecting input neurons and hidden neurons and \([w]\) represents weights of synapses connecting hidden neurons and output neurons. Initialize the weights to small random values usually from -1 to 1. for general problems, \(\lambda\) can be taken as zero.

\[
[v]^0 = \text{random weights} \\
[w]^0 = \text{random weights} \\
[\Delta v]^0 = [\Delta w]^0 = [0]
\]

**Step 4:** For the training data, present one set of inputs and outputs. Present the pattern to the input layer \(\{I\}_1\) as inputs to the input layer by using linear activation functions.

**Step 5:** Compute the inputs to the hidden layer by multiplying corresponding weights of synapses.

**Step 6:** Let the hidden layer units evaluate the output using the sigmoidal function.
Step 7: Compute the inputs to the output layer by multiplying corresponding weights of synapses.

Step 8: Let the output layer units evaluate the output using sigmoidal function.

Step 9: Calculate the error and the difference between the network output and the desired output as for the $i^{th}$ training set as

$$E^p = \sqrt{\frac{\sum (T_j - O_j)^2}{n}}$$  \hspace{1cm} (3.20)

Step 10: Find error rate as

$$\text{Error rate} = \frac{\sum E_p}{n_{set}}$$  \hspace{1cm} (3.21)

Step 11: Repeat steps 4-10 until the convergence in the error rate is less than the tolerance value.

The process of computing the gradient and adjusting the weights is repeated until a minimum error is found. In practice, one develops an algorithm termination criterion so that the algorithm does not continue this iterative process forever.

In summary the error back - propagation algorithm can be outlined as

Step 1: Initialize all weights to small random values.

Step 2: Choose an input-output training pair.
The salient features of neural networks are:

i. They are universal approximates.

ii. They have a simple topological structure.

iii. They can implement fast learning algorithms because of locally tuned neurons.

3.9 TEXTURE ANALYSIS

Texture analysis is a method that differentiates image texture in terms of the feature calculated from the image. It is based on two approaches:

- Statistical approach
- Transform method
**Statistical Approach**

This approach describes textures in a form suitable for statistical pattern i.e.

- Mean
- Variance

Texture Features Vectors are used to classify the liver images. These features can be extracted from Texture Spectrum.

### 3.10 TEXTURE SPECTRUM

The basic idea of the texture spectrum approach is to transform an image using the texture units and to characterize the global texture of an image by its texture spectrum. The texture spectrum can be defined as the occurrence frequency function of all the texture units. The global texture characteristics of the image, are maintained to the corresponding texture spectrum and the resulting image, confirming that the textural spectrum approach can be used with success to texture characterization and texture classification.

Brief description of the principles used for the estimation of the texture spectrum: The methodology is known in general as a filtering approach of the texture but it will be used here as a preprocessing procedure for the extraction of the textural features. The texture will be faced as an interwoven distribution of the intensities of the pixels. A statistical approach for the description of the texture properties seems more reasonable compared with a structural one as given by Julesz (1986). A complete definition of the texture spectrum employs the determination of values as the texture unit, the
texture unit number and finally the texture spectrum. For details, kindly refer Appendix 1.

**The Mean Gray Level**

It is the brightness or echogenicity of the texture, and is expressed mathematically as the average of the brightness of all Pixels of the sampled Ultrasound photo. Equation (1) is used to calculate the mean gray level.

\[ g_{\text{ave}} = \frac{1}{N} \sum_{(i,j) \in R} g(i,j) \]  

(3.22)

Where \( g(i,j) \): Gray level of pixel \( i, j \) and \( R \): Region-of-Interest, selected by the operator.

**Standard Deviation**

It is defined as

\[ V_g = \left( \frac{1}{N} \right) \sum_{(i,j) \in R} (g(i,j) - g_{\text{ave}})^2 \]  

(3.23)

Where \( g(i,j) \): Gray level of pixel \( i, j \) and \( R \): Region-of-interest; selected by the operator.

**Transform methods**

Transform methods of texture analysis used in our system is Gabor Wavelet

3.11 **GABOR WAVELET**

Gabor wavelet allows study of the spatial distribution of texture and useful for segmentation and local/pixelwise classification. It enables analysis
of “texture in context”. The advantages of using gabor wavelet are to reduce noise in an image, to minimize classification error, and to isolate texture according to particular frequencies and orientations.

3.12 CLASSIFICATION

Two types of classifier are used. Binary classifier is used to find whether the image is normal or abnormal. Multiple Classifier is used to find the type of disease. Texture Feature Vectors are used for the classification.

3.13 EXPERIMENTAL RESULTS

An image is fed into system, automatic division of block of size 64x64 is performed. The generated block is passed into Gabor filter and the following observation is made.

3.14 OBSERVATION

It is found that the Mean Gray Level and Variance of malignant liver is found to be higher than normal and fatty liver. It is found from the following graph in next section malignant are of highest echogenicity. Fatty Liver has lowest Mean gray level.

3.15 FIGURES

In Figures 3.8, 3.9 and 3.10, the features of three types of liver are shown.
Figure 3.8  Normal Liver Features

Figure 3.9  Fatty Liver Features

Figure 3.10  Malignant Liver Features
3.16 CONCLUSION AND FUTURE ENHANCEMENT

A Computer aided diagnosis tool is developed in order to detect and classify the disease of the ultrasonic image of liver. The system is an automatic diagnostic system which can be used not only by the clinician but also persons who are not related to medical field. The wavelet transform system produces better results in extraction of features. The future enhancement can be done to increase the accuracy of the system and including more feature extraction methods. Advanced Texture Spectrum also provides Texture Feature Vectors that are vital for multilevel classification. It can be extended to classify images using various sizes of images. The defect detection efficiency is 95%.