Character Recognition using Artificial Neural Networks

Chetan S, Department of Computer Science & Engineering, NIT Calicut

***Abstract*—One of the main challenges faced by current technology is the accurate identification of objects. It is very useful to have a machine perform pattern recognition. In particular, machines that can read symbols are very cost effective. A machine that reads banking checks can process many more checks than a human being in the same time. This kind of application saves time and money, and eliminates the requirement that a human perform such a repetitive task. In this paper, we introduce Character Recognition using the techniques of Artificial Neural Networks (ANN) such as a backpropagation network.**

***Index Terms*—Character recognition, Artificial Neural Network**

I. Introduction

Machinery was invented to aid man in automating tasks and aiding production. While the growth of machines have been exponential, certain shortcomings have crept up from time to time. One of the key challenges faced by modern technology fields like Computer Vision (CV) or Pattern Recognition (PR) is to recognize a generic object with the best possible accuracy. Although there have been several techniques to recognize objects ranging from the field of Signal Processing (SP) to Artificial Neural Networks (ANN), accuracy has always been a concern.

Many applications of Character Recognition (CR) such as the Automatic Guided Vehicles (AGVs), Robotic Vision among others, require dynamic processing of images or characters to perform future actions. Incorrectly recognizing an object or failure to recognize an object will prove to be too costly in such systems. Hence, accuracy and processing time are quintessential in object character recognition (OCR).

In this paper, we provide a method for character recognition using backpropagation network as given in [1]. We also perform comparitive studies based on the number of neurons in the network.

The rest of the paper is organized as follows. Section II defines the problem, Section III builds up on the backpropagation network using ANN and Section III-A gives details about the experimental setup. Section IV provides the simulation results and comparitive studies and Section V concludes the paper.

II. Problem Statement

A network is to be designed and trained to recognize the 26 letters of the English alphabet. An imaging system that digitizes each letter centered in the system's field of vision is available. The result is that each letter is represented as a 5-by-7 grid of Boolean values.

For example, the letter 'A' is represented in Figure 1

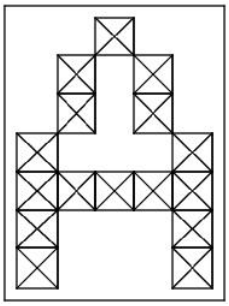

Fig. 1. Letter 'A'

Ideally, all the characters are without fault. But in practical situations, the imaging system is not perfect and the letters may suffer from noise. Figure 2 shows the letter 'A' with noise.

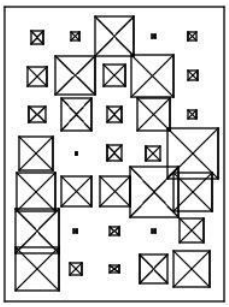

Fig. 2. Letter 'A' suffering from noise

Perfect classification of ideal input vectors is required, and reasonably accurate classification of noisy vectors.

A. Features in `MATLAB`

The twenty-six 35-element input vectors are defined in the function `prprob` as a matrix of input vectors called `alphabet`. The target vectors are also defined in this file with a variable called `targets`. Each target vector is a 26-element vector with a '1' in the position of the letter it represents, and 0's everywhere else. For example, the letter A is to be represented by a 1 in the first element (as A is the first letter of the alphabet), and 0's in elements two through twenty-six.

III. Neural Network

The network receives the 35 Boolean values as a 35-element input vector. It is then required to identify the letter by responding with a 26-element output vector. The 26 elements of

the output vector each represent a letter. To operate correctly, the network should respond with a 1 in the position of the letter being presented to the network. All other values in the output vector should be 0.

In addition, the network should be able to handle noise. In practice, the network does not receive a perfect Boolean vector as input. Specifically, the network should make as few mistakes as possible when classifying vectors with noise of mean 0 and standard deviation of 0.2 or less.

A. Architecture

The neural network needs 35 inputs and 26 neurons in its output layer to identify the letters. The network is a two-layer `log-sigmoid/log-sigmoid` network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output boolean values.

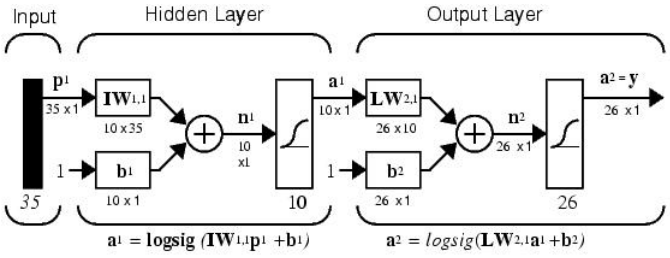

Fig. 3. Neural network architecture

The hidden (first) layer has 10 neurons. This number was picked by guesswork and experience. If the network has trouble learning, then neurons can be added to this layer.

The network is trained to output a 1 in the correct position of the output vector and to fill the rest of the output vector with 0's. However, noisy input vectors may result in the network not creating perfect 1's and 0's. After the network is trained the output is passed through the competitive transfer function `compet`. This makes sure that the output corresponding to the letter most like the noisy input vector takes on a value of 1, and all others have a value of 0. The result of this post-processing is the output that is actually used.

B. Initialization of network

The two-layer network is created with `newff`.

```
S1 = 10;
[R,Q] = size(alphabet);
[S2,Q] = size(targets);
P = alphabet;
net = newff(minmax(P),[S1 S2],
      {'logsig' 'logsig'},'traingdx');
```

C. Training of network

To create a network that can handle noisy input vectors it is best to train the network on both ideal and noisy vectors. To do this, the network is first trained on ideal vectors until it has a low sum-squared error.

Then, the network is trained on 10 sets of ideal and noisy vectors. The network is trained on two copies of the noise-free alphabet at the same time as it is trained on noisy vectors. The two copies of the noise-free alphabet are used to maintain the network's ability to classify ideal input vectors.

Unfortunately, after the training described above the network may have learned to classify some difficult noisy vectors at the expense of properly classifying a noise-free vector. Therefore, the network is again trained on just ideal vectors. This ensures that the network responds perfectly when presented with an ideal letter.

All training is done using backpropagation with both adaptive learning rate and momentum with the function `trainbpx`.

1) Training without Noise: The network is initially trained without noise for a maximum of 5000 epochs or until the network sum-squared error falls beneath 0.1.

```
P = alphabet;
T = targets;
net.performFcn = 'sse';
net.trainParam.goal = 0.1;
net.trainParam.show = 20;
net.trainParam.epochs = 5000;
net.trainParam.mc = 0.95;
[net,tr] = train(net,P,T);
```

2) Training with Noise: To obtain a network not sensitive to noise, we trained with two ideal copies and two noisy copies of the vectors in `alphabet`. The target vectors consist of four copies of the vectors in `target`. The noisy vectors have noise of mean 0.1 and 0.2 added to them. This forces the neuron to learn how to properly identify noisy letters, while requiring that it can still respond well to ideal vectors.

To train with noise, the maximum number of epochs is reduced to 300 and the error goal is increased to 0.6, reflecting that higher error is expected because more vectors (including some with noise), are being presented.

```
netn = net;
netn.trainParam.goal = 0.6;
netn.trainParam.epochs = 300;
T = [targets targets targets targets];
for pass = 1:10
P = [alphabet, alphabet, ...
      (alphabet + randn(R,Q)*0.1), ...
      (alphabet + randn(R,Q)*0.2)];
[netn,tr] = train(netn,P,T);
end
```

Once the network is trained with noise, it makes sense to train it without noise once more to ensure that ideal input vectors are always classified correctly. Therefore, the network

is again trained without noise.

IV. System Performance and Comparitive Results

The reliability of the neural network pattern recognition system is measured by testing the network with hundreds of input vectors with varying quantities of noise. The script file `appcr1` tests the network at various noise levels, and then graphs the percentage of network errors versus noise. Noise with a mean of 0 and a standard deviation from 0 to 0.5 is added to input vectors. At each noise level, 100 presentations of different noisy versions of each letter are made and the network's output is calculated. The output is then passed through the competitive transfer function so that only one of the 26 outputs (representing the letters of the alphabet), has a value of 1.

The number of erroneous classifications is then added and percentages are obtained.

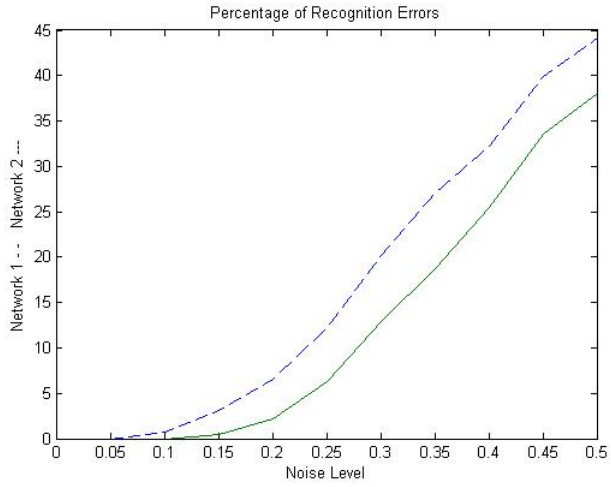

Fig. 4. Performance

The solid line on the graph shows the reliability for the network trained with and without noise. The reliability of the same network when it had only been trained without noise is shown with a dashed line. Thus, training the network on noisy input vectors greatly reduces its errors when it has to classify noisy vectors.

The network did not make any errors for vectors with noise of mean 0.00 or 0.05. When noise of mean 0.2 was added to the vectors both networks began making errors.

If a higher accuracy is needed, the network can be trained for a longer time or retrained with more neurons in its hidden layer. Also, the resolution of the input vectors can be increased to a 10-by-14 grid. Finally, the network could be trained on input vectors with greater amounts of noise if greater reliability were needed for higher levels of noise.

To test the system, a letter with noise can be created and presented to the network.

```
noisyJ = alphabet(:,10)
                  + randn(35,1) * 0.2;
plotchar(noisyJ);
A2 = sim(net,noisyJ);
A2 = compet(A2);
answer = find(compet(A2) == 1);
plotchar(alphabet(:,answer));
```

Figure 5 the noisy letter and the letter the network picked (correctly).

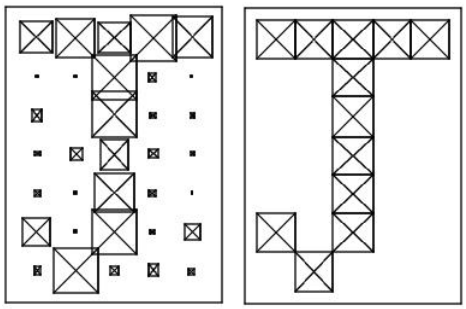

Fig. 5. The noisy input 'J' and the correctly identified version

A. Comparitive study

The network was trained and simulated with varying number of neurons. The results of the computations of untrained network are organized in Figure 6

Performance (P)				
Neurons	**Goal (0.1)**	**Epochs**	**Pure P**	**Noisy P**
2	not met	665	74	296
5	not met	293	26	104
10	not met	3338	1	4
15	not met	2274	14	56
20	not met	1966	15	52
25	not met	1434	6	24

Fig. 6. Performance with respect to number of neurons - untrained network

The trained network was also tested based on the number of neurons. The various performance measures are shown in Figure 7. Network 1 refers to the neural network which is not trained on the noisy data and network 2 represents the neural network trained on both pure and noisy data.

Errors (E)				
Neurons	**Goal (0.1)**	**Epochs**	**Network1 E**	**Network2 E**
2	met	261	1090	974
5	met	223	1178	1021
10	met	370	1146	991
15	met	564	1177	1061
20	not met	1715	1382	1328
25	met	440	1164	1073

Fig. 7. Performance with respect to number of neurons - trained network

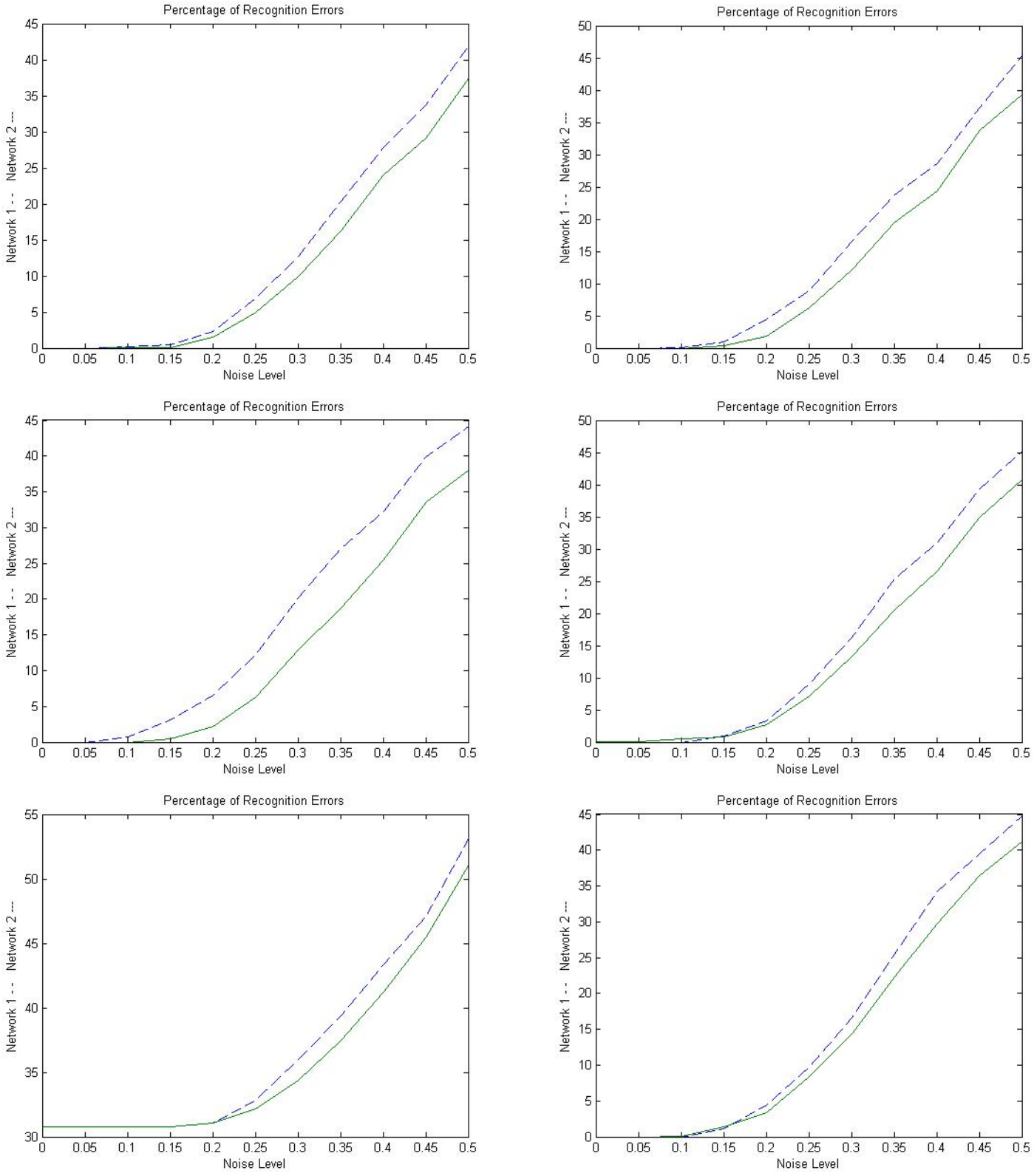

Fig. 8. Simulation results of Network 1(pure) and Network 2(noisy) for different values of neurons (L-R in sequential order) Networks with 2,5,10,15,20 and 25 neurons in hidden layer

B. Our findings

From the simulations, we found that the best performance occurs when the number of neurons in the hidden layer is 10. Figure 8 shows that the maximum error separation between Network 1 and Network 2 occurs when the neurons in the hidden layer is equal to 10.

We can also infer from Figure 7 that the performance is maximized when the number of neurons is 10.

V. Conclusion

This problem demonstrates how a simple pattern recognition system can be designed. Note that the training process did not consist of a single call to a training function. Instead, the network was trained several times on various input vectors.

In this case, training a network on different sets of noisy vectors forced the network to learn how to deal with noise, a common problem in the real world.

Acknowledgement

I would like to thank Prof. S Kumaravel for providing me an opportunity for working on the exciting field of Artificial Neutral Networks (ANN). I would also like to thank Kiran H for the many interesting discussions we had while discussing the implementation.

References

[1] H Demuth, *Neural Network Toolbox 7 User's Guide*, The MathWorks Inc, 1992.

[2] Alexander J. Faaborg, *Using Neural Networks to Create an Adaptive Character Recognition System*, Technical Report, Cornell University, 2002.

[3] Shashank Araokar, *Visual Character Recognition using Artificial Neural Networks*, Technical Report, University of Mumbai, 2005.