CHAPTER 3

RESEARCH WORK DONE IN MACRO-MACHINING

3.1 PREDICTION OF $R_a$ IN CNC-TURNING USING ANN-BP

In this experimental work, average gray level ($G_a$) of the machined surface was used to predict the average surface roughness ($R_a$) through machine vision in turning operation.

3.1.1 Experimental set-up for the measurement of surface image of a work-piece

The surface image of a work piece is grabbed using frame buffers through high resolution digital camera and sent to the personal computer. The Schematic diagram of machine vision system is shown in Figure 3.1.

![Figure 3.1 Schematic diagram of machine vision system](image)
The captured image is processed in the computer using Matlab Software (version7). The image is digitized into a rectangular array of intensity values. Each array element called “pixel” corresponds to the mean intensity in a small rectangular area of the original image. These values are referred as the gray levels of the corresponding pixels. Each pixel corresponds to a gray intensity level. The arithmetic average of the $G_a$ can be expressed as:

$$G_a = \frac{1}{n} \sum_{i=1}^{n} |g_i|$$  \hspace{1cm} (3.1)

Where $g_i$ is the gray level of the surface image and $n$ is the total number of pixels. The photographic view of the Rapid I machine vision system used for the image acquisition and subsequent processing is shown in Figure 3.2.

Figure 3.2 The photographic view of Rapid I machine vision system
Specifications of Machine Vision System (Rapid I)

Magnification Range : 11X to 67X

High Resolution Digital Cameras
Imaging : upto 1600 x 1200 pixel
Lighting System : 5 Zones
Built in Rapid 1 software : A 2X magnifying lens

The grabbed Images of different jobs using machine vision are shown in Figure 3.3.

Figure 3.3 Grabbed images of different jobs using machine vision
Compared to the human visual system most technical vision systems appear crude and simple. They are almost invariably based on monochrome pictures and in most of the machine vision systems special lighting and viewing techniques are employed to simplify the task of processing the obtained images.

3.1.2 Experimental Procedure

The experimental works were carried out in a CNC turning centre TN S 25 600 ABC under different cutting conditions to investigate $R_a$. EN-24 steel was used as the workpiece material and tungsten carbide as the cutting tool.

- **Specifications of CNC turning centre**

  Mack : PMT machine Ltd, Pune

  X-axis travel : 170 mm

  Z-axis travel : 400 mm

  Tool size : 25 x 25mm

  System : 80 Fanuc O, mute TC

  Spindle speed range : 40-4500 rpm

  Turret : 8 Stations

  Spindle motor power : 5.5 kW

  Maximum length between centers : 400mm
Machine dimensions:

L x W x H (approx) : 2500 x 1500 x 1650 mm

Weight (approx.) : 2500 kg

The selected machining parameter levels in turning are presented in Table 3.1.

**Table 3.1 Selected machining parameter levels in turning**

<table>
<thead>
<tr>
<th>Factors</th>
<th>Low(-)</th>
<th>Medium(0)</th>
<th>High(+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cutting Speed ((V)) in m/min</td>
<td>42</td>
<td>132</td>
<td>200</td>
</tr>
<tr>
<td>Feed rate ((F)) in mm/rev</td>
<td>0.05</td>
<td>0.16</td>
<td>0.33</td>
</tr>
<tr>
<td>Depth of cut ((D)) in mm</td>
<td>0.5</td>
<td>1.5</td>
<td>2.5</td>
</tr>
</tbody>
</table>

A \(3^3\) factorial design was used in order to get the output results, uniformly distributed all over the ranges of the input parameters. 27 experiments have been conducted for the various sets of cutting conditions i.e. cutting speed \((V)\), feed rate \((F)\) and depth of cut \((D)\) as shown in Table 3.2.
Table 3.2  Experimental turning data sets for training the network models

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Cutting Speed (V) m/min</th>
<th>Feed rate (F) mm/rev</th>
<th>Depth of cut (D) mm</th>
<th>Average gray level (G_a)</th>
<th>Surface roughness (R_a) μm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42</td>
<td>0.05</td>
<td>0.5</td>
<td>10.31</td>
<td>6.713</td>
</tr>
<tr>
<td>2</td>
<td>132</td>
<td>0.05</td>
<td>0.5</td>
<td>8.39</td>
<td>0.882</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>0.05</td>
<td>0.5</td>
<td>7.78</td>
<td>0.664</td>
</tr>
<tr>
<td>4</td>
<td>42</td>
<td>0.05</td>
<td>1.5</td>
<td>10.33</td>
<td>7.176</td>
</tr>
<tr>
<td>5</td>
<td>132</td>
<td>0.05</td>
<td>1.5</td>
<td>8.41</td>
<td>0.887</td>
</tr>
<tr>
<td>6</td>
<td>200</td>
<td>0.05</td>
<td>1.5</td>
<td>7.8</td>
<td>0.825</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>0.05</td>
<td>2.5</td>
<td>10.34</td>
<td>8.54</td>
</tr>
<tr>
<td>8</td>
<td>132</td>
<td>0.05</td>
<td>2.5</td>
<td>8.41</td>
<td>2.30</td>
</tr>
<tr>
<td>9</td>
<td>200</td>
<td>0.05</td>
<td>2.5</td>
<td>7.81</td>
<td>2.265</td>
</tr>
<tr>
<td>10</td>
<td>42</td>
<td>0.16</td>
<td>0.5</td>
<td>28.52</td>
<td>9.0</td>
</tr>
<tr>
<td>11</td>
<td>132</td>
<td>0.16</td>
<td>0.5</td>
<td>23.21</td>
<td>7.02</td>
</tr>
<tr>
<td>12</td>
<td>200</td>
<td>0.16</td>
<td>0.5</td>
<td>21.54</td>
<td>6.713</td>
</tr>
<tr>
<td>13</td>
<td>42</td>
<td>0.16</td>
<td>1.5</td>
<td>28.58</td>
<td>9.20</td>
</tr>
<tr>
<td>14</td>
<td>132</td>
<td>0.16</td>
<td>1.5</td>
<td>23.26</td>
<td>8.70</td>
</tr>
<tr>
<td>15</td>
<td>200</td>
<td>0.16</td>
<td>1.5</td>
<td>21.58</td>
<td>7.273</td>
</tr>
<tr>
<td>16</td>
<td>42</td>
<td>0.16</td>
<td>2.5</td>
<td>28.61</td>
<td>9.62</td>
</tr>
<tr>
<td>17</td>
<td>132</td>
<td>0.16</td>
<td>2.5</td>
<td>23.28</td>
<td>8.934</td>
</tr>
<tr>
<td>18</td>
<td>200</td>
<td>0.16</td>
<td>2.5</td>
<td>21.6</td>
<td>7.81</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
<td>0.33</td>
<td>0.5</td>
<td>53.73</td>
<td>16.461</td>
</tr>
<tr>
<td>20</td>
<td>132</td>
<td>0.33</td>
<td>0.5</td>
<td>43.72</td>
<td>12.41</td>
</tr>
<tr>
<td>21</td>
<td>200</td>
<td>0.33</td>
<td>0.5</td>
<td>40.57</td>
<td>10.50</td>
</tr>
<tr>
<td>22</td>
<td>42</td>
<td>0.33</td>
<td>1.5</td>
<td>53.85</td>
<td>17.98</td>
</tr>
<tr>
<td>23</td>
<td>132</td>
<td>0.33</td>
<td>1.5</td>
<td>43.82</td>
<td>13.121</td>
</tr>
<tr>
<td>24</td>
<td>200</td>
<td>0.33</td>
<td>1.5</td>
<td>40.66</td>
<td>11.71</td>
</tr>
<tr>
<td>25</td>
<td>42</td>
<td>0.33</td>
<td>2.5</td>
<td>53.91</td>
<td>21.32</td>
</tr>
<tr>
<td>26</td>
<td>132</td>
<td>0.33</td>
<td>2.5</td>
<td>43.87</td>
<td>17.631</td>
</tr>
<tr>
<td>27</td>
<td>200</td>
<td>0.33</td>
<td>2.5</td>
<td>40.7</td>
<td>12.93</td>
</tr>
</tbody>
</table>
3.1.3 Machining Parameters

- **Cutting speed (V)**

  It is the travel of a point on the cutting edge relative to the surface of the cut in unit time in the process of accomplishing the primary cutting motion.

  The cutting speed (V) is given by the relation

  \[ V = \frac{\pi dN}{1000} \text{ m/min} \]  \hspace{1cm} (3.2)

  Where, \( d \) is the diameter of the work piece in mm and \( N \) is speed of the spindle in rpm.

- **Feed rate (F)**

  The feed or more precisely rate of feed is the amount or tool advancement per revolution of job parallel to the surface being machined. \( F \) is expressed as the distance moved by the tool in one minute or it is expressed in millimeter per revolution. Feed rate depends on depth of cut, rigidity of cutting tool and type of cutters and heavy machine tools. Low feed rate is used for finishing cuts, hard work materials and weak cutters. Normally feed rate varies from 0.1 to 1.5mm for medium cuts. In lathe work, distinction is made between longitudinal \( F \) when the tool travels in a direction parallel to the work axis and cross \( F \) when the tool travels in a direction perpendicular to the work axis.

- **Depth of cut (D)**

  It is the thickness of the layer of metal removed in one cut or pass measured perpendicular to the machined surface. The \( D \) is always perpendicular to the direction of the feed motion.
3.1.4 Surface roughness measurement ($R_a$)

The $R_a$ of the turned work piece was measured by using SE-1100 portable Surfcorder, within a sampling length of 8 mm and with a measurement speed of 0.5 mm/s. The $R_a$ is the arithmetic average of the absolute value of the heights of roughness irregularities from the mean value measured:

$$R_a = \frac{1}{n} \sum_{i=1}^{n} |y_i|$$  \hspace{1cm} (3.3)

Where $y_i$ is the height of roughness irregularities from the mean value and $n$ is the number of sampling data. The machining parameters $V$, $F$, $D$ and $G_a$ were taken as input parameters and the response parameter $R_a$ was considered as output response.

3.1.5 Methodology of ANN trained by Back propagation (BP) algorithm

An ANN is a parallel, distributed information processing structure that mimics the human brain to learn from examples or mistakes (Freeman and Skapura 1991). Neural networks, based on their biological counterparts, attempt to model the parallel, distributed nature of processing in the human brain. Since this concept was introduced in 1950s, ANN technology has been adapted in many applications that are complex and non-linear in nature, with an unknown and hard-to-identify algorithm (Lippmann 1999).

The mathematical model of an artificial neuron's behavior is the simplification of the biological brain neuron shown in Figure 3.4.
The structure of a neural network is made up of the interconnection architecture between the neurodes, the function that will determine whether or not the neuron will fire, and the rules that determine the changes in the importance (weighting) of the neurodes inputs. A neuron is basically an extremely simple processing element that has a number of input signals and only one output signal. Each input signal $X_i$ has an associated weight $W_i$, so that the effective input to the neuron is the weighted total input (or the sum of all of the products of each input and its assigned weight). If the input is greater than this threshold, the neuron will fire or generate an output signal. Otherwise, the processing element will not fire and no output will be generated.

In this work, multi-layer perceptron model with each layer consisting of a number of computing neurons has been used. A perceptron is nothing but a computing unit or neuron. The activation function used in both the hidden layer and output layer is a non-linear function. For the input layer, no activation function is used since no computation involved in the input layer. The activation function $F(x)$ is a non-linear function and is given by:
\[ F(x) = \frac{1}{1 + e^{x}} \]  

(3.4)

Where \( F(x) \) is differentiable and

\[ x = \sum_{i=1}^{m} W_{ij} U_i + \text{threshold}. \]

Where \( W \) is the weight, \( U \) is the input node value, \( i = 1 \) to \( m \), \( j = 1 \) to \( n \) and \( m \) and \( n \) are the number of input nodes and hidden nodes respectively.

Therefore, output of a neuron in the successive layer is given by:

\[ \text{Output, } y_i = \frac{1}{1 + e^{\left( \sum_{i=1}^{m} W_{ij} U_i + \text{threshold} \right)}} \]  

(3.5)

The back-propagation is used as learning procedure for a multi-layer perception network. The algorithm makes it possible to propagate error from the output layer to the input layer and correct the weight vectors, which will result in minimum error. The back-propagation algorithm minimizes the squares of the differences between actual output and desired output units and for all training pairs.

3.1.6 Neural Network for the prediction of \( R_a \)

An A.N.N. model has been developed for the prediction of \( R_a \) using Matlab software. The input parameters are taken as \( V, F, D, G_a \) and the output parameter is \( R_a \). Twenty-seven experimental readings were carried out to train the neural network as shown in Table 3.2. The structure of ANN-BP neural network is shown in Figure 3.5.
A suitable network topology is selected based on the trial and error approach. After training the different topologies of network, the optimal network 4-5-5-1 is selected to train the neural network. The schematic flow chart for $R_a$ prediction using ANN-BP is shown in Figure 3.6.
The data obtained from the input parameters are initialized for training the network. The simulation parameters are taken for training and testing the network and 0.001 allowable error is chosen to train the network. The Figure 3.7 shows the training performance of ANN-BP with iteration number. This network is trained till the error is 0.001.

![Figure 3.7 Training Performance of ANN-BP with Iteration Number](image)

Figure 3.7 Training Performance of ANN-BP with Iteration Number

After the training parameters selection, now the network is trained with suitable, allowable error and iterations. Finally the trained network is simulated with new testing inputs for the predictions of surface roughness.
3.2 PREDICTION OF $R_a$ IN CNC-TURNING USING ANFIS

In this part of study, An ANFIS model is used to predict the average surface roughness ($R_a$) in turning operation. This model can effectively predict the process output i.e. $R_a$, when the process inputs namely $V$, $F$, $D$ and $G_a$ of the machined surface are given, due to integration of fuzzy logic and neural network. The neuro-fuzzy systems have potential to capture the benefits of both the fascinating fields into a single frame-work. This system eliminates the basic problem in fuzzy system design (i.e. obtaining a set of fuzzy if-then rules) by effectively using the learning capability of an ANN for automatic fuzzy if-then rules generation. While fuzzy logic uses approximate human reasoning in knowledge-based systems, the neural networks aim at pattern recognition, optimization and decision making. Fuzzy adaptive networks are capable of providing both learning ability and tolerance for imprecision, uncertainty and vagueness. The neuro fuzzy hybrid system combines the advantages of fuzzy logic system, which deals with explicit knowledge that can be explained and understood, and neural networks, which deals with implicit knowledge that can be acquired by learning. These systems can utilise linguistic information form of the human expert as well as measured data during modeling. A combination of these two technological innovations delivers the best results. Such applications have been developed for signal processing, automatic control, process control, data-base management etc. However, there is little discussion in the literature of more pragmatic surface roughness application of this hybrid computing system. The major objective of this work is to investigate the potential of neuro-fuzzy systems in $R_a$ prediction modeling.
3.2.1 ANFIS Architecture

ANFIS is fuzzy Sugeno models put in the framework of adaptive systems to facilitate learning and adaptation. Both ANN and fuzzy system (FS) are used in ANFIS architecture (Jang et al 2004). Such a framework makes fuzzy system more systematic and less relying on expert knowledge.

The integration of ANN and FS as shown in Figure 3.8, has demonstrated the potential to extend the technologies when applied individually. Figure 3.8(a) shows the case where one piece of equipment used the two systems for different purposes without mutual cooperation. The model Figure 3.8(b) used the ANN to optimize the parameters of the FS by minimizing the error between the output of the FS and given specification or to apply ANN learning capabilities to FS more adaptive to changing environment. Figure 3.8(c) shows a model where the output of an FS is corrected by the output of an ANN to increase the precision, the final system output.

Figure 3.8 Integration of ANN and FS
Figure 3.8(d) shows a cascade combination of an FS and ANN where the output of an FS or ANN becomes the input of another ANN or FS. Figure 3.8 (b) and (c) model refer to a combination model with an equal structure, respectively. These are described in detail in Figure 3.8(a), (b).

Figure 3.9(a) shows the total system which is described by means of FS, but the membership function (MF) of the FS is produced and adjusted by the learning of ANN. The model in Figure 3.9(b) shows the FS can be described by ANN, the max (V) and min (Λ) operators for fuzzy interface which are used to map the relationship between input and output of the ANFIS. The inference processing of the FS is responded to by the ANN.

**Figure 3.9 Combination model with an equal structure**

**Figure 3.10 ANFIS architecture**
For simplicity, it is assumed that the fuzzy inference system under consideration has two inputs $x$ and $y$ and one output. For first order Sugeno fuzzy model, a common rule set with two fuzzy ‘If then rule’ is as follows:

**Rule 1:** IF $x$ is $A_1$ and $y$ is $B_1$, THEN $f_1 = p_1x + q_1y + r_1$

**Rule 2:** IF $x$ is $A_2$ and $y$ is $B_2$, THEN $f_2 = p_2x + q_2y + r_2$

Where $p_1$, $q_1$, $r_1$, $p_2$, $q_2$, $r_2$ are the linear parameters. And $A_1$, $B_1$, $A_2$ and $B_2$ are non-linear parameters. The corresponding equivalent ANFIS architecture is shown in Figure 3.10. The entire system consists of five layers. It can be noted that the circle indicates a fixed node whereas a square indicates an adaptive node (the parameters are changed during adaptation or training).

**Layer 1: Fuzzy layer**

In this layer $x$ and $y$ are the input nodes. $A_1$, $B_1$ and $A_2$, $B_2$ are the linguistic labels in the fuzzy theory (such as low or high) for dividing the MF. The membership relationship between the output and input functions of this layer can be expressed as follows.

$$O_{1,i} = \mu_{A_i}(x) \quad i = 1,2$$

$$O_{1,i} = \mu_{B_{i-2}}(x) \quad i = 3,4$$

(3.6)

Here the membership for $A_i$ can be any appropriate parameterized MF such as generalized bell shaped function as follows. As the value of these parameters change, the bell shaped function varies accordingly, thus exhibiting various forms of MF for fuzzy set.
\[ \mu_{A_i}(x) = \frac{1}{1 + \left( \frac{x - c_i}{\sigma_i} \right)^2 \beta_i} \]  

Where \( \mu_{A_i}(x) \) and \( \mu_{B_i}(x) \) are appropriate parameterized MF, \( \{a_i, b_i, c_i\} \) are premise parameters and \( O_{1,i}, O_{1,I} \) denote the output functions.

**Layer 2: Product layer**

In this layer nodes are labeled as \( M \). Each node output represents the firing strength of a rule. In general fuzzy AND operators can be used as the node function in this layer. The output \( W_1 \) and \( W_2 \) are the weight function of the next layer. The output of this layer is the product of the all incoming signals. This is defined as follows.

\[ O_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(x) \quad i = 1,2 \]  

Where \( O_{2,i} \) denote the output of the layer 2.

**Layer 3: Normalized layer**

In this layer the nodes are labeled as \( N \). Its function is to normalize the weight function in the following process.

\[ O_{3,i} = \overline{W_i} = \frac{w_i}{w_1 + w_2} \quad i = 1,2 \]  

Where \( O_{3,i} \) denotes the output of layer 3.

**Layer 4: Defuzzification layer**

In this layer the nodes are adaptive. The defuzzy relationship between the input and output of this layer can be defined as follows.
\[ O_{4,i} = \overline{W}_i f_i = \overline{W}_i \left( p_i x + q_i y + r_i \right) \tag{3.10} \]

Where \( \{p_i, q_i, r_i\} \) are consequent parameters and \( O_{4,i} \) denotes the output of layer 4.

**Layer 5: Total output layer**

The single node in this layer is a fixed node labeled sum, which computes the overall output as the summation of all incoming signals:

\[ \text{Overall output} = O_{5,1} = \sum_i \overline{W}_i f_i = \frac{\sum_i \overline{W}_i f_i}{\sum_i W_i} \tag{3.11} \]

Where \( O_{5,1} \) denote the output of the system.

### 3.2.2 Hybrid Learning Algorithm

When the premise parameters are fixed, the overall output is a linear combination of the Consequent parameters. In symbols, the output \( f \) can be written as:

\[
\begin{align*}
    f = & \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\
    = & \overline{W} f_1 + \overline{W} f_2 \\
    = & (\overline{W}_1 x) p_i + (\overline{W}_1 y) q_i + (\overline{W}_1) r_i + (\overline{W}_2 x) p_2 + (\overline{W}_2 y) q_2 + (\overline{W}_2) r_2 \\
\end{align*}
\tag{3.12}
\]

This is linear in the consequent parameters \( \{p_i, q_i, r_i\} \). A hybrid algorithm adjusts the consequent parameters in a forward pass and the premise parameters \( \{a_i, b_i, c_i\} \) in a backward pass. In the forward pass the network inputs propagate forward until layer 4, where the consequent parameters are identified by the least-squares method. In the backward pass,
the error signals propagate backwards and the premise parameters are updated by gradient descent. The Figure 3.11 shows the membership functions initial and final training. Because of the decoupling of update rules for the premise and consequent parameters are decoupled in the hybrid learning rule, a computational speedup may be possible by using variants of the gradient method or other optimization techniques on the premise parameters.

![Figure 3.11 Membership Functions initial (left) and final (right) Learning](image)

Since ANFIS and radial basis function networks are functionally equivalent under some minor conditions, a variety of learning methods can be used for both of them.

### 3.2.3 Implementation of ANFIS

The morphology for the $R_d$ prediction by ANFIS model is depicted in the Figure 3.12.
Step 1: Training the ANFIS model

The experimental turning data sets summarized in Table 3.2 were used to train the ANFIS model.
Step 2: Defining the Input and Output parameters set and Membership function

In turning process, the gauss - linear membership function is used for distribution of the input variable. The Figure 3.13 shows the initial membership function of the input parameter ‘cutting speed’ in turning.

![Fig 3.13 Initial membership function plot for ‘cutting speed’](image)

Table 3.3 Defining Input and Output Parameters

<table>
<thead>
<tr>
<th>Process parameter</th>
<th>Input / output</th>
<th>Minimum value</th>
<th>Maximum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V$</td>
<td>Input</td>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>$F$</td>
<td>Input</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>$D$</td>
<td>Input</td>
<td>0</td>
<td>2.5</td>
</tr>
<tr>
<td>$G_a$</td>
<td>Input</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>$R_a$</td>
<td>Output</td>
<td>0</td>
<td>25</td>
</tr>
</tbody>
</table>
The ranges for the variable that represent the relevant conditions of the given process are identified. The range of the values that inputs and output may take is called universe of discourse or dimension which is shown in Table 3.3. The meaningful linguistic statements are selected for each variable and expressed by appropriate fuzzy sets.

**Table 3.4 Fuzzy Expression for the Input Parameters (Turning)**

<table>
<thead>
<tr>
<th>VL</th>
<th>Very low</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Low</td>
</tr>
<tr>
<td>M</td>
<td>Medium</td>
</tr>
<tr>
<td>H</td>
<td>High</td>
</tr>
<tr>
<td>VH</td>
<td>Very high</td>
</tr>
</tbody>
</table>

The fuzzy expression for the input parameters is given in Table 3.4.

**Step 3: ANFIS structure by Grid partition**

The grid partition method is often chosen in designing a fuzzy controller, which usually involves only several state variables as the input to the controller. This partition method needs only small number of membership functions for each input. For instance, a fuzzy model with 4 inputs and 5 membership functions on each input would generate $5^4=625$ fuzzy ‘If-Then rules’.

The typical $i^{th}$ rule of the fuzzy system will look like as follows.

\[
IF \text{ speed } (v_i) \text{ is High (H) AND feed } (f_i) \text{ is medium (M) AND depth of cut is } (d_i) \text{ very high (VH) AND gray level } (g_{\alpha_i}) \text{ is low (L)}
\]
THEN $R_a$ is $p_i v_i + q_i f_i + r_i d_i + s_i g_i + c_i$.

Where $p_i$, $q_i$, $r_i$, $s_i$ and $c_i$ are the consequent parameters.

The ANFIS structure for 4 input and 5 membership functions is shown in Figure 3.14.

![Figure 3.14 ANFIS Structure for 4 Inputs 5 Membership Function](image)

**Step 4: Training ANFIS**

During the training of ANFIS model in turning, twenty seven sets of experimental data were used to conduct 45 cycle of learning.

The ANFIS learning scenario for prediction of the $R_a$ in turning is follows.

- Number of nodes : 1297
- Number of linear parameters : 3125
- Number of nonlinear parameters : 60
Total number of parameters : 3185
Number of training data pairs : 27
Number of fuzzy rules : 625

The training and checking error performance of ANFIS based on gauss-linear membership function is shown in Figure 3.15.

![Figure 3.15 Training and checking Error of ANFIS](image)

**Figure 3.15 Training and checking Error of ANFIS**

**Step 5: Optimum parameters values**

Before training the ANFIS model, the optimal consequent parameters are found. The gradient descent method is used to adjust optimally
the premise parameters corresponding to the sets in input domain. Then, the optimal consequent membership function is obtained.

**Step 6: Input testing data**

The experimental testing data sets are same as summarized in Table 3.3 were used to test the ANFIS model.

**Step 7: ANFIS Testing**

During the testing of ANFIS model for the $R_a$ prediction in turning, eight sets of experimental data were used to check the validity of the model.

**Step 8: ANFIS Test output**

Finally, the test output results obtained with ANFIS model are compared with the experimental results.

### 3.3 PREDICTION OF $R_a$ IN CNC-TURNING USING ANN TRAINED BY DEA

An evolutionary optimization method over continuous search spaces, differential evolution, has recently been successfully applied to real world and artificial optimization problems and proposed also for neural network training. However, differential evolution has not been comprehensively studied in the context of training neural network weights, i.e., how useful is differential evolution in finding the global optimum for expense of convergence speed. In this part of the work, a new differential evolution algorithm (DEA) is applied to train feed-forward multilayer perceptron neural networks (MLPNN) for the prediction of surface roughness in turning operations. Cutting speed ($V$), feed rate ($F$), depth of cut ($D$) and average gray value ($G_a$) of the surface image of work piece, acquired by
computer vision, are taken as the input parameters and average surface roughness ($R_a$) as the output parameter. The results obtained from the DEA based ANN model are compared with the back propagation (BP) based ANN and ANFIS models. It is found that the error percentage is very close, and it is also observed that the convergence speed for the DEA based ANN is higher than the BP based ANN.

3.3.1 Differential evolution methodology

Differential evolution is a type of evolutionary algorithm developed by Rainer Storn and Kenneth Price for optimization problems over a continuous domain. The prime idea of Differential evolution is to adapt the search during the evolutionary process. During the initial stage of evolution, the perturbations are large since parent individuals are far away from each other. As the evolutionary process matures, the population converges to a small region, and the perturbations adaptively become small. Hence, the Differential evolution performs a global exploratory search during the early stages of the evolutionary process and local exploitation during the mature stage of the search.

In DE, a solution, $l$, in a generation is a multi-dimensional vector $x^{l}_{G=0} = (x_1, \ldots, x_N)^T$. A population, $P_{G=K}$ at generation $G=K$ is a vector of $M$ solutions ($M > 4$). The initial population, $P_{G=0} = \{x^{l}_{G=0} \mid l = 1, 2, \ldots, M\}$, is initialized as

$$
x^{l}_{G=0} = \text{lower}(x_i) + \text{rand}[0, 1] \times (\text{upper}(x_i) - \text{lower}(x_i))
$$

(3.13)

\[ l = 1, 2, \ldots, M \quad i = 1, 2, \ldots, N \]
Where $M$ is the population size, $N$ is the solution’s dimension and each variable $i$ in a solution vector $l$ in the initial generation $G = 0$, $x_{G=0}^i$, is initialized within its boundaries $(\text{lower}(x_i), \text{upper}(x_i))$. For each target vector $x_{i,G=K}^j, i = 1, 2, 3,..., M$, a mutant vector $V'_{i,G=K}$ is generated according to the following equation:

$$V'_{i,G=K} = x_{i,G=K}^{r_2} + F \times (x_{i,G=K}^{r_1} - x_{i,G=K}^{r_2})$$  \hspace{1cm} (3.14)

Where the random indices $r_1$, $r_2$, $r_3 \in \{1, 2... M\}$, $F$ is a scaling factor $\in [0, 1]$ which controls the amplification of the differential variation $(x_{i,G=K}^{r_1} - x_{i,G=K}^{r_2})$ and it represents the amount of perturbation added to the main parent. The values of each variable in the mutant vector are changed with some cross-over probability (CR) to

$$\forall \leq N, x_{i,G=K}^j = \begin{cases} 
  x_{i,G=K}^{r_3} + F \times (x_{i,G=K}^{r_1} - x_{i,G=K}^{r_2}) & \text{if } \text{random}[0,1] m \geq CR \land i = i_{\text{rand}} \\
  x_{i,G=K}^{j} & \text{otherwise} 
\end{cases}$$  \hspace{1cm} (3.15)

The new solution replaces the old one if it is better than it and at least one of the variables should be changed. The latter is represented in the algorithm by randomly selecting a variable, $i_{\text{rand}} \in (1, N)$. After crossover, if one or more of the variables in the new solution are outside their boundaries, the following repair rule is applied.

$$x_{i,G=K}^j = \begin{cases} 
  \frac{x_{i,G=K}^j + \text{lower}(x_i)}{2} & \text{if } x_{i,G+1}^j < \text{lower} (x_i) \\
  \text{lower} (x_i) + \frac{x_{i,G=K}^j - \text{upper}(x_i)}{2} & \text{if } x_{i,G+1}^j > \text{lower} (x_i) \\
  x_{i,G+1}^j & \text{Otherwise}
\end{cases}$$  \hspace{1cm} (3.16)
Storn and Price (2005) suggested ten different working strategies of DE and some guidelines in applying these strategies for any given problem for which DE is applied. The strategies can vary based on the vector to be perturbed, number of difference vectors considered for perturbation, and finally the type of crossover used. The following are the ten different working strategies proposed by Price and Storn:

1. DE/best/1/exp
2. DE/rand/1/exp
3. DE/rand-to-best/1/exp
4. DE/best/2/exp
5. DE/rand/2/exp
6. DE/best/1/bin
7. DE/rand/1/bin
8. DE/rand-to-best/1/bin
9. DE/best/2/bin
10. DE/rand/2/bin

The general convention for the above strategies is DE/x/y/z. DE represents differential evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover used (exp, exponential; bin, binomial). However, strategy-7(DE/rand/1/bin) is the most successful and most widely used strategy. In this work, strategy-7 is attempted. The detailed algorithm (Pseudo code) is available in the literature (Nikunj Chauhan et al 2009, Basturk and Gunay 2009, Dos Santos Coelho et al 2010).
3.3.2 Neural network training using differential evolution algorithm

Differential evolution algorithm is a heuristic method for optimizing nonlinear and non-differentiable continuous space functions. Hence it can be applied to global searches within the weight space of a typical neural network. In this work, a most popular feed forward MLPNN is used. Training an MLPNN to recognize the objectors is typically realized by adopting an error correction strategy that adjusts the network weights through minimization of learning error:

\[ E = E (Y_0, Y) \]  \hspace{1cm} (3.17)

Where, \( Y \) is the real output vector of a MLPNN, \( Y_0 \) is the target output vector and \( Y \) is a function of synaptic weights \( w \) and input values \( X \). In the MLPNN, the input vector \( x \) and the target output vector \( Y_0 \) are known and the synaptic weights in \( W \) are adapted to obtain appropriate functional mappings from the input \( x \) to the output \( Y_0 \). Normally, the adaptation can be carried out by minimizing the network error function \( E \), i.e. network training procedure. The optimization goal is to minimize the objective function \( E \) by optimizing the values of the network weights:

\[ W = (W_1, W_2... W_D) \] \hspace{1cm} (3.18)

Differential Evolution maintains a population \( M \) of constant size and the real value vector \( l_i^G \), where \( i (i=1,2,...,M) \) is the index to the population and \( G(G=1,2,...,G_{\text{max}}) \) is the generation to which the population belongs:

\[ P^G = \{ l_1^G, l_2^G, ..., l_M^G \} \]  \hspace{1cm} (3.19)

Based on the differential evolution methodology discussed in section 3.3.1, each individual of the population is compared with its counterpart in the current population and the vector with the lower objective function value wins a place in the next generation’s population. As a result,
all the individuals of the next generation are as good as or better than their counterparts in the current generation.

### 3.3.3 Implementation of neural network trained differential evolution

To implement the proposed methodology in DEA, initially, a population of size $M$ is randomly generated for neural network weightages. The error between the target output vector and the real output vector of a MLPNN is evaluated by using the experimental data sets. Then the differential evolution method is applied to train the neural network. From the initial population, a target vector and base vector ($r_3$) are chosen. Thereafter, two vectors are selected randomly ($r_1, r_2$) from the same population and the weighted difference vector ($r_1 \sim r_2$) is computed. Then, the weighted difference is added to the base vector. In order to increase the diversity of the perturbed vectors, crossover is performed. After the cross-over, a trial vector as shown in equation (3.15) is formed. This trial vector is then compared with the target vector using greedy criterion to decide the better one. In this manner, a new population is generated and evaluated till the solution converges.

### 3.3.4 Training ANN-DEA

The experimental turning data sets summarized in Table 3.2 are used to train the neural network with DEA and then the response values are predicted. The crossover constant (CR) and scaling factor (F) are taken (optimal values obtained after different trials) as 0.8 and 0.5, respectively. Schematic Flow chart for the neural network training using differential evolution algorithm is shown in Figure 3.16.
Figure 3.16 Schematic flow chart for ANN training using DEA
Neural network with different topologies has been trained, and the optimal structure (4-6-6-1) is found out by trial-and-error approach for the error convergence 0.0001 as shown in Figure 3.17 and 3.18. The results obtained from the DEA-based ANN model are compared with the BP-based ANN and ANFIS models. The explanation of results and discussion are given in the separate chapter 6.