EAST WEST UNIVERSITY

Department of Electronics and Communication Engineering

Comparison of Backpropagation and Hopfield Model in De-noising of Speech Signal

Prepared By

Md. Robayet Ferdous

ID: 2015-2-55-030

Mokarromah Akter

ID: 2015-2-55-004

Nowsadul Islam

ID: 2015-2-55-011

Supervised By

Dr. M Ruhul Amin
Department of Mathematical and Physical Sciences,
East West University.

Dr. Md. Imdadul Islam
Department of Computer Science and Engineering,
Jahangirnagar University.

Letter of Transmittal

To
Department of Electronics and Communication Engineering
East West University

Subject: Submission of Project Work Report on Comparison of Backpropagation and Hopfield Model in De-noising of Speech Signal (ETE-498)

Dear Sir,

We are pleased to let you know that we have completed our Project work program on Comparison of Backpropagation and Hopfield Model in De-noising of Speech Signal. The attaché contain the Project work report that has been prepared for your evaluation and consideration. The Project work has given us a great opportunity to work with the de-noising of speech signal closely and allowed us to apply the theoretical knowledge in real-life criteria, which we have acquired since last four years from you and the other faculties of EWU, which would be a great help for us in future.

We are very grateful to you for your guidance throughout the Thesis period, which helped us a lot to acquire knowledge.

Thanking you.

Md. Robayet Ferdous	Mokarromah Akter	Nowsadul Islam
2015-2-55-030	2015-2-55-004	2015-2-55-011

Declaration

We hereby declare that this Project work was done under ETE 498 and has not been submitted elsewhere for the requirement of any degree or diploma or any purpose except for publication.

Md. Robayet Ferdous	Mokarromah Akter	Nowsadul Islam
2015-2-55-030	2015-2-55-004	2015-2-55-011

Acceptance

We hereby declare that this thesis is from the student's work and best effort of us, and all other source of information used, have been acknowledged. This Project work has been submitted with our approval.

Supervisor:

_____

Dr. M Ruhul Amin
Department of Mathematical and Physical Sciences
East West University

Supervisor:

_____

Dr. Md. Imdadul Islam
Department of Computer Science and Engineering
Jahangirnagar University

Chairperson:

_____

Dr. Mohammed Moseeur Rahman
Department of Electronics and Communications Engineering
East West University

Acknowledgement

Firstly, our most heartfelt gratitude goes to our beloved parents for their endless support, continuous inspiration, great contribution and perfect guidance from the beginning to end. We owe our thankfulness to our supervisors for their skilled, almost direction, encouragement and care to prepare ourselves. Our sincere gratefulness for the faculty of Electronics and Communications Engineering whose friendly attitude and enthusiastic support that they have given us for four years. We are very grateful for the motivation and stimulation of our good friends and seniors. We also thank the researchers for their works that help us to learn and implement the comparison of Backpropagation and Hopfield Model in de-noising of the speech signal.

Abstract

In this project work, we have used two algorithms of Neural Network (NN): Backpropagation and Hopfield NN to de-noise speech signal. The backpropagation algorithm is found suitable to remove random noise but very poor in the removal of awgn (Additive White Gaussian Noise). The Hopfield NN shows completely reverse performance i.e. suitable for awgn but very poor for random noise. The performance of both algorithms is measured graphically with an original recovered signal, MSE, the convergence of regression and error histogram.

Table of Content

List of Figures

Chapter **Page no**

List of Abbreviations

NN	Neural Network
ANN	Artificial Neural Network
OCR	Optical Character Recognition
CNN	Convolutional Neural Network
BPN	Backpropagation Algorithm
HNN	Hopfield Neural Network
AWGN	Additive White Gaussian Noise
MSE	Mean Squared Error
AN	Artificial Neuron
ASIC	Application Specific Integrated Circuit
DSP	Digital Signal Processing
GDR	Generalized Delta Rule
SNR	Signal-to-Noise Ratio
PSNR	Peak Signal-to-Noise Ratio

List of Technical Symbols

θ_j^h *and* θ_k^o	Bias Weights
ω_{ji}^h	Weights on the Hidden Layer
h	Quantities on Hidden Layer
o	Quantities on Output Layer
μ	Positive Constant
η	Learning-rate Parameter
E_p	Total Error
net_{pj}^h	Net-inputs to the Hidden Layer units
i_{pj}	Outputs from the Hidden Layer
net_{pk}^o	Net-inputs to the Output Layer units
o_{pk}	Outputs
δ_{pk}^o	Error terms for the Output units
δ_{pj}^h	Error terms for the Hidden units
f_{HL}	Hard Limiter
$\bullet$	Possible Summing Junction
∇E	Energy Gradient Vector
Z^{-1}	Unit Delay

Comparison of Backpropagation and Hopfield Model in De-noising of Speech Signal

Chapter 1

Introduction

Speech is probably the most competent way to talk to each other. Speech is considered as a useful interface for associating with computers as well as human beings is conceivable, analyzed by NN in [1]. Speech signal de-noising is a construction field that reviews approaches used to recover from loud flags a distinctive speech undermined by different types of noise. Noises could be like repetitive sound, clamor, prattle clamor, and many distinct kinds of noise in nature. Other kinds of noise include channel noise that affects both easy and sophisticated transmission, quantization noise resulting from over-pressure of discourse signals, multi-talker babble, resonation noise, or delayed noise form are also present in some circumstances, found in [2]. The addition of substance-based noise is of an arbitrary nature and uncorrelated to discourse, as discussed in [3]. It presents scenarios such as workplaces, cars, fans of town highways, condition of manufacturing line, helicopters and so on in distinct conditions. In the case of an added substance base noise, the suspicions produced for the creation of methods of enhancement are: 1) Speech and noise signals are at any rate uncorrelated over a short timeframe. 2) Noise is stationary or gradually moving more than a few discourse edges and 3) noise can be described as uneven zero-mean procedures, available in [4]. In the event of resonation, speech impressions from various papers will mix in a convoluted model with expression. In this way, if resonation occurs, debasement is subordinate to the flag, while it is free in the event of additional substance-based noise. Speech from distinct speakers may also be mixed in an added substance model with the speech of the perfect speaker. Because the degradation characteristics are unique for each scenario, it may be necessary to process degraded speech in distinct ways. Consequently, for some flag-preparing undertakings, robotized techniques to evacuate the commotion would be a valuable first phase. Noise expulsion from speech signals has been an area of analysts ' excitement during discourse handling over the last centuries and yet there is always space for enhancement. A Neural Network is an information-processing framework, animated by organic sensory systems, comparable to mental process information, found in [5]. Neural networks incorporate basic computing elements operating in parallel, discussed in [1]. The network

function is largely solved by the connections between parts. It is possible to create a neural network so that a specific input guides a specific target output, available in [1]. It is possible to use the neural network in different parts. There are countless uses of NNs that are restricted by our thoughts alone. Development is a response to advances, so we are using NNs to create something that will modernize the world! For composing, there are a few applications for Neural Networks in Speech Recognition, Optical Character Recognition (OCR), Modeling Human Conduct, Example Classification, Loan Hazard Review, Music Age, Image Investigation, Creation of New Fine Arts, Stock Market Expectations, and so on. We used both Backpropagation and Hopfield neural network algorithms in our research job, and we used deep-learning CNN processes to de-noise a predefined voice signal according to the demands of the Matlab environment. Using the Matlab environment characteristics, all the techniques and algorithms mentioned in this article were introduced.

Outline of the project report: The entire project report is organized as: chapter 2 gives basic idea about Artificial Neural Network (ANN). Chapter 3 deals with complete Backpropagation (BPN) algorithm of Artificial Neural Network. Chapter 4 provides complete analysis of Hopfield Neural Network (HNN) with some basic idea of Deep learning. Chapter 5 provides results based on analysis of previous chapters and finally chapter 6 concludes entire analysis.

Chapter 2

Artificial Neural Network

An Artificial Neural Network (ANN) is a model for the processing of information. It is a kind of structure, which is based on the biological nervous system in its processing process. A straightforward explanation of the ANN is that it is a series of linked input / output units with a combined weight. It comprises of a body of easy processing components that interact through a big amount of weighted links by sending signals to each other, as discussed in [6]. It is inspired by a biological nervous system whose basic unit is the biological neuron shown in Figure 2.1. It has been created as a generalization of neural biology mathematical models.

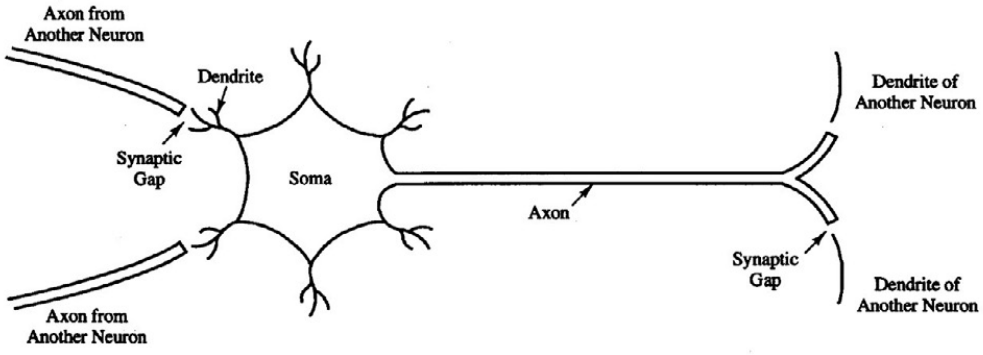


Figure 2.1: A Biological Neuron [7]

Since an ANN is comparable to a biological neural network, the mathematical model of a neuron whose schematic is shown in Figure 2.2 is its basic construction block.

2.1 Fundamental Parts of AN:

The Artificial Neuron (AN) has three fundamental parts and they are:

- Synapses or linking connections to the input values, w_{kp}, x_p for $p = 1,...,n$

- An adder part that sums the weighted input values and calculates the activation function input.
- Activation function mapping the amount of x_p to y_k, the neuron's output value. It has also called a function of squashing.

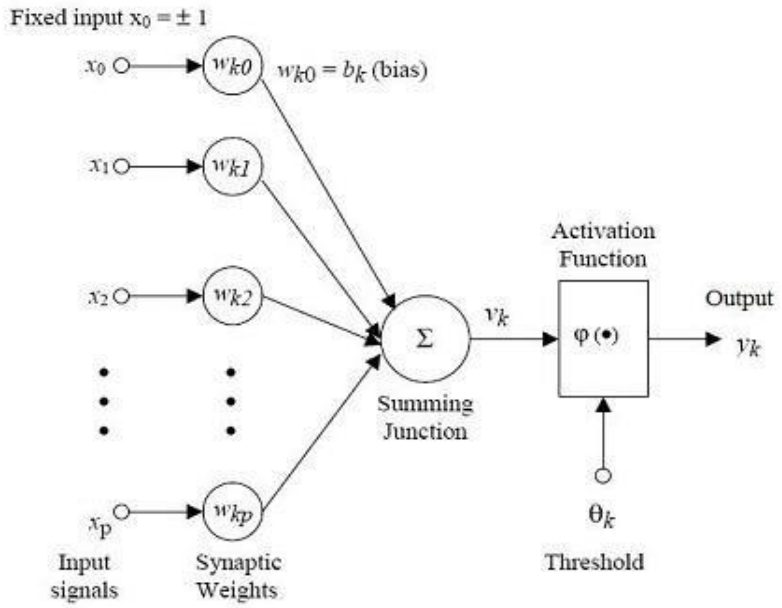


Figure 2.2. Schematic of an Artificial Neuron. [8]

ANN discovers its usefulness in various sectors such as aerospace, defense, electronics, manufacturing, medical, robotics, voice and transport in classification, pattern recognition, data mining, control, optimization, and so on. ANN's energy system implementation fields include load prediction, fault diagnosis / fault place, economic dispatch, safety evaluation, and transient stabilization.

2.2 Benefits and Drawbacks of ANN:

As mentioned in [8], the benefits of ANN are:

- Its ability to support a non-linear mapping of input and output variables.

- It also has functions such as solid performance in noisy settings and incomplete information environments that allow it to generalize.
- High parallel computing, which implies it can be implemented with very large-scale embedded circuits such as ASIC, DSP, etc.
- Its capacity to learn on its own (with or without a supervisor) using information input into the template, making it an adaptive method (generally accomplished by altering the link strengths).
- If a model neuron is damaged throughout the process, the entire network will not be shut down; it can still work very well.
- With a measure of confidence, it can create choices.

Authors of [9] provide some drawbacks of ANN:

- ANN's processing time improves with its size.
- ANN requires to be trained before starting operation.
- The ANN architecture varies from the microprocessor architecture.

Chapter 3

Backpropagation

A neural system is known as a mapping system in the event that it can register some useful connection between its info and its output. For instance, if the contribution to a system is the estimation of a point and the output is the cosine of that edge, the system plays out the mapping $\Theta \rightarrow cos\theta$. For such a straightforward capacity, we need not bother with a neural system; nevertheless, we should need to play out a confused mapping where we do not know to portray the utilitarian relationship ahead of time however we do know about instances of the right mapping.

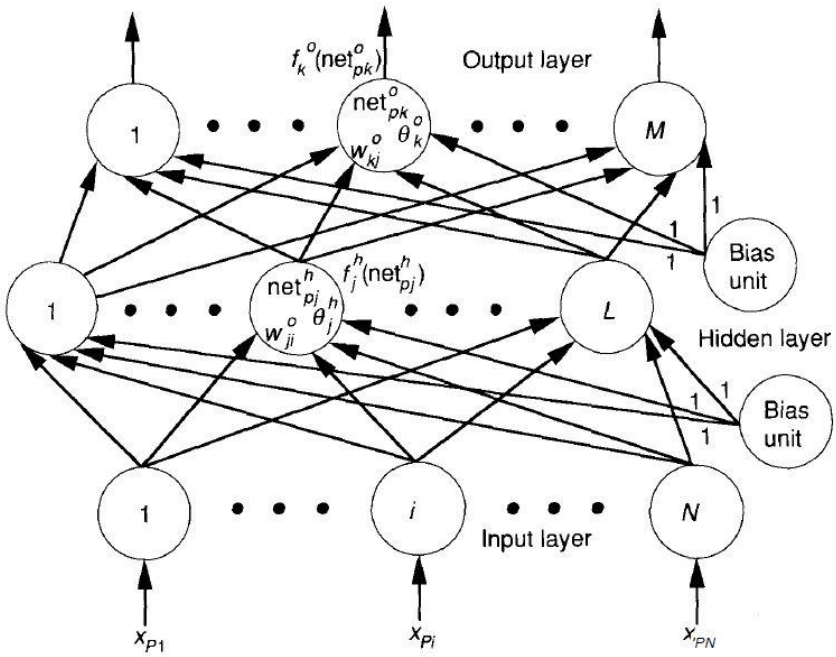


Figure 3.1. The BPN architecture of Three Layers

Here, The weights of bias, amble θ_j^h, and range θ_k^o, and the units of bias are optional. The bias units on a connection to the bias weight provide a fictional input value of 1. The bias weight (or merely bias) can then be treated as any other weight. It adds to the unit's net input value and it participates as any other weight in the learning process.

Here the $X_p = (x_{p1}, x_{p2}, \ldots, x_{pN})^t$ data vector is linked to the system's info layer. The units of information distribute the qualities to the units of the hidden-layer. The net input to the j^{th} hidden unit is,

$$net_{pj}^{h} = \sum_{i=1}^{N} \omega_{ji}^{h} x_{pi} + \theta_j^h \quad 3.1$$

Where, ω_{ji}^{h} is the weight on the connection, from the i^{th} input unit, and θ_j^h is the bias term. On the hidden-layer, the "h" superscript alludes to quantities. Assuming that this activated node is to be initiated as net input, then the output of this node would be,

$$i_{pj} = f_j^h(net_{pj}^{h}) \quad 3.2$$

The equation for the output nodes is,

$$net_{pk}^{o} = \sum_{j=1}^{L} \omega_{ki}^{o} x_{\text{pi}} + \theta_k^o \quad 3.3$$

$$o_{pk} = f_k^o(net_{pk}^{o}) \quad 3.4$$

Where amounts are referred to on the output layer in the "o" superscript. The fundamental weight esteem scheme refers to a first conjecture about the best possible loads for the problem. The technique we use here does not depend on creating a good first speculation as opposed to a few approaches. Nevertheless, there are rules for selecting the underlying loads, and we will discuss them in condition 3. The accompanying depiction illustrates the vital method for preparing the system:

1. Apply a system info vector and calculate the related output values.
2. Compare the actual outputs with the correct outputs and determine a share of the error.
3. Determine the course (+ or-)to modify each weight in order to decrease the inaccuracy.
4. Determine the amount by which each weight is changed.
5. Use the adjustments to the loads.
6. Keep repeating the same thing from 1 to 5 with all the preparation vectors until the err or in the preparation set for all vectors decreases to a dignified esteem.

In past we depicted an intuitive weight-change law for system with no concealed units and straight out units, called delta rule.

$$w(t+1)i = w(t)i + 2\mu\varepsilon_k x_{ki} \quad 3.5$$

3.1. Update of Output-layer Weights:

In a single output unit, we will characterize the error to be δ_{pk} = (ypk — opk), where the subscription "p" relates to the pth preparation vector, and "k" relates to the k^{th} output unit. y_{pk} is the ideal output esteem in this case, and o_{pk} is the real output from the kth unit. The GDR-limited error is the aggregate of the error squares for all output units:

$$Ep = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^2 \qquad 3.6$$

The factor of $\frac{1}{2}$ in this Eq is there for comfort in ascertaining subsidiaries afterward. Since a subjective consistent will show up in the last outcome, the nearness of this factor does not negate the deduction. We ascertain the negative of the inclination of E_p, ∇E_p ,as for the loads,w_{kj}.
We consider each element of ∇E_p separately to maintain stuff straightforward.

From Eq. 3.6 and the definition of δ_{pk},

$$Ep = \frac{1}{2}\sum_{k=1}^{M} \left(y_{pk} - o_{pk}\right)^2 \qquad 3.7$$

$$= \frac{1}{2}\sum_{k=1}^{M} \{y_{pk} - f_k^o(net_{pk}^o)\}^2$$

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk})\frac{\partial f_k^o}{\partial (net_{pk}^o)}\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} \qquad 3.8$$

Where y_{pk} is the desired output.
Where we used the Eq. 3.4 Output value, o_{pk}, and partial derivatives chain rule. For this occasio n, we will not attempt to evaluate the f f_k^o subsidiary, yet rather will compose it essentially as $f_k^{o1}(net_{pk}^o)$. The last factor in Eq. 3.7 is

$$\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} = \left(\frac{\partial}{\partial w_{kj}^o}\sum_{j=1}^{L} \omega_{ki}^o x\text{pi} + \theta_k^o\right) = i_{pj} \qquad 3.9$$

Combining Eq. 3.7 and 3.8, we have for the negative gradient

$$-\frac{\partial E_p}{\partial w_{kj}^o} = \left(y_{pk} - o_{pk}\right) f_k^{o1}(net_{pk}^o) i_{pj} \qquad 3.10$$

As far as the magnitude of the weight change is concerned, we take it to be proportional to the negative gradient. Thus, the weights on the output layer are updated according to

$$w_{kj}^{o}(t+1)i \; = w_{kj}^{o}(t) + \Delta_p w_{kj}^{o}(t) \qquad 3.11$$

Where

$$\Delta_p w_{kj}^{o} = \eta\big(y_{pk} - o_{pk}\big) f_k^{o1}(net_{pk}^{o}) i_{pj} \qquad 3.12$$

The factor η is called the learning-rate parameter.

Let us go back to look at the function f_k^{o1}.

Two types of the output function are of concern here:

- $f_k^{o1}\big(net_{pk}^{o}\big) = (net_{pk}^{o})$
- $f_k^{o1}\big(net_{pk}^{o}\big) = (1 + e^{-net_{jk}^{o}})^{-1}$

In the first case, f_k^{o1} = 1; in the second case, $f_k^{o1} = f_k^{o}(1 - f_k^{o}) = o_{pk}(1 - o_{pk})$; for these two cases, we have

$$w_{kj}^{o}(t+1)i \; = w_{kj}^{o}(t) + \eta\big(y_{pk} - o_{pk}\big) i_{pj} \qquad 3.13$$

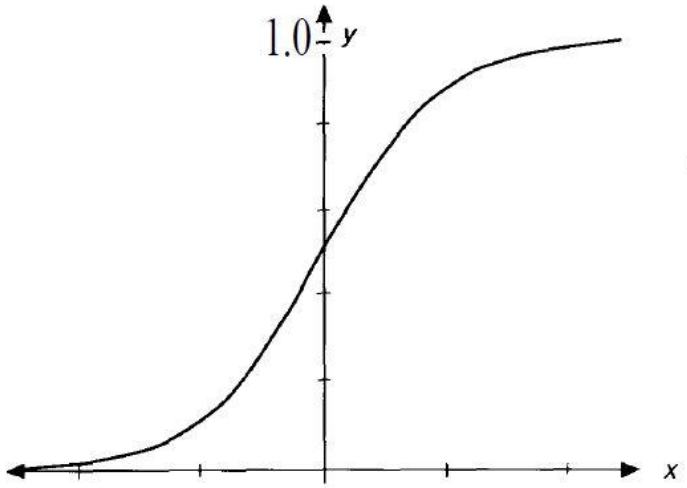


Figure 3.2. The Sigmoid Function's distinctive S-shape.

For the linear output, and

$$w_{kj}^{o}(t+1)i \; = w_{kj}^{o}(t) + \eta\big(y_{pk} - o_{pk}\big) o_{pk}\big(1 - o_{pk}\big) i_{pj} \qquad 3.14$$

For the sigmoidal output.

By identifying a amount, we want to summarize the weight update equations

$$\delta_{pk}^{o} = \left(y_{pk} - o_{pk}\right) f_k^{o1}\left(net_{pk}^{o}\right)$$
$$= \delta_{pk} f_k^{o1}\left(net_{pk}^{o}\right) \quad 3.15$$

We can then write the equation for weight updates as

$$w_{kj}^{o}(t+1)i \; = w_{kj}^{o}(t) + \eta \delta_{pk}^{o} i_{pj} \quad 3.16$$

Regardless of the output substring functional form, f_k^o.

3.2. Updates of Hidden-Layer Weights:

We may want to rehash a comparable kind of computation for the hidden-layer as we did for the output layer. A problem arises when we try to determine a measure of the error of the hidden-layer unit's outputs. We understand what the real output is, but we don't have the opportunity to know in advance what the correct output should be for these units.. Of course, the absolute error, E_p, should be recognized with the yield esteems on the hidden-layer in one manner or another. By returning to Eq, we can verify our instinct. 3.7:

$$Ep = \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)^2$$
$$= \frac{1}{2}\sum_{k=1}^{M}\{y_{pk} - f_k^o(net_{pk}^o)\}^2$$
$$= \frac{1}{2}\sum_{k=1}^{M}\{y_{pk} - f_k^o(\sum_j \omega_{ki}^o \, x\mathrm{pi} + \Theta_k^o)\}^2$$

We comprehend that i_{pj} needs to rely through Eq. on the weights of the hidden-layer (3.1) and (3.2) respectively. We can take advantage of this reality to calculate the E_p gradie nt for the weights of the hidden-layer.

$$\frac{\partial E_p}{\partial w_{kj}^o} = \frac{1}{2}\sum_k \frac{\partial}{\partial w_{kj}^o}\left(y_{pk} - o_{pk}\right)^2$$
$$= -\sum_k \left(y_{pk} - o_{pk}\right) \frac{\partial o_{pk}}{\partial (net_{pk}^o)} \frac{\partial (net_{pk}^o)}{\partial i_{pj}} \frac{\partial i_{pj}}{\partial (net_{pj}^h)} \frac{\partial (net_{pj}^h)}{\partial w_{kj}^h} \quad 3.17$$

Each of the factors in Eq. (3.17) can be calculated explicitly from previous equations. The result is,

$$\frac{\partial E_p}{\partial w_{kj}^o} = -\sum_k \left(y_{pk} - o_{pk}\right) f_k^{o1}(net_{pk}^o) w_{kj}^o f_k^{h1}(net_{pj}^h) x_{pi} \quad 3.18$$

We update the weights of the hidden-layer in proportion to the Eq's negative. (3.18):

$$\Delta_p w_{ji}^h = \eta f_k^{h1}(net_{pj}^h) x_{pi} \sum_k \left(y_{pk} - o_{pk}\right) f_k^{o1}(net_{pk}^o) w_{kj}^o \quad 3.19$$

Where η is once again the learning rate.

We can utilize the meaning of δ_{pk}^o given in the past segment to compose

$$\Delta_p w_{ji}^h = \eta f_k^{h1}(net_{pj}^h) x_{pi} \sum_k \delta_{pk}^o w_{kj}^o \quad 3.20$$

Note that every weight update on the hidden-layer is based on all the inaccurate terms, δ_{pk}^o, on the yield layer. This outcome is the place where the idea of backpropagation emerges. The know n blunders on the yield layer are proliferated back to the hidden-
layer to determine the fitting weight changes on that layer. By characterizing a hidden-
layer of error term

$$\delta_{pj}^h = f_k^{h1}(net_{pj}^h) \sum_k \delta_{pk}^o w_{kj}^o \quad 3.21$$

We cause the weight-updating conditions to be similar to those for the output layer.

$$w_{ji}^h(t+1) = w_{ji}^h(t) + \eta \delta_{pk}^h x_i \quad 3.22$$

3.3. BPN Summary:

We are collecting all the important conditions for the BPN here to reduce the need to flip pages to find the right conditions. They are shown in the appropriate order in which they would be used in the preparation of a single training-vector pair.

1. Apply the input vector, $X_p = (x_{p1}, x_{p2}, \ldots, x_{pN})^t$ to the input units.
2. Calculate the net-input values to the hidden-layer units:

$$net_{pj}^h = \sum_{i=1}^N \omega_{ji}^h x_{pi} + \theta_j^h$$

3. Calculate the outputs from the hidden-layer:

$$i_{pj} = f_j^h(net_{pj}^h)$$

4. Move to the output layer. Calculate the net-input values to each unit:

$$net_{pk}^o = \sum_{j=1}^L \omega_{ki}^o x_{pi} + \theta_k^o$$

5. Calculate the outputs:

$$o_{pk} = f_k^o(net_{pk}^o)$$

6. Calculate the error terms for the output units:

$$\delta_{pk}^o = (y_{pk} - o_{pk}) f_k^{o1}(net_{pk}^o)$$

7. Calculate the error terms for the hidden units:

$$\delta_{pj}^h = f_k^{h1}(net_{pj}^h) \sum_k \delta_{pk}^o w_{kj}^o$$

Here we can see that the error terms are calculated on the hidden units before updating the connection weights to the output layer units.

8. Update weights on the output layer:

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta \delta_{pk}^o i_{pj}$$

9. Update weights on the hidden-layer:

$$w_{ji}^h(t+1) = w_{ji}^h(t) + \eta \delta_{pk}^h x_i$$

The order of the weight updates on an individual layer is not important. So, we have to calculate the error term using,

$$Ep = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^2$$

Because this quantity is the indicator of how well the network is learning, instruction can be stopped when the error is acceptably low for each of the training vector pairs.

Chapter 4
Hopfield Neural Network

In 1982, at the California Institute of Technology and AT&T Bell Laboratories, John J. Hopfield conceptualized a model consistent with the asynchronous nature of biological neurons. As such, in contrast to the perception, it was a more abstract, fully interconnected, random and asynchronous network that required a clock to synchronize its circuit operation, similar to a digital computer. Generally speaking, the Hopfield network is a fully connected auto associative network of a single node layer. It is also a symmetrically weighted network. The network takes two-value inputs: binary (0 1) or bipolar (+1 -1); the bipolar simplifies the analysis of mathematics.

Hopfield's fundamental model unit has a two-output processing component, one non-inverting and one inverting element (Figure 4.1). Each processing element's outputs is linked back to any other processing element's inputs expect itself. The connections are resistive (a resistor parallel to a capacitor) and represent the strength (weight), ω_{ij} of the connection. Since no negative resistors are present, excitatory connections use positive outputs, and inhibitory connections use inverted outputs. Connections are made excitatory when the processing element output is the same as the input; when the inputs differ from the processing element output, they are inhibitory. Although all outputs are shown in (Figure 4.1) to be fed back, one of them will eventually make the connection. The sigmoid was used as the nonlinearity. A link between processing elements *i* and *j* is connected with a connecting force ω_{ij} which, if positive, depicts the situation where, if unit *i* is on, unit *j* is also on (i.e., some sort of excitatory synapse). If the strength of the connection is negative, it represents the situation where unit *j* is not on when unit *i* is on. In addition, the weights are symmetric: the strength of connection ω_{ij} is the same as ω_{ji}.

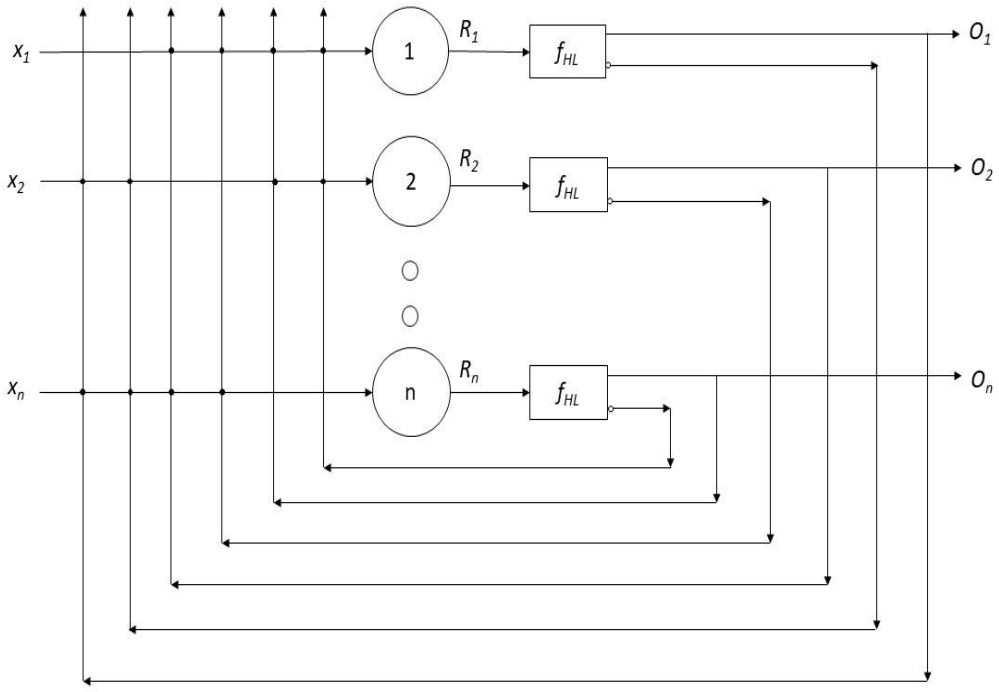


Figure 4.1. Basic Hopfield Paradigm

Here, f_{HL} = hard limiter and $\bullet$ = possible summing junction

Hopfield defined his model as an energy function that relies on the state of interconnected neurons *j* with *i*, the state of firing *V*, and their strength of connections, ω_{ij}:

$$E = -\frac{1}{2}\sum\sum_{i\neq j}\omega_{ij}\, V_i\, V_j \qquad 4.1$$

He stated that changes in V_i monotonically decrease E until a minimum is reached. By differentiating *E* over V_i, the latter is mathematically verified:

$$\frac{\delta E}{\delta V_i} = -\sum_{j\neq i}\omega_{ij}\, V_j \qquad 4.2$$

Equation 4.2 produces all minima, local and global. However, we are not interested in local minima, as they are not necessarily given true results (targets). The neural network must

therefore be able to escape local minima and settle to the global minimum, i.e. generate real results. The Hopfield network has found interesting applications by calculating the weighted sum of the inputs and quantizing the outputs. An analog-to-digital converter was shown on the basis of this.

4.1. Mathematical Analysis:

The output before nonlinearly R_i of the i th neuron is equation for a n-neuron single-layer Hopfield network.

$$R_i = \sum_j^n \omega_{ij} O_j + x_i - \Theta_i \qquad \text{for i = 1,2,……..n.} \qquad 4.3$$

Where x_i is the external input to the i th node, O_j is the output of the neurons (after nonlinearity), Θ_i is the threshold of the i th neuron, and ω_{ij} is the connection weights.

Equation 4.3, for one neuron in vector notation

$$R_i = w_i^T O + x_i - \Theta_i \qquad \text{for } i = 1, 2,…..n. \qquad 4.4$$

Where the weight vector and the output vector of the i th neuron are

$$w_{ij} = \begin{bmatrix} \omega_{i\,1} \\ \omega_{i\,2} \\ . \\ . \\ . \\ \omega_{i\,n} \end{bmatrix}, \qquad O = \begin{bmatrix} O_1 \\ O_2 \\ . \\ . \\ . \\ O_n \end{bmatrix} \qquad 4.5$$

Considering all the nodes, we can write

$$R = WO + x - \Theta. \qquad 4.6$$

Where,

$$R = \begin{bmatrix} R_1 \\ R_2 \\ . \\ . \\ . \\ R_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}, \qquad \text{and} \quad \Theta = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ . \\ . \\ . \\ \Theta_n \end{bmatrix} \tag{4.7}$$

Where weights can be displayed in an n × n symmetric matrix (i.e., $w_{ij} = w_{ji}$) with diagonal terms $\omega_{ii} = 0$:

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ . \\ . \\ . \\ w_n^T \end{bmatrix} \tag{4.8}$$

Moreover, w_i^T is in the form of the *n*-vector.

Now, if the activation function is sgn(.), the outputs O_i take values (+1, -1). If $R_i < 0$ or $O_i = +1$ if $R_i > 0$, the response of the network will be $O_i = -1$ if $R_i < 0$ or $O_i = +1$ if $R_i > 0$.
To study the network's stability characteristics, one starts with the energy function.

$$E = -\frac{1}{2} O^T W O - x^T O + \Theta^T O . \tag{4.9}$$

The energy gradient vector is

$$\nabla E = -\frac{1}{2} (W^T + W) O - x^T + \Theta^T. \tag{4.10}$$

Outputs are updated asynchronously during learning (i.e., only one at a moment, here the *i* th). Then

$$\Delta O = \begin{bmatrix} 0 \\ . \\ . \\ . \\ \Delta O_i \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \qquad 4.11$$

The increase in energy reduces to the form

$$\Delta E = (- w_i^T O - x_i^T + \theta_i) \Delta O_i \qquad 4.12$$

Or

$$\Delta E = - (\sum_{j=1}^{n} \omega_{ij} O_j + x_i - \theta_i) \Delta O_i \qquad \text{for } j \neq i . \qquad 4.13$$

4.2. The Hopfield Learning Algorithm

1. Assign random connection weights with values ω_{ij} = +1 or -1 for all $i \neq j$ and 0 for $i = j$ (i.e. all diagonal values are zero – an asynchronous update requirement).
2. Initialize the network with an unspecified sequence: $x_i = O_i$ (k), $0 \leq i \leq N - 1$, where O_i (k) is the output of the node i at the moment $t = k = 0$ and x_i is an element at the input i of the input pattern, + 1 or -1; i.e., the input pattern consists of the signs + 1 and -1 and the nodes threshold is zero.
3. Iterate until it reaches convergence, using the relationship

$$O_i (k+1) = f (\sum_{i=0}^{N-1} \omega_{ij} O_i (k)) , \qquad 0 \leq j \leq N - 1, \qquad 4.14$$

   Where function f (.)is a non-linearity that is hard-limiting. Repeat the process until the outputs of the node stay the same. Then the node outputs best depict the model pattern that best fits the unknown input.
4. Return to step 2 and repeat for the next x_i, and so on.

4.3. Discrete-Time Hopfield Network

The *i* th output is for a recurrent Hopefield network (Figure 4.2) at a discrete time is,

$$O_i\,(\mathrm{k+1}) = f_{HL}\,(R_i\,(\mathrm{k})) \qquad 4.15$$

Where,

$$R_i\,(\mathrm{k}) = \sum_{j=1}^{n} \omega_{ij}\, O_j + x_i - \Theta_i \qquad 4.16$$

And k is the recursive process index. The recursion begins with the initialization vector $O(0)$, resulting in $O_i(1)$ for the first iteration.

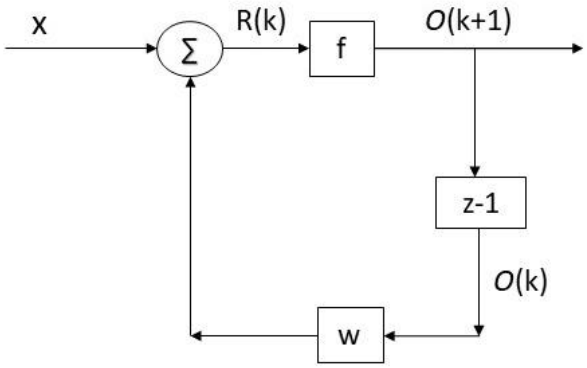


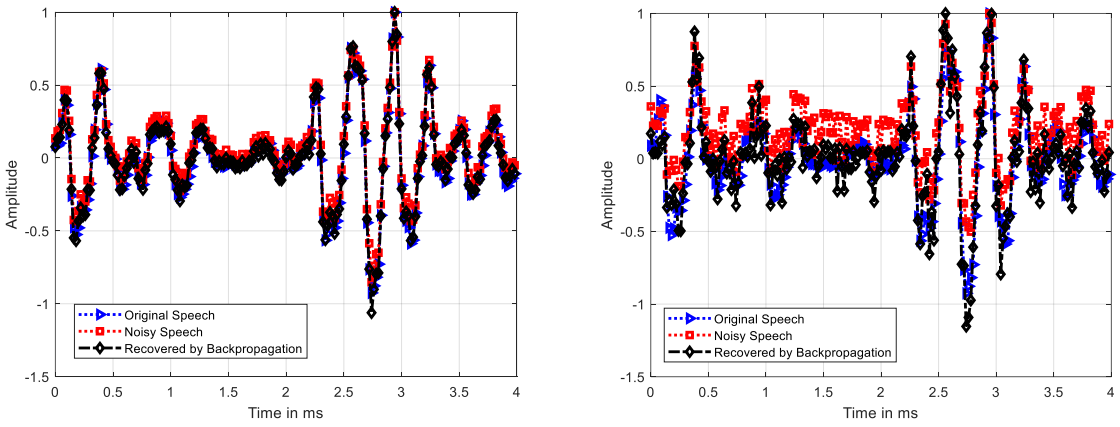
Figure 4.2. Discrete time Hopfield model

Here, Z^{-1} = unit delay

Chapter 5

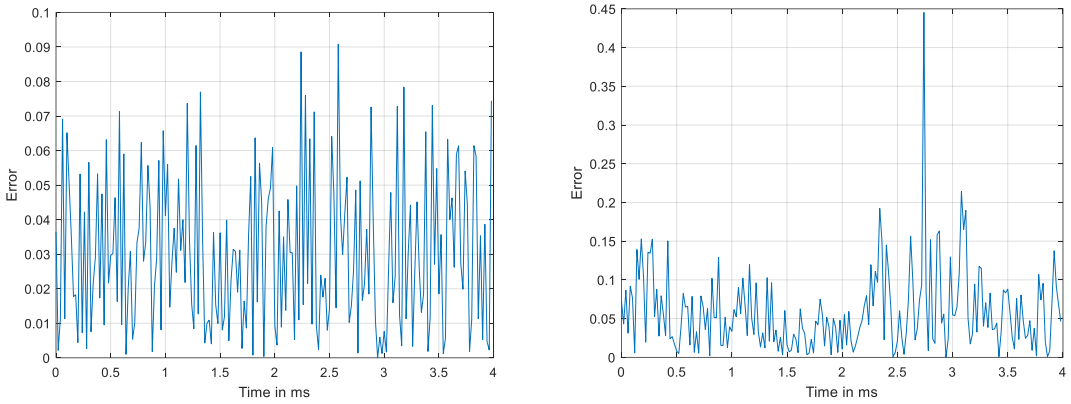
Results and Discussion

First, we consider random noise to contaminate the original speech signal of 100 samples. Figure 5.1. (a) and (b) shows the original, noisy and recovered signal at 12 dB and 6 dB respectively applying Backpropagation algorithm with 10 hidden-layers. The corresponding absolute error between original and recovered signals is show in figure 5.2. (a) - (b) and the mean square error of validation is shown in figure 5.3. (a) - (b). The error histogram of both cases are shown in figure 5.4. (a) - (b) for 20 bins and very few samples are found to exceed the error of 8%. Finally, the convergence data are shown with linear regression in figure 5.5. (a) - (b). The performance is found better in the entire diagram figure 5.1-5.5 for 12dB case compared to 6dB. Next, we consider awgn instead of random noise where the performance of BP is found poor as shown in figure 5.6. The Hopfiled network shows better performance under awgn as shown in figure 5.7. but poor performance under random noise shown in figure 5.8.



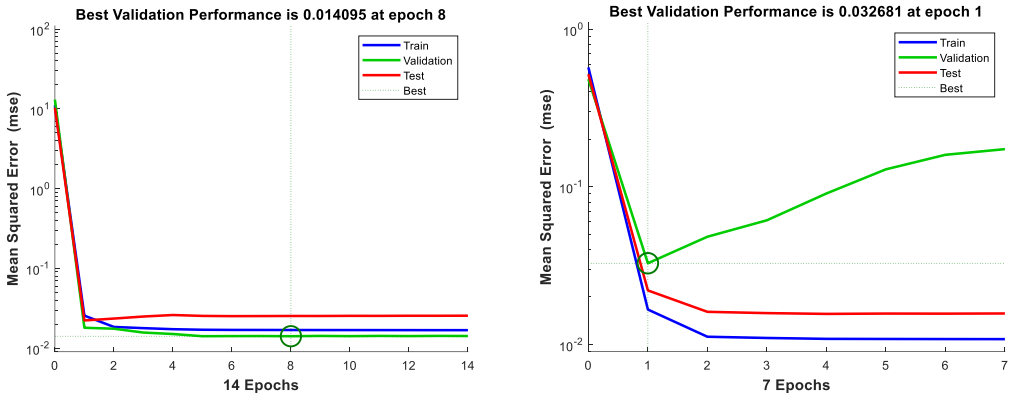
(a) SNR of 12 dB (b) SNR of 6 dB

Figure 5.1. Comparison of original, noisy and recovered signal under BPN



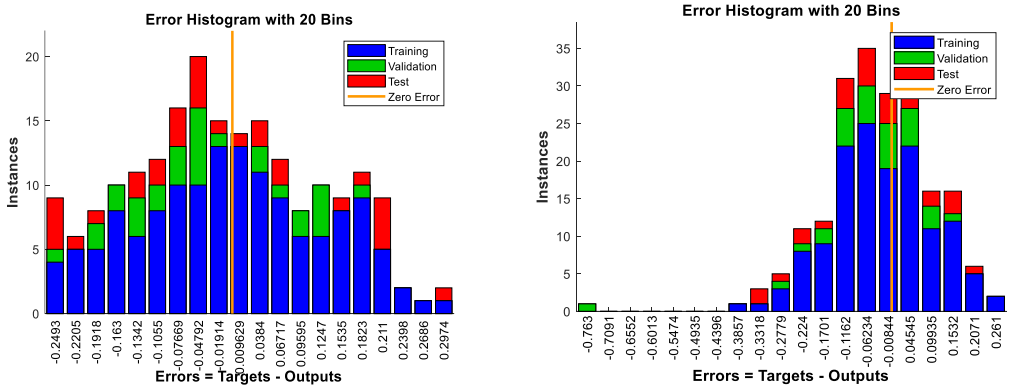
(a) SNR of 12 dB (b) SNR of 6 dB

Figure 5.2. Error signal of BPN under random noise



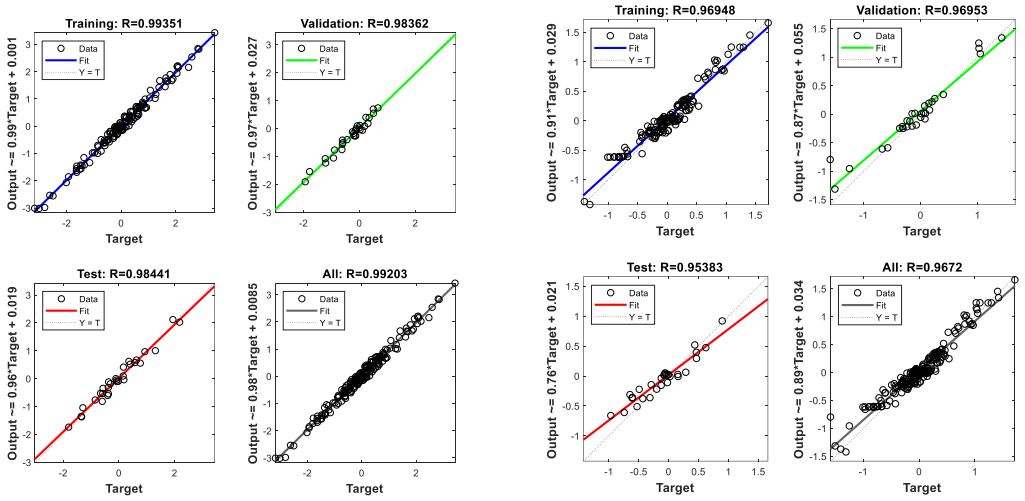
(a) SNR of 12 dB (b) SNR of 6 dB

Figure 5.3. MSE of train and validation



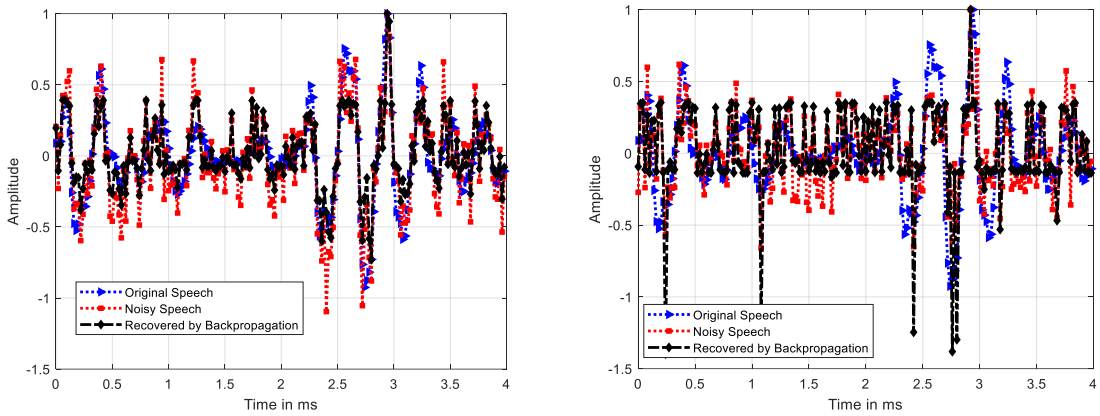
(a) SNR of 12 dB (b) SNR of 6 dB

Figure 5.4. Comparison of error histogram



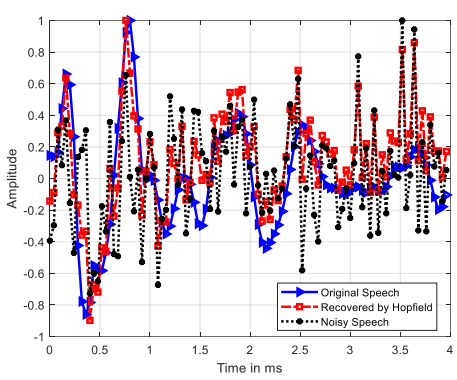
(a) SNR of 12 dB (b) SNR of 6 dB

Figure 5.5. Comparison of convergence of original and recovered data

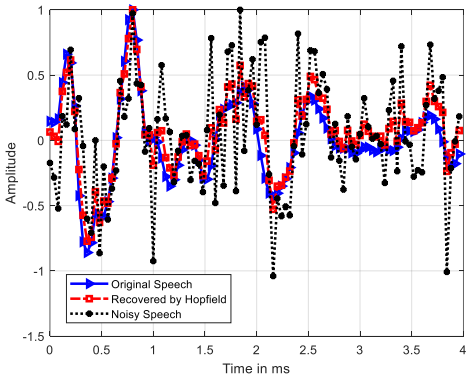


(a) SNR of 12 dB (b) SNR of 6 dB

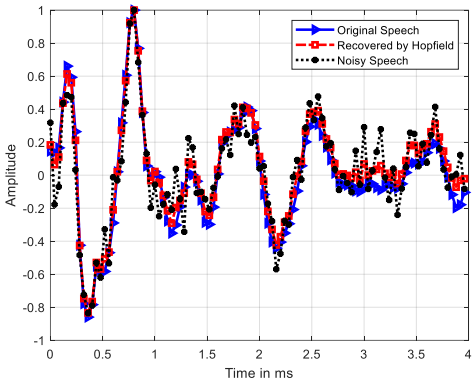
Figure 5.6. Poor performance of BPN under awgn



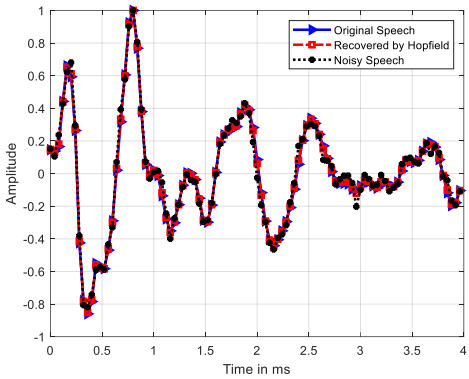
(a) 12 dB



(b) 20 dB

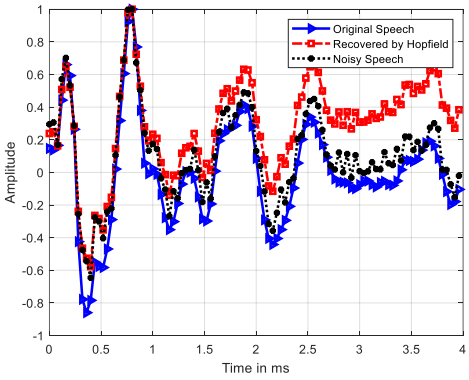


(c) 30 dB

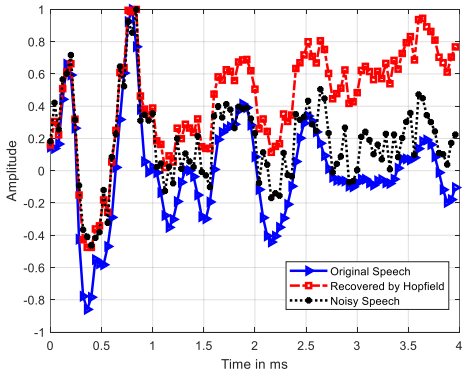


(d) 40 dB

Figure 5.7. Good Performance of Hopfield under awgn



SNR of 12 dB



(b) SNR of 6 dB

Figure 5.8. Poor performance of Hopfield under random noise

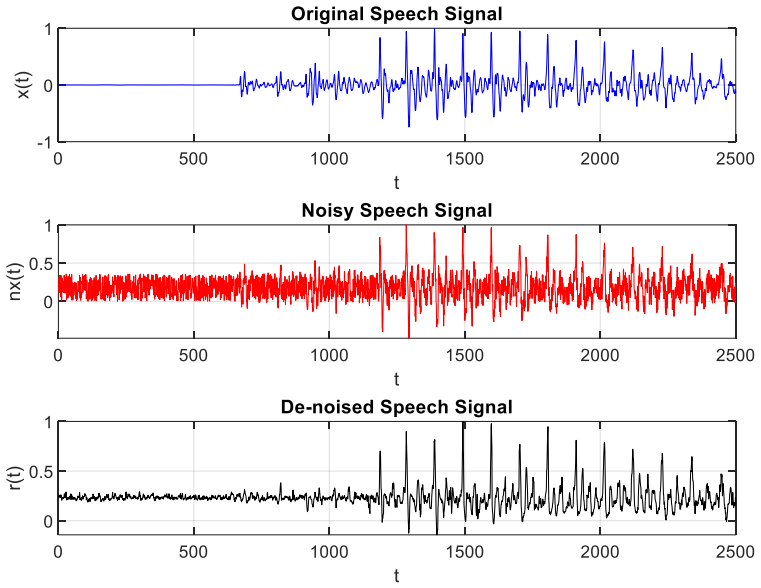


Figure 5.9. De-noising of speech signal by deep learning CNN under random noise

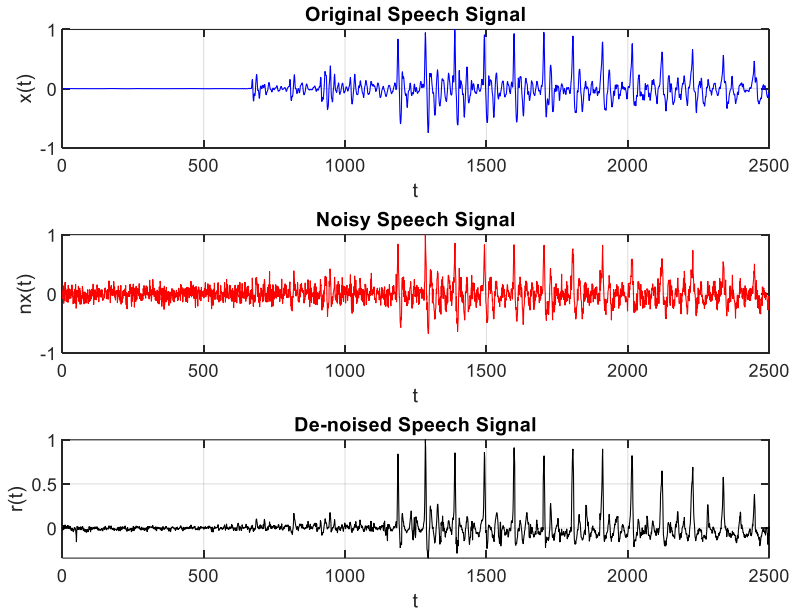


Figure 5.10. De-noising of speech signal by deep learning CNN under awgn

Finally we apply deep learning CNN on noisy speech signal (2500 samples, which is the minimum requirement of MATLAB) to recover the original signal. Figure 5.9. shows the original speech signal, noisy speech signal and de-noised signal under random noise. Similar results are shown in figure 5.10. under awgn. Here we add awgn of 20 dB and random noise with amplitude of 50% of peak amplitude of the speech signal. The deep learning CNN can resolve both awgn and random noise; where we found the cross correlation co-efficient of 84.3% and 85.67% for awgn and random noise case respectively.

Chapter 6

Conclusion and Future Works

In this research project, we mainly emphasize on Backpropagation algorithm of Neural Network and Hopfield Neural Network in de-noising of speech signal and a comparison is drawn between this two algorithms. Using the same algorithms, we can also de-noise digital images converting them from 2D matrix to 1D vector. Then a comparison can be made between these two algorithms in de-noising of different types of noises like Gaussian, Salt-and-pepper, Shot, Quantization (uniform noise), Anisotropic and Periodic noise etc. We can also include different types of digital filters like Median, Gaussian, Gabor, Mean and Wiener filter etc. to observe the improvement of the signal. Then we can measure the quality of the image/speech signal using SNR and PSNR. In future, we will apply the similar concept on Convolutional Neural Network (CNN).

References:

[1] Wouter Gevaert, Georgi Tsenov, Valeri Mladenov, “Neural Networks used for Speech Recognition”, Journal of Automatic Control, University of Belgrade, VOL.20, pp.1-7, January 2010.

[2] Soon Ing Yann, “Transform based Speech Enhancement Techniques”, PhD Thesis Nanyang Technological University, April 2003.

[3] P Krishnamoorthy, Mahadeva Prasanna, “Processing Noise Speech for Enhancement”, IETE Technical Review, Volume 24, No 5, pp.351-357, Sept-Oct 2007.

[4] S. Boll, “Suppression of acoustic noise in speech using Spectral Subtraction”, IEEE transactions on Acoustic Speech and Signal Processing, Vol-ASSP.27, pp.113-120, April 1979.

[5] Preet Ind, Gour Sundar Mitra, Thakurer Singh, “Enhanced Password Based Security System Based on User Behavior using Neural Networks”, Journal of Information Engineering and Electronic Business, Vol.2, pp.29-35, April 2012.

[6] Learning Artificial Neural Networks (ANN) [Internet].2016 [cited 2016 Aug 9]. Available from: Crossref .

[7] Haque MT, Kashtiban AM, “Application of neural networks in power systems: a review. World Academy of Science, Engineering and Technology”, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering. Vol-1(6), pp.889–93, 2007.

[8] Adepoju GA, Ogunjuyigbe SOA, Alawode KO. “Application of neural network to load forecasting in Nigerian electrical power system”, The Pacific Journal of Science and Technology. Vol.8 (1), pp.68–72, 2007.

[9] Hsu YY, Yang L-C. “Design of artificial neural networks for short term load forecasting”, In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)– C Generation, Transmission and Distribution, Vol.138(5), pp.407–13, 1992.