Pakistan Journal of Applied Sciences 2(3): 281-287, 2002

Object Recognition Using ANN with Backpropogation Algorithm

I. M. Qureshi and A. Jalil
Department of Electronics, Quaid-e-Azam University, Islamabad, Pakistan

Abstract: An object recognition procedure using artificial neural network is proposed in this paper. This algorithm will recognize objects in the images which are invariant to rotation, translation and scaling of objects. The Sobel operators are proposed to detect the boundary of the object. A Fourier descriptor pattern classifier is able to classify object without regard to rotation, translation and scale variation. Fourier descriptors of boundary image generate Feature vectors by truncation the high frequency components. A back-propagation neural networks is proposed to recognize the object based on these Feature vectors.

Keywords: Object Recognition, Edge Detection, Feature Vector, Back-Propagation, Algorithm, Neural Networks

Introduction

When talking about object recognition, it means information about the object to be recognized is usually known in advance. Energy from a scene is converted into a 2-D entity called image. How these images are processed to form conclusion is the basis of object recognition. The problem of object recognition (or pattern recognition) turns out to be that of object classification or pattern matching.

The main focus is on the feature extraction and classification stage. Two major steps are involved in object recognition:

- Low level vision processing, in which edges or boundary of objects are detected and a feature vectors for each set of visual image data is made. This process is often called feature extraction.
- Classification of feature vectors extracted from the low level step.

The following procedure is adopted for object recognition:

First the bitmap file of the image is taken and its edges are extracted by utilizing the Sobel operator (Gonzalez and Wintz, 1987). By this operation boundary position data is obtained. Taking the discrete Fourier transform of the data the Fourier coefficients are computed. Finally the Fourier descriptors are acquired that have the invariant property with respect to rotation, translation and size (Hardie and Boncelet, 1995).

Secondly, back-propagation neural net is trained using the feature vectors of some sample objects. The feature vectors of a test object are given to a trained neural net as an input, to recognize that object. Due to the robustness of neural networks, this algorithm is very robust with respect to noise.

When an image is acquired it may contain undesired data such as noise, may be blurred due to focusing errors and any other errors, which can distort the image. All such type of errors introduced into the image during acquisition needs to be removed, before any meaningful processing can be done. This involves removing noise, enhancing the picture, and, if necessary, segmenting the image into meaningful regions to be analyzed separately.

Feature Extraction: There are two phases for feature extraction, first edge detection and then formulation of feature vector using Discrete Fourier Transform (DFT).

Edge Detection using Sobel operator: Edge detection is most common approach for detecting meaningful discontinuities in gray level. Edges are places in the image with strong intensity contrast. Since edges often occur at image locations representing object boundaries. Edge detection is extensively used in image segmentation, when image is to be divided into areas corresponding to different objects. Representing an image by its edges has the further advantage that the amount of data is reduced significantly while most of the image information is retained.

Since edges consist mainly of high frequencies, these can be detected by applying high pass filter in the frequency domain or by convoluting the image with an appropriate kernels (mask), in the spatial domain. These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation called G_x and G_y. These can then be combined together to find the absolute magnitude of the gradient G at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{G_x} + \sqrt{G_y} \quad (1)$$

An approximate magnitude is computed using:

$$|G| = |G_x| + |G_y| \quad (2)$$

Which is much faster to compute.

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \quad (3)$$

In this case, orientation 0 is taken to mean that the direction of maximum contrast from black to white runs from left to right on the image, and other angles are measured anti-clockwise from this. We use a Sobel operators as given in Fig 2. These are slightly more computationally heavy but give more accurate results. They involve pixels in 3×3 neighborhood about point (x,y). This approach can be formulated by considering

sub-image area shown in Fig. 1, where x_5 represents the gray level at location left (x,y) and other x_i represent the gray levels of the 8-neighbours of (x,y). We define the component of the gradient vector in the x-direction as

$$G_x=(x_7+2x_8+x_9)-(x_1+2x_2+x_3) \quad (4)$$

and in y-direction as

$$G_x=(x_3+2x_6+x_9)-(x_1+2x_4+x_7) \quad (5)$$

Using a 3×3 area in the computation of the gradient has the advantage of increased smoothing over 2×2 operators, tending to make the derivative operations less sensitive to noise. Weighting the pixels closest to the center by 2 also produces additional smoothing.

X_1	X_2	X_3
X_4	X_5	X_6
X_7	X_8	X_9

Fig. 1: 3×3 Image Region

-1	0	+1
-2	0	+2
-1	0	+1

G_x

+1	+2	+1
0	0	0
-1	-2	-1

G_y

Fig. 2: Sobel Operators

Feature Vectors Using DFT: By applying Sobel operator edges of the image were obtained from which boundary position data $(x[m],y[m])$, for $1\le m\le L$ was extracted. Where L is total data points.

By taking the discrete Fourier transform of the data $(x[m],y[m])$,we obtained the Fourier coefficients as

$$a[k]=\frac{1}{L}\sum_{m=1}^{L}x[m]e^{-jk(\frac{2\pi}{L})m} \quad (6)$$

under

$$b[k]=\frac{1}{L}\sum_{m=1}^{L}y[m]e^{-jk(\frac{2\pi}{L})m} \quad (7)$$

where

$1\le k\le L-1.$

We discard the dc components, $a[0]$and $b[0]$ since they carry information only about the position of the image center. We let

$$r[k]=\sqrt{|a[k]|^2+|b[k]|^2} \quad (8)$$

and

$$s[k]=\frac{r[k]}{r[1]} \quad (9)$$

where

$1\le k\le L-1$ and $|a[k]|,|b[k]|$ denote the absolute value of the complex numbers $a[k],b[k]$.Then it is easy to say that $r[k]$ are invariant to rotation or translation, and further that $s[k]$ are invariant under scaling. The invariant descriptors $s[k]$are called the Fourier descriptors. The low frequency part of $s[k]$ is determined by the overall shape of an object, while the high frequency part gives the detailed shape. The high frequency part is easily degraded by noise and does not contribute much to object recognition compared with low frequency part, therefore the high frequency part is discard. This compresses the data significantly, which is one of the reasons for using Fourier descriptors. We used only the first 16 components $s[1],s[16]$as feature vector. First16 components constitute 99.9% of the total length (Kim and Nam, 1995)

$$total\ length\ =\sum_{i=1}^{L/2}s[i]$$

Artificial Neural Networks

Back-Propagation Algorithm for Multilayer Perceptron: The approach of the adaptive learning (updating) of synaptic weights can be extended to multilayer perceptron. For simplicity, we consider a multilayer perceptron of three layers: first hidden layer consists of n_0 inputs and n_1 units, the second layer with n_2 units and the output layer with n_3 units.

The behavior of the network is determined on the basis of inputs and outputs pairs. The input output pairs are expressed as stable states of neurons, which are usually represented, by +1 (ON) and -1(OFF). The learning of multilayer perceptron for a specific task is equivalent to finding the values of all synaptic weights such that the desired output is generated for a given input. Therefore learning of multilayer perceptron consists in adjusting all weights such that error measure between the desired output signal d_{jp} and the actual output signals y_{jp}averaged over all learning examples $p$will be minimum (possibly zero). The standard back-propagation

algorithm uses the steepest descent gradient approach to minimize the mean square error function. Such a local error function for any p^{th} learning example can be written as

$$E_p = \frac{1}{2}\sum_{j=1}^{n_3}(d_{jp} - y_{jp})^2 = \frac{1}{2}\sum_{j=1}^{n_3} e_{jp}^2 \quad (10)$$

Now the global error function may be written as

$$E = \sum_p E_p = \frac{1}{2}\sum_p\sum_j (d_{jp} - y_{jp})^2 \quad (11)$$

Where

d_{jp} and y_{jp} are the desired and actual output signals of the j^{th} output neuron for the p^{th} pattern respectively.

Now here we will discuss the online learning approach in which the gradient search in the synaptic weight space is carried out on the basis of a local error function E_p. First determine and updating formula for the synaptic weights $w_{ji}^s(s=3)$ of the output layer. Using the chain rule for Eq. 10 we can write

$$\Delta w_{ji}^{[3]} = -\eta\frac{\partial E_p}{\partial w_{ji}^{[3]}} = -\eta\frac{\partial E_p}{\partial u_j^{[3]}}\frac{\partial u_j^{[3]}}{\partial w_{ji}^{[3]}} \quad (12)$$

Here

$$u_j^{[3]} = \sum_{i=1}^{n_3} w_{ji}^{[3]} x_i^{[3]} = \sum_{i=1}^{n_3} w_{ji}^{[3]} o_i^{[2]} \quad (13)$$

and local error, called delta is define as

$$\delta_j^{[3]} = -\frac{\partial E_p}{\partial u_j^{[3]}} = \frac{\partial E_p}{\partial e_{jp}^{[3]}}\frac{\partial e_{jp}^{[3]}}{\partial u_j^{[3]}} = e_{jp}\frac{\partial \Psi_j^{[3]}}{\partial u_j^{[3]}} \quad (14)$$

we obtained general formula for updating the weights in the output layer as

$$\Delta w_{ji}^{[3]} = \eta\delta_j^{[3]} x_i^{[3]} = \eta\delta_j^{[3]} o_i^{[2]} \quad (15)$$

where

$$\delta_j^{[3]} = e_{jp}(\Psi_j^{[3]})' = (d_{jp} - y_{jp})\frac{\partial \Psi_j^{[3]}}{\partial u_j^{[3]}} \quad (16)$$

Updating the synaptic weights in the hidden layers is a little more complicated. For the second hidden layer we can still write

$$\Delta w_{ji}^{[2]} = -\eta\frac{\partial E_p}{\partial w_{ji}^{[2]}} = -\eta\frac{\partial E_p}{\partial u_j^{[2]}}\frac{\partial u_j^{[2]}}{\partial w_{ji}^{[2]}} = \eta\delta_j^{[2]} x_i^{[2]} = \eta\delta_j^{[2]} o_i^{[1]} \quad (17)$$

where the local error for the second hidden layer is defined as

$$\delta_j^{[2]} = -\frac{\partial E_p}{\partial u_j^{[2]}} \qquad (j = 1,2,3.....n_2) \quad (18)$$

However this local error cannot be directly evaluated as is done for the local errors in the output layer. Using the chain rule we can write

$$\Delta w_{ji}^{[2]} = -\frac{\partial E_p}{\partial u_j^{[2]}} = -\frac{\partial E_p}{\partial o_j^{[2]}}\frac{\partial o_j^{[2]}}{\partial u_j^{[2]}} \quad (19)$$

As we know that

$$\delta_j^{[2]} = \Psi_j^{[2]}(u_j^{[2]}) \quad (20)$$

We have

$$\delta_j^{[2]} = -\frac{\partial E_p}{\partial o_j^{[2]}}\frac{\partial \Psi^{[2]}}{\partial u_j^{[2]}} \quad (21)$$

The factor

$-\frac{\partial E_p}{\partial o_j^{[2]}}$ can be evaluated as

$$-\frac{\partial E_p}{\partial o_j^{[2]}} = -\sum_{i=1}^{n_3}\frac{\partial E_p}{\partial u_i^{[3]}}\frac{\partial u_i^{[3]}}{\partial o_j^{[2]}} = \sum_{i=1}^{n_3}\left[-\frac{\partial E_p}{\partial u_i^{[3]}}\right]\frac{\partial}{\partial o_j^{[2]}}\left[\sum_{i=1}^{n_3} w_{ik}^{[3]} x_k^{[3]}\right] = \sum \delta_j^{[3]}\frac{\partial}{\partial o_j^{[2]}}\left[\sum_{i=1}^{n_3} w_{ik}^{[3]} o_k^{[3]}\right] = \sum \delta_j^{[3]} w_{ij}^{[3]} \quad (22)$$

The local error in the second layer can be evaluated as

$$\delta_j^{[2]} = \frac{\partial \Psi^{[2]}}{\partial u_j^{[2]}}\sum \delta_j^{[3]} w_{ij}^{[3]} \quad (23)$$

Analogously,

$$\Delta w_{ji}^{[1]} = \eta\delta_j^{[1]} x_i^{[1]} = \eta\delta_j^{[1]} o_i^{[0]} = \eta\delta_j^{[1]} x_i \quad (24)$$

Generally, the local error of the hidden layers is determined on the basis of the local errors at an upper layer. Starting with the highest output layer we compute $\delta_j^{[3]}$, we can propagate the errors $\delta_j^{[3]}$ backward to the lower layers. Fig. 3 show the functional scheme of the back-propagation algorithm. The major difference of the

learning rule for the output layer and the hidden layers

is the evaluation of the local error $\delta_j^{[s]}$ $(s=1,2,3)$.In the output layer the error is the function of the desired and the actual output and the derivative of the sigmoid activation function. For the hidden layers the local errors are evaluated the basis of the local errors in the upper layer.

Realizing the following steps can perform the basic back-propagation algorithm:

Step 1: Initialize all synaptic weights $w_{ij}^{[s]}$ to small random values

Step 2: Present input for the class of learning examples and calculate the actual outputs of all neurons using the present values of

$w_{ij}^{[s]}$ and the pattern.

Step 3: Specify the desired output and evaluate the local errors

$\delta_j^{[s]}$ for all layers.

Step 4: Adjust the synaptic weights according to the iterative formula

$$\Delta w_{ji}^{[s]} = \eta \delta_j^{[s]} x_i^{[s]} \qquad (25)$$

Step 5: Present another input pattern corresponding to the next learning example and go back to step 2.

All the training examples are presented cyclically until the synaptic weights are stabilized, i.e. until the error of the entire set is acceptably low and the network converges. After training a multilayer perceptron usually has the feature of generalization, i.e. it has the ability for proper response to input patterns not presented during the learning process. Such a generalization is an important feature of multilayer perceptrons.

Estimation of the Classification Error: An important parameter in any pattern classification problem is the estimation of the classification error. To compute it the available samples must be divided into two sets: one for training and one for testing. There were twelve training images and twelve testing images per character. The estimation of the error rate is done by finding the ratio of the misclassified test samples to the total number of tested samples. So

$$Estimate\ of\ Error = \frac{number\ of\ misclassified\ test\ samples}{total\ number\ of\ tested\ samples}$$

Results and Discussion

Major task in the recognition system is to recognize the first three capital English characters. In our research, a feature vector for each character was obtained as described in Feature Extraction portion and we took first 16 Fourier descriptors as they contain the maximum useful information. This feature vector for each character was taken as the input for the Neural Networks to train them and also for the test purpose. The detail results of these processes are as given below:

An object image was captured in 100×100 pixels with 256 gray level resolutions in a bmp file. The different images taken for the character ``X'', ``Y'' and ``Z'' are shown in the Fig. 4a, 5a and 6a.

The pixel data was obtained from the bmp file and stored in 100×100 matrix. For edge detection, we took two Sobel operators i.e., one for horizontal edge detection and other for vertical edge detection. These operators are shown in Fig. 2.

These two operators are convolved with the pixel data to get two 100×100 matrix. One containing horizontal edges and other vertical edges. These two data matrices are then added to get the edges of the image. The second order derivative operator was used on the original image on the specified detected edge image points. It will detect weather a given edge detected pixel belonged to the dark or light side of an edge. This reduced the position data and it also refined the boundary edge of the object i.e., we obtained a single pixel boundary of the given object. The results of the edge detection for characters ``X'', ``Y'' and ``Z'' are shown in the Fig. 4b, 5b and 6b.

The first 16 Fourier descriptors for different training images are shown in the table 1, 2 and 3 for the characters ``X'', ``Y'' and ``Z'', respectively. So we selected the first half values. In our work, only the first 16 components were used. For each character twelve test images were prepared.

Now the algorithm tested 12 test images for each character. Then I calculated mismatched results and divided it by the total number of input test images. It was the estimate of classification error, which is

$$Estimate\ of\ Error = \frac{number\ of\ misclassified\ test\ samples}{total\ number\ of\ tested\ samples}$$

We have found that if the test file contains Fourier descriptors of the training images then we found that the algorithm accurately recognized all the images. This showed 0% estimation error. It also represented the neural network was trained well.

When the Fourier descriptors of the test images were applied to the trained neural networks, the results showed that four characters were not accurately recognized out of thirty-six images. Therefore

$$Estimate of\ Error = \frac{4}{36} = 0.11$$

which means the accuracy of the results is about 90%.

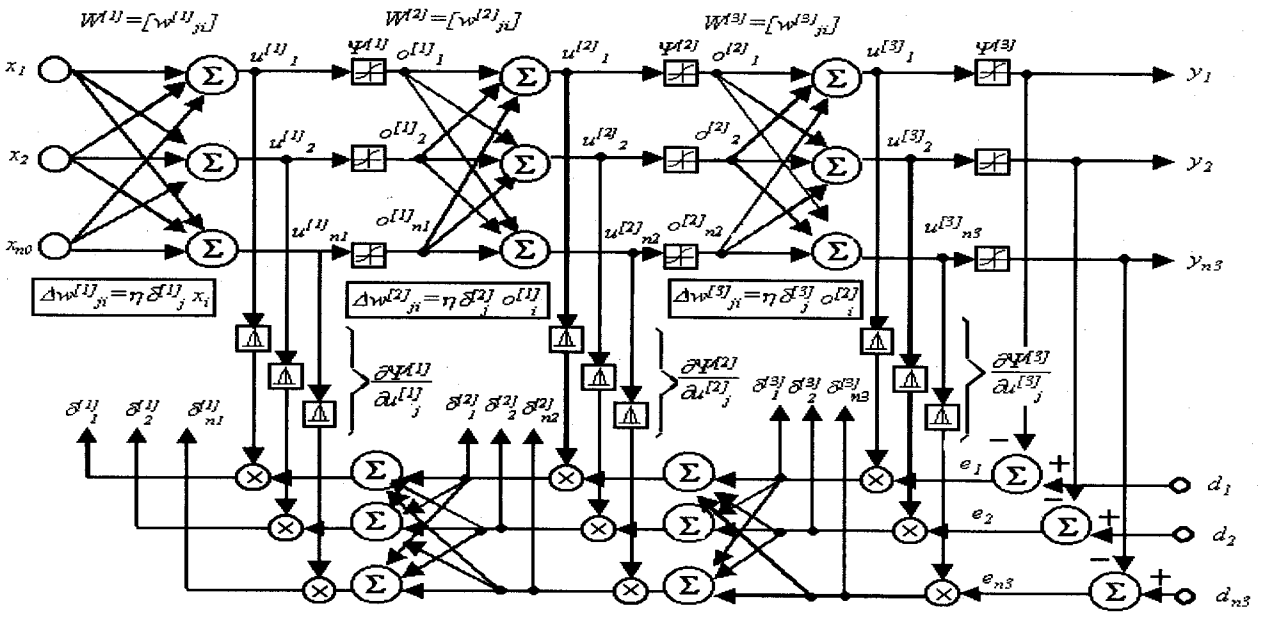

Fig. 3: Network Architecture for Tree-layered NN with Back-propagation Algorithm

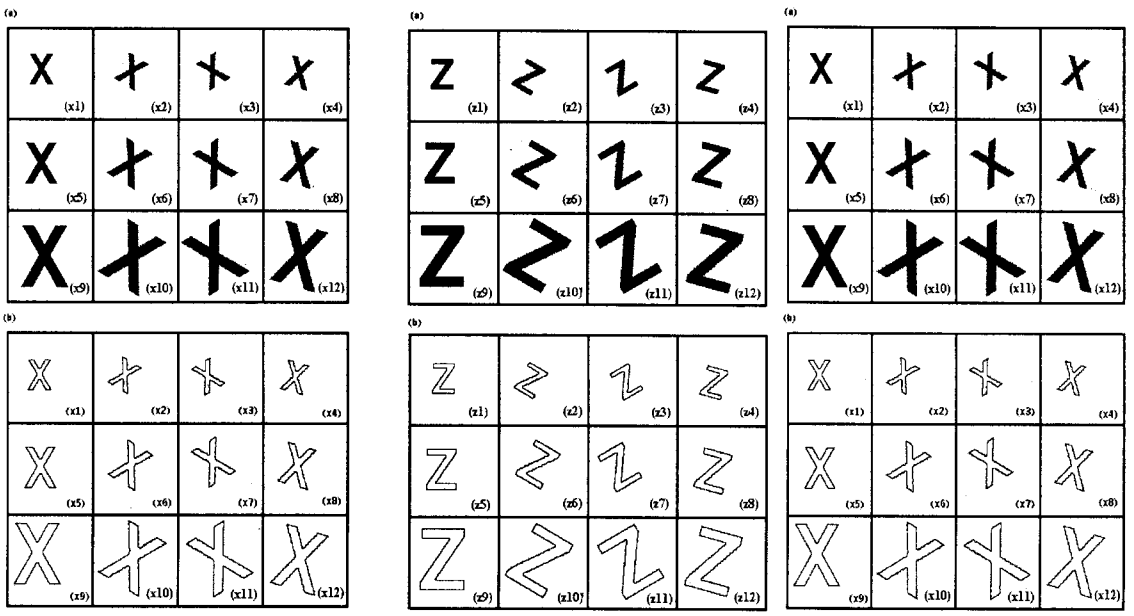

Fig. 4: The Different Positions of 'X', 'Y', 'z'

Fourier Descriptors of Different Training Images

TABLE. 5.1 The Variations of the Fourier Descriptors of 12 Figures of character "X"

n	(x1)	(x2)	(x3)	(x4)	(x5)	(x6)	(x7)	(x8)	(x9)	(x10)	(x11)	(x12)
	1	1	1	1	1	1	1	1	1	1	1	1
1	0.372759	0.440404	0.413457	0.37028	0.378639	0.414195	0.4286	0.359078	0.381953	0.405054	0.437759	0.366892
2	0.317258	0.347826	0.342756	0.323208	0.316065	0.346057	0.359223	0.311867	0.316607	0.345127	0.354191	0.309376
3	0.19925	0.29962	0.292726	0.305871	0.20234	0.300273	0.305903	0.305619	0.202181	0.294012	0.309594	0.298556
4	0.177085	0.220138	0.219251	0.24947	0.176858	0.211241	0.212613	0.246764	0.177411	0.207002	0.212341	0.247852
5	0.136074	0.183681	0.177153	0.20374	0.136731	0.177199	0.171387	0.202231	0.137635	0.175023	0.17341	0.209589
6	0.114988	0.145651	0.147324	0.145922	0.113528	0.151556	0.144678	0.145067	0.114209	0.135608	0.145893	0.144378
7	0.104199	0.120018	0.11638	0.120019	0.104112	0.122648	0.117725	0.124944	0.101887	0.125807	0.117245	0.12197
8	0.11613	0.107204	0.118766	0.107291	0.100512	0.109646	0.106423	0.102974	0.101045	0.106827	0.117285	0.103353
9	0.107785	0.109966	0.122369	0.096045	0.106134	0.100629	0.107591	0.091189	0.113332	0.098072	0.103728	0.091199
10	0.124497	0.101906	0.114057	0.08579	0.11869	0.101662	0.123057	0.087055	0.132413	0.101375	0.112082	0.080312
11	0.209437	0.079013	0.088832	0.088455	0.136281	0.105123	0.082206	0.089438	0.144192	0.079168	0.081868	0.075735
12	0.200496	0.07982	0.078922	0.112298	0.167092	0.0762	0.074644	0.077751	0.131965	0.073749	0.074489	0.072538
13	0.142725	0.076018	0.071416	0.1015	0.161714	0.079477	0.071484	0.076909	0.128569	0.0685	0.074288	0.076422
14	0.125932	0.065744	0.068021	0.079053	0.158499	0.062901	0.062373	0.072042	0.140608	0.063479	0.067761	0.071758
15	0.094148	0.078691	0.062103	0.073431	0.130241	0.058008	0.061168	0.093595	0.131189	0.072602	0.078047	0.079114

TABLE. 5.2 The Variations of the Fourier Descriptors of 12 Figures of character "Y"

n	(y1)	(y2)	(y3)	(y4)	(y5)	(y6)	(y7)	(y8)	(y9)	(y10)	(y11)	(y12)
1	1	1	1	1	1	1	1	1	1	1	1	1
2	0.493153	0.583042	0.706504	0.659251	0.832664	0.751357	0.911324	0.815117	0.314897	0.342622	0.48147	0.305287
3	0.5859	0.550143	0.703392	0.613022	0.400136	0.375591	0.372884	0.238326	0.417689	0.304588	0.425775	0.2748
4	0.302604	0.365261	0.282597	0.309145	0.420088	0.461716	0.518733	0.471492	0.196837	0.331674	0.321501	0.322961
5	0.229451	0.332505	0.383497	0.367005	0.377487	0.325308	0.21777	0.273013	0.184469	0.241947	0.25123	0.310871
6	0.316917	0.32739	0.309865	0.349151	0.213759	0.328163	0.334226	0.348399	0.219148	0.156661	0.145285	0.193802
7	0.187361	0.152702	0.118741	0.170646	0.295968	0.180832	0.17708	0.202365	0.140047	0.140888	0.147405	0.122695
8	0.104626	0.212043	0.259031	0.239029	0.111818	0.226893	0.237027	0.22275	0.126921	0.148931	0.163039	0.15318
9	0.185473	0.1666	0.155729	0.148653	0.20527	0.121611	0.151671	0.116449	0.130623	0.106575	0.116521	0.115927
10	0.113664	0.106335	0.10997	0.121586	0.110441	0.173188	0.173686	0.189984	0.079477	0.080175	0.101505	0.081076
11	0.05621	0.156563	0.178411	0.174057	0.122171	0.100222	0.098299	0.09847	0.092377	0.093754	0.102632	0.081004
12	0.131578	0.11018	0.091559	0.101824	0.129856	0.140471	0.145701	0.158096	0.107698	0.091736	0.091482	0.117019
13	0.104374	0.097188	0.118693	0.15251	0.101188	0.1057	0.103603	0.09994	0.093962	0.078774	0.07987	0.105198
14	0.067777	0.121957	0.145705	0.1357	0.145308	0.111423	0.122614	0.121023	0.098979	0.055918	0.053476	0.046758
15	0.10215	0.077088	0.06087	0.061088	0.07501	0.093476	0.085493	0.089597	0.098647	0.055044	0.069891	0.089528
16	0.0914	0.075725	0.121562	0.124743	0.129122	0.111161	0.118027	0.1016	0.089653	0.086636	0.08365	0.102849

TABLE. 5.3 The Variations of the Fourier Descriptors of 12 Figures of character "Z"

n	(z1)	(z2)	(z3)	(z4)	(z5)	(z6)	(z7)	(z8)	(z9)	(z10)	(z11)	(z12)
1	1	1	1	1	1	1	1	1	1	1	1	1
2	0.5814	0.469309	0.803576	0.551818	0.505922	0.369918	0.60321	0.401587	0.378372	0.459215	0.4137	0.455422
3	0.466287	0.241258	0.392876	0.279186	0.257285	0.359299	0.300116	0.364235	0.283055	0.288129	0.325237	0.261771
4	0.336686	0.502446	0.550091	0.519967	0.419325	0.303668	0.322668	0.305916	0.212353	0.202752	0.222671	0.207875
5	0.407104	0.239168	0.283494	0.251173	0.299862	0.272189	0.289428	0.252672	0.245194	0.176935	0.183145	0.204325
6	0.152117	0.264333	0.317001	0.27037	0.218184	0.182665	0.200461	0.194369	0.232163	0.141283	0.131132	0.14465
7	0.204219	0.117131	0.135938	0.11161	0.182145	0.142163	0.144124	0.173444	0.290192	0.122422	0.116524	0.112069
8	0.145375	0.208046	0.262741	0.223284	0.213063	0.145541	0.170525	0.154881	0.124903	0.109007	0.112238	0.095834
9	0.182858	0.109056	0.1238	0.121498	0.163611	0.141747	0.157303	0.169352	0.091781	0.096855	0.103627	0.086145
10	0.171446	0.119476	0.160266	0.121381	0.158984	0.096519	0.094364	0.09093	0.079426	0.091427	0.084727	0.085896
11	0.049168	0.094931	0.122024	0.105969	0.194585	0.128775	0.132342	0.120746	0.089638	0.077736	0.088917	0.078297
12	0.227253	0.186766	0.223036	0.193835	0.174299	0.140092	0.183007	0.167759	0.110257	0.072478	0.075952	0.06654
13	0.12401	0.090361	0.112097	0.098111	0.074993	0.114216	0.107974	0.11243	0.079419	0.072279	0.067186	0.063479
14	0.087235	0.071497	0.106185	0.079195	0.065976	0.083839	0.105212	0.051991	0.069062	0.060934	0.060402	0.064247
15	0.159584	0.089428	0.090731	0.087183	0.060918	0.113933	0.108944	0.110428	0.083509	0.05717	0.059396	0.069885
16	0.14018	0.149079	0.190095	0.165716	0.151064	0.119764	0.121801	0.11788	0.086343	0.056046	0.057601	0.067869

Conclusion

A neural network was proposed for classification and recognition of an object. Classifier was so robust enough to accommodate the variations in the feature representation. For feature we have utilized Fourier descriptors that are invariant under the image translation, rotation and scaling. Three characters X, Y and Z were used for experiment. Feedforward neural networks using back-propagation algorithm was trained for these objects. After sufficient training it was able to distinguish the test objects. When we used any object as test object having one of the alignment among the alignments used for the training, the result was 100%. In case we gave any other alignment the result was roughly 90% accurate. This accuracy can be increased if we increased the size of the training samples of each object. Obviously, this demands more number of computations and more time for training. One of the way is to tryout another transform instead of Discrete Fourier Transform and that is Discrete Wavelet Transform. It is expected to get feature vectors of objects, which are more distinguished from each other.

References

A.V. Oppenheim and R.W. Schafer, 1989. Discrete Time Signal Processing, Englewood Cliffs, N.J: Prentice Hall.

A. Khotanzad and J. Lu, 1990." Classification of invariant image representation using a neural network," IEEE Trans. Acoust., Speech, Signal Process., vol. 38.

H. Kim and K. Nam, 1995. "Object Recoginition of One DOF Tool by Back-propagation Neural Nets," IEEE Trans. Neural Network, vol. 6.

J. A. Anderson and E.Rosenfeld (Eds), 1988. Neurocomputing Foundation of Research, Cambridge: The M.I.T. Press,

M.Shridhar and A. Badreldin, 1984." High accuracy character recognition algorithm using Fourier and topological descriptors," Pattern Recognition, vol. 17, pp.515-524.

R.C. Gonzalez and P. Wintz, 1987. Digital Image Processing, Addison-Wesley.

R.C Hardie and C. G Boncelet, 1995. "Gradient-based edge detection using non-linear edge enhancing prefilters," IEEE Trans. Image Processing, vol. 4.

R. P. Lippmann, 1987. "An introduction to computing with neural nets," IEEE ASSP Magazine.