DEFECTS INSPECTION PROCESS OF FLAT PANEL DISPLAY FABRICATION THROUGH NEURAL NETWORKS

March 2015

HAPU ARACHCHILAGE ABEYSUNDARA

Department of System Cybernetics
Graduate School of Engineering, Hiroshima University, Japan

Doctoral Dissertation

DEFECTS INSPECTION PROCESS OF FLAT PANEL DISPLAY FABRICATION THROUGH NEURAL NETWORKS

By

HAPU ARACHCHILAGE ABEYSUNDARA

Supervisors

Professor	Dr. Masatoshi Sakawa
Professor	Dr. Ichiro Nishizaki
Professor	Dr. Katsuhiko Takahashi
Associate Professor	Dr. Katsumi Morikawa

March 2015

Department of System Cybernetics
Graduate School of Engineering, Hiroshima University, Japan

This doctoral dissertation is submitted to the Department of System Cybernetics, Graduate School of Engineering, Hiroshima University, Japan for partial fulfillment of requirements of Doctor of Engineering in System Cybernetics.

HAPU ARACHCHILAGE ABEYSUNDARA

March 2015

To my daughter Charu and son Minoru

Preface

Modern automated factory environments, especially flat panel display (FPD) assembly lines, require a very cost-intensive and sophisticated machinery. With the huge investment involved and with the stiff competition among manufacturers for high throughput product lines and low priced manufacturing, optimization of every possible task in production lines is always paramount. Thin film transistor (TFT) array fabrication process on a glass substrate is one of the most important steps in FPD fabrication. During fabrication of TFT array lines on glass substrates, electrical defects such as open circuits and short circuits often exist on them that have to be inspected and detected in early manufacturing stages in order to repair and restore them. Recently, manufacturers and researchers have started more attention and R&D initiatives on defects inspection and repairing of TFT arrays in order to maximize production yield and to stay competitive.

Currently, the most advanced technique of detecting such electrical defects is the non-contact inspection method by a capacitor based sensor. The sensor scans TFT lines, while a known voltage is applied into TFT lines on glass substrates through a feeding electrode and is captured through a receiving electrode. Captured voltage waveforms are digitized through an analogue to digital converter and taken into analysis. The detection of defects is based on analyzing peaks and troughs on those waveforms by using a known thresholding value. However determining a proper threshold for correctly identifying such peaks and troughs on waveforms is still not easy as the measured voltage signals are mixed with various noises such as random noises, external vibrations and noises due to environmental effects such as fluctuations of machine temperature.

Once such defective TFT lines are detected on glass surface of flat panel displays, a camera based sensor, called NG sensor, is used to scan them in order to determine exact positions of defect points. Currently, this scanning method is a mere primitive top-down unidirectional path which is heavily time consuming.

Under these circumstances, in this doctoral dissertation, focusing on features at defect points (Open NG and Short NG) on waveform data we formulate the problem as a highly data driven non-linear classification problem. Then, neural network based solutions are proposed for defect inspection process of FPDs for the non-contact detection method instead of existing thresholding method. There is no any global threshold value is considered, instead, possible candidates points are picked up from waveform data and then three parameters, which are characteristics on and around candidate points, are considered as inputs to the neural network. Signal to noise (SNR) at the candidate point, residual difference in a neighborhood of the candidate point and the change of wave length in the same neighborhood in the differential waveform are the three input parameters considered.

Firstly a feed-forward neural network with two hidden layers is adopted and proved its successes. Then an extension, a topology optimized recurrent neural network, is proposed in order to overcome some drawbacks in the feed-forward neural network method. A genetic algorithm based evolutionary multi-objective optimization algorithm is proposed for evolving the topology of a suitable recurrent neural network and its training by back-propagation through time. This method was proven much better than both of the existing thresholding method and feed-forward network based method after applying with a large set of real input data.

Finally, by focusing on the existing line scanning method of the NG sensor over defective TFT lines, first the shortest possible scan path is approximated to an asymmetric traveling salesman problem with precedence constraints. Then an algorithm, a combination of modified self-organizing map, a 2-Opt algorithm and a repair algorithm, is proposed for optimizing a shortest possible scan path. The superiority of the method is proven by a number of simulation examples.

Acknowledgment

At the outset I would like to express my deepest gratitude and profound thanks to Professor Dr. Masatoshi Sakawa for giving me the opportunity to prepare a PhD thesis under his supervision, and for his key advices, valuable suggestions and encouragement at all stages of this work.

I would like to thank Professor Dr. Ichiro Nishizaki, Professor Dr. Katsuhiko Takahashi and Associate Professor Dr. Katsumi Morikawa for reading the manuscript and offering valuable comments.

My sincere thanks and gratitude also goes to Assistant Professor Dr. Takeshi Matsui, for his encouragement, fruitful guidance and all the support in the development of this thesis.

I must make a special note of thanks to Dr. Hiroshi Hamori, head of R&D division and the president, OHT Incorporation for encouraging me to pursue this research. This thesis would not have been possible without the help, support and patience of him, not to mention his advice and unsurpassed knowledge in flat panel industry. My cordial thanks also go to all the members of R&D division, OHT Incorporation

Finally, I am indebted to OHT Incorporation for the financial support and making all the resources available for me throughout this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prospects of Flat Panel Display Industry

As a vital human-machine visual interface system, flat panel displays (FPD) are indispensable to most electronic devices that require human operation and as a result LCD and plasma technologies are now in commonplace. Global FPD market has reached to a scale of multi-billion dollar and further growth is expected to continue worldwide. The growth will be primarily driven by its widespread use as display devices in various electronic gadgets such as TVs, notebook computers, personal computers, mobile phones, and public display systems. Technology innovations, falling prices, and robust demand from developing markets bode well for the future of the market. Thanks to the numerous R&D initiatives, FPDs, which carried high price tags until recently, are becoming cheaper. In addition, the FPD technology is influencing the product development and design stages of the end-users manufacturing processes.

Relentless R&D initiatives by both FPD manufacturers and researchers are likely to unveil a range of advanced FPD models in the near future. Furthermore, as LCD TVs become larger, it has become necessary for manufacturers to adopt larger mother glass substrates of FPDs and to increase production speed both to increase screen size and to reduce the production cost. The most modern factories for producing large LCD panels typically operate production lines that use glass substrate sizes of $2,200 \times 2,500$ mm (ninth-generation) or $2,880 \times 3,130$ mm (tenth generation). Figure 1.1 shows the remarkable growth of glass sizes of FPDs during the last decade.

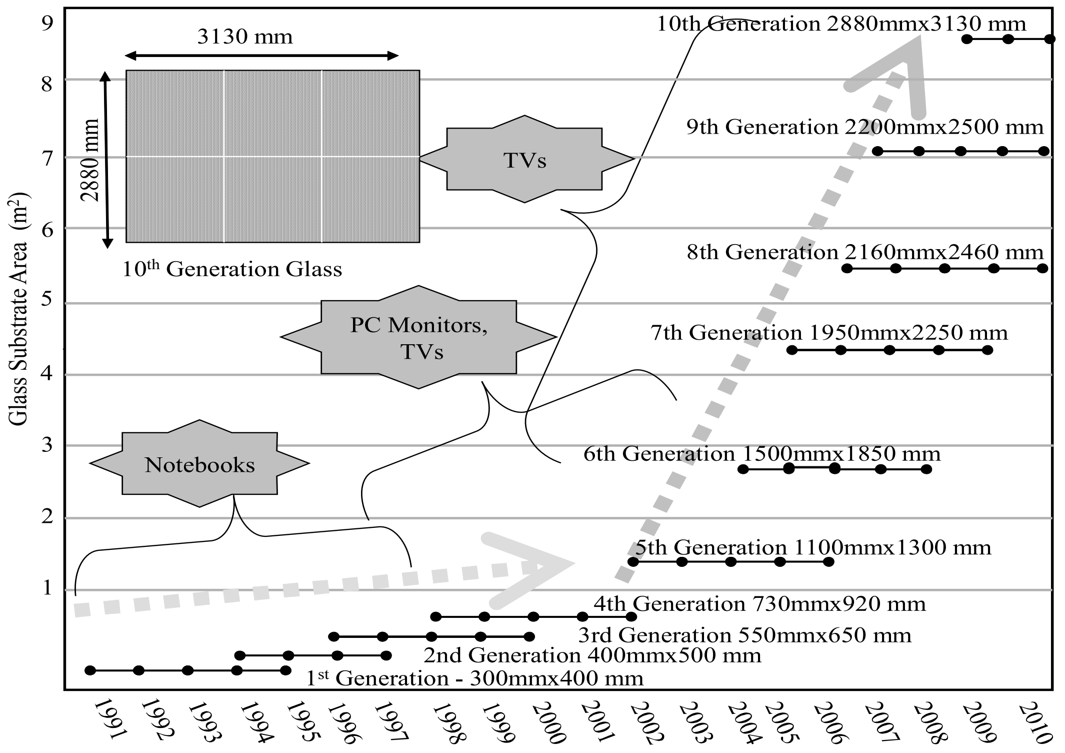


Figure 1.1: Growth of the size of mother glass substrates of flat panel displays.

1.2 Defect Inspection Process

As the demand for larger sizes of mother glasses, such as Generation 10,11 and 12, have been increasing, the demand for FPDs with high density pitch pattering of TFT arrays have also been increasing. Particularly, with the emergence of ultra-high definition 4K and 8K TVs and their expected growth in coming years, large scale FPDs with high pitch TFT arrays would be in the mainstream [58, 59, 80]. However with increasing pitch pattern density there is a tendency for having defects, such as inter-layer short circuits between TFT lines, to increase. Under these circumstances, detection and repair of defects in early manufacturing stages have become significantly important [64]. As a result, the speed and the precision of defects detection have been major issues for the manufactures of FPDs and researchers alike.

During the fabrication process, FPDs produce surface defects due to various reasons such as open circuits and short circuits within inside circuitry, dust, cracks, etc. Therefore, finished panels often show unevenness and non-uniformity (mura) due to those defects. Since flat panel display manufacturing is highly automated, most if not all flat panels are examined for defects during and after fabrication. This inspection stage is costly, and becomes more difficult when panel sizes increase. Reliability of inspection is also generally unknown. There are mainly two defect inspection processes, one is mura detection that takes place after panel fabrication, and the other is TFT array testing that takes place in early fabrication stage.

1.2.1 Region Mura Detection

Mura, the word “mura” derived from the Japanese word for blemish, are typically low-contrast imperfections that are larger than a single pixel, and are visible when the display is driven at a constant gray level. They may be caused by non-uniform distribution or impurities of liquid crystal or mechanical imperfections in the display assembly.

In the literature, there are several methods proposed for mura region detection. Lee and Yoo [44] proposed a region-mura defect detection method by using an image regression technique. Then, Kim et al. [41] proposed a high pass frequency filtering method which was also based on images. Li and Tsai [45] proposed a Hough transform-based method to identify low-contrast defects in unevenly-illuminated images, and especially focus on the inspection of mura defects in liquid crystal display (LCD) panels. There is another promising mura detection method, proposed by Tseng et al.[79], using multiple image back ground subtraction that can detect both white mura and black mura. Alam and Guoqing [6] also proposed an image based mura detection method by a fast Fourier transform technique.

However these mura detection methods are only for confirming FPDs to be defect free at latter stages of FPD fabrication, and if it is found to be defective those panels have to be disposed. Disposing a finished FPD, particularly of a larger generation size as described in Section 1.1, would be a huge loss and non-affordable for a modern manufacturer. Therefore, if there is a method of repairing and restoring those defects, it must be during early fabrication process.

1.2.2 TFT Array Testing

One of the main components of FPDs is the array of thin film transistor lines, or TFT array, which is basically a two-dimensional pixel cell array. Each pixel cell, in general, consists of a storage capacitor and a TFT switch. Each time, the gate drivers activate one row of the TFT array by setting the corresponding gate line voltage high; this turns on the pixel switches so that the source drivers can write data to the storage capacitors in the selected row. The testing of this array characterizes the electrical performance of the two-dimensional pixel array. It's most important goal is to identify and report the nature and location of the open circuits (open NGs) and short circuits (short NGs). Comprehensive array testing is crucial in yield management and quality control since it is the first opportunity to evaluate the electrical performance of a FPD and the last reliable opportunity to perform repair on defective pixels and gate/data lines. A glass panel with a defective TFT array that enters the assembly line is a waste of not only the assembly cost but also the other defect-free components.

When looking at literature, there are several methods proposed and already in use in the industry in which image based automatic optical inspection (AOI) methods were the first. Lu and Tsai [49, 50] proposed an image based defect detection method of singular value decomposition based image reconstruction technique. This method mostly answered for non-electrical defects such as pinholes, scratches, particles and fingerprints on the surface of TFT array. Then Tsai and Hung [78] proposed another

AOI based method using a Fourier reconstruction and wavelet decomposition technique and Yu and Jian [82] proposed a fuzzy classifier based method. Morphological characteristics on binary images were used by Noh et al. [63] and a technique called kernel principal component analysis was proposed by Liu et al.[47, 48] which was also able to classify the types of defects. Liu and Chen[46] have proposed another promising AOI method based on an statistical approach, a support vector machine. All of these methods are non-contact, hence no damage occurs to the panel surface, and almost real-time. However non of them can detect electrical defects, open NGs or short NGs, on TFT lines on the glass surface.

Another commonly used method is the pin probe method where electrode pins make direct contacts with each and every TFT line of the entire TFT array on panel surface and measure the current flown after applying a known voltage. Though this method has the advantage of detecting nothing but electrical defects, it also has disadvantages such as poor inspection speed, difficulty of adjusting for changes of TFT circuit design and line pitch and the necessity of frequent replacing of pin probing fixtures, which is an expensive process.

The non-contact FPD inspection method proposed by Hamori et al. [29, 30, 31, 32] is the most promising technique to-date, which is totally non-contact, utilizing a capacitor based sensor that scans over the TFT lines of mother glass panels of FPDs. The detection of defects is based on analyzing peaks and troughs on a waveform of a voltage signal captured by the sensor using a threshold method. However determining a proper threshold to correctly detect such peaks and troughs on wave form data is still not easy as the measured voltage signal is mixed with various noises such as random noises, external vibrations and noises due to environmental effects such as fluctuations of machine temperature. Another problem in this method is that, once defectives lines are observed by the above non-contact sensor, a camera based sensor, called NG sensor, scans all defective lines to determine exact defect points. This line scanning method is a mere top-down left to right unidirectional path, which is heavily time consuming.

Henley et al.[33] proposed another non-contact FPD inspection method, by using a voltage image of the TFT array, which looked promising in its early stage in 1990s. However it didn't get any recognition from manufacturers mainly due to it's inability to adapt to different TFT arrays as well as some practical problems in implementations.

Under these circumstances, in this doctoral dissertation, focusing on optimization of defect inspection process of flat panel displays, we address two main sub-processes in the main process, namely, a highly data driven defects detection sub process and the subsequent path optimization sub-process.

In the defects detection sub-process, focusing on features at defect points (Open NG and Short NG) on waveform data we formulate the problem as a highly data driven non-linear classification problem. Then, neural network based solutions are proposed for defect inspection process of FPDs for the non-contact detection method instead of existing thresholding method. There is no any global threshold value is considered, instead, possible candidates points are picked up from waveform data and three parameters, which are characteristics on and around candidate points, are considered as inputs to the neural network. Signal to noise (SNR) at the candidate point, residual difference in a neighborhood of the candidate point and the change of wave length in the same neighborhood in the differential waveform are the three input parameters considered.

Firstly a feed-forward neural network, with two hidden layers, is adopted and proved it's success over the existing thresholding method. Then an extension to it, a topology optimized recurrent neural network, is proposed. The optimization algorithm is a genetic algorithm based multi-objective evolutionary process that evolves the topology of the recurrent neural network and trained it through back-propagation algorithm through time. This method was proven much better than the existing thresholding method after applying and verified with a large set of real input data.

Finally in the NG sensor's path optimization sub-process, by focusing on the existing heavily time consuming top-down unidirectional line scanning method, first the shortest scan path is approximated to a traveling salesman problem with precedence constraints. Then a combination of self-organizing map with some modifications and 2-Opt algorithm is proposed for optimization of a shortest possible scan path. The superiority of the method is proven by a number of simulation examples.

The remaining part of this chapter gives the outline of the dissertation and published papers of the author.

1.3 Outline of the Thesis

Defects inspection process of flat panel display fabrication through neural networks is considered in this doctoral dissertation. Initially a feed-forward neural network with two hidden layers is proposed for detection of NG points on waveform data, where input waveform data are from the existing non-contact defect inspection sensor proposed by Hamori et al. [29, 30, 31, 32]. Then an extension with a recurrent neural network, in which the the topology and training of the recurrent network is optimized by a genetic algorithm based multi-objective evolutionary optimization technique, is proposed to address the drawbacks of feed-forward network. An optimization method, combining a modified self-organizing map, a 2-Opt algorithm and a repair algorithm, is also proposed for optimization of the shortest path of the NG sensor, which scans individual defective lines for locating exact positions of NG points on lines. The organization of each chapter is briefly summarized in the following.

In chapter 2, the basic concepts and methods of artificial neural network used in the dissertation are given briefly. Firstly, concepts of artificial neuron, historical background and engineering approach of an artificial neural network are discussed. Then different architectures of artificial neural networks including fee-forward neural networks, recurrent neural networks and Kohonen's network are presented. The perceptron learning and the learning process of a neural network is discussed next. Finally, the back-propagation algorithm and learning with back-propagation together with back-propagation through time are explained.

In chapter 3, the non-contact defect inspection of flat panel displays proposed by Hamori et al. [29, 30, 31, 32] is briefly described. Scanning of TFT lines on flat panel displays and waveform data capture system by a non-contact sensor and the features of waveform data are also briefly described. Then the thresholding method currently used by Hamori et al. is explained followed by its drawbacks, where those drawbacks were inspired for using neural network approach in this dissertation.

In chapter 4, focusing on feed-forward neural networks in defect detection on flat panel displays, some related literature and the applicability of neural networks to the problem is discussed. Then an analysis of input waveform data and how to select input parameters is included. The topology of the selected feed-forward neural network and its training by back-propagation algorithm is presented next, and finally, some training results and detection results using the trained network are presented

In chapter 5, focusing on recurrent neural networks in defect detection on flat panel displays, some related literature and the applicability of recurrent neural networks to the problem is discussed. The topology of the recurrent neural network and its training is evolved by a genetic algorithm based multi-objective evolutionary optimization technique and therefore some literature about optimization is also presented. Then the multi-objective evolutionary optimization algorithm is presented step by step and finally some detection results using the optimized recurrent network together with a comparison with existing method is presented.

In chapter 6, focusing on Kohonen's self-organizing maps and its applications for traveling salesman problem solvers, some related literature in both self-organizing maps and traveling salesman problem are discussed. Then how the problem, of finding the shortest path of the NG scan sensor over defective TFT lines scattered throughout a panel structure, is approximated to an asymmetric traveling salesman problem with precedence constraints is presented. Then its solution using a combination of modified self-organizing map, a 2-Opt algorithm and a repair algorithm is presented. Finally some simulated results together with a comparison between existing method are also presented for the verification of the method's superiority over existing top-down unidirectional method.

Chapter 7 concludes the doctoral dissertation and briefly summarizes this research together with some remarks of our future directions.

1.4 List of Publications

[A-1] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, Defects Detection on TFT lines of Flat Panel Displays using a Feed Forward Neural Network, *Artificial Intelligence Research*, Vol. 2, No. 4, pp. 1-12, 2013.

[A-2] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, Defects Detection on TFT lines of Flat Panel Displays using an Evolutionary Optimized Recurrent Neural Network, *American Journal of Operations Research*, Vol. 4, No. 3, pp. 113-123, May 2014.

[A-3] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, Path Optimization for Line Scanning on Flat Panel Displays using a Self-Organizing Map, *Computational Research*, Vol. 2, No. 4, pp. 63-68, 2014.

[B-1] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, A Neural Network Approach for Non-Contact Defects Inspection of Flat Panel Displays, *17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems (KES 2013)*, Kita Kyushu, Japan, pp. 28-38, September 2013.

[B-2] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, A Multi-Objective Evolutionary Optimized Recurrent Neural Network for Defects Detection on Flat Panel Displays, *11th International Conference on Modeling Decisions for Artificial Intelligence (MDAI 2014)*, Tokyo, Japan, pp. 170-181, October 2014.

Chapter 2

Artificial Neural Networks

2.1 Introduction

Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. They are being successfully applied across an extraordinary range of problem domains, in areas as diverse as engineering, medicine, physics, biology and finance. The excitement stems from the fact that these networks are attempts to model the capabilities of the human brain. From a statistical perspective neural networks are interesting because of their potential use in prediction and classification problems.

ANNs are non-linear, data driven and self adaptive approaches as opposed to the traditional model based methods. They are powerful tools for modeling, especially when the underlying data relationship is unknown. They can identify and learn correlated patterns between input data sets and corresponding target values. Once trained, ANNs can be used to predict the outcome of new independent input data. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy.

Neural networks have been used for a wide variety of applications where statistical methods are traditionally employed. They have been used in classification problems, such as recognizing speech, image pattern recognition, data classification, and predicting the secondary structure of input patterns. In time-series applications, ANNs have been used in predicting stock market performance. These problems are normally solved through classical statistical methods, such as discriminant analysis, logistic regression, Bayesian analysis, multiple regression, etc. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons and is true of ANNs as well.

2.2 Historical Background

Research in Artificial Neural Networks (ANN) has experienced three period of extensive activities in the history. The first peak of activities happened in 1940s due to McCulloch and Pitt's pioneering work [57]. The first artificial neuron was produced by them but the technology available at that time did not allow them to do much. The second rise occurred in the 1960s with the introduction of perceptron theorem by Rosenblatt [68]. However with results shown by the perceptron convergence theorem by Minsky and Papert [60] , the enthusiasm of most of the researches dampened and the lull continued another two decades. ANNs received renewed interest again in early 1980s initially due to Hopfield's energy approach [34] in 1982 and the error back-propagation learning algorithm for multilayer feed-forward neural networks first proposed by Werbos [81] in 1974 and then reinvented several times by Rumelhart and McClelland [69, 70] in 1987. That gained recognition and led to a renaissance in the field of artificial neural network research among a large community of researchers from many scientific disciplines.

2.3 Biological Motivation

The long course of evolution has given the human brain many desirable characteristics not present in Von Neumann or modern parallel computers. These include massive parallelism, distributed representation and computation, learning ability, generalization ability, adaptivity, inherent contextual information processing, fault tolerance, and low energy consumption. Modern digital computers outperform humans in the domain of numeric computation and related symbol manipulation. However, humans can effortlessly solve complex perceptual problems (like recognizing a man in a crowd from a mere glimpse of his face) at such a high speed and extent as towards the world's fastest computer. Why is there such a remarkable difference in their performance? The biological neural system architecture is completely different from the von Neumann architecture. This difference significantly affects the type of functions each computational model can best perform.

Numerous efforts to develop intelligent programs based on Von Neumanns centralized architecture have not resulted in general-purpose intelligent programs. Inspired by biological neural networks, ANNs are massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections. ANN models attempt to use some organizational principles believed to be used in the human brain. Modeling a biological nervous system using ANNs can also increase our understanding of biological functions. State-of-the-art computer hardware technology has made this modeling feasible.

Much is still unknown about how the brain trains itself to process such information. In the human brain, a typical neuron (or nerve cell) collects signals from others through a host of fine structures called dendrites (Figure 2.1). The neuron sends out spikes of electrical activity through a long and thin stand known as axon, which splits into thousands of branches. At the end of each branch, structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neuron. When a neuron receives excitatory input that sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

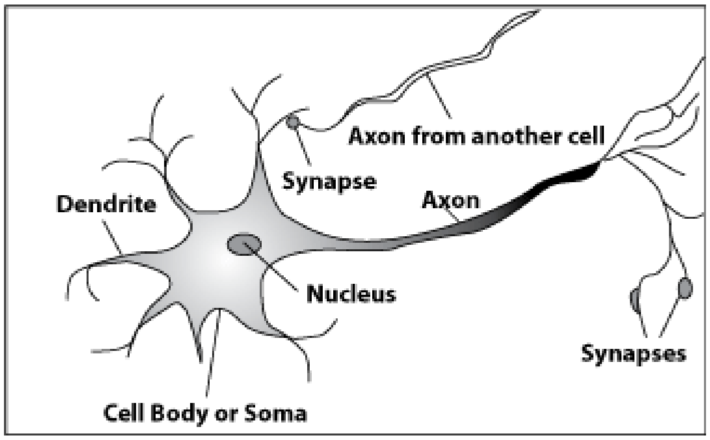


Figure 2.1: A biological neuron and its components.

2.4 Characteristics of Neural Networks

Some attractive characteristics of neural networks that make it superior to even the most sophisticated computer system for pattern recognition are the following:

- Robustness and fault tolerance: The decay of nerve cells does not seem to affect the performance significantly.
- Flexibility: The network automatically adjusts to a new environment without using any preprogrammed instructions.
- Ability to deal with a variety of data situations: The network can deal with information that is fuzzy, probabilistic, noisy, and inconsistent.
- Collective computations: The network performs routinely many operations in parallel and also a given task in a distributed manner.
- Mapping capabilities: ANNs can map input patterns to their associated output patterns.
- Learn by example: Thus, the networks can be trained with known examples of a problem before they are tested with unknown instances of the problem,

where it can identify new objects previously untrained.

- Capability to generalize: Thus, they can predict new outcomes from past trends.

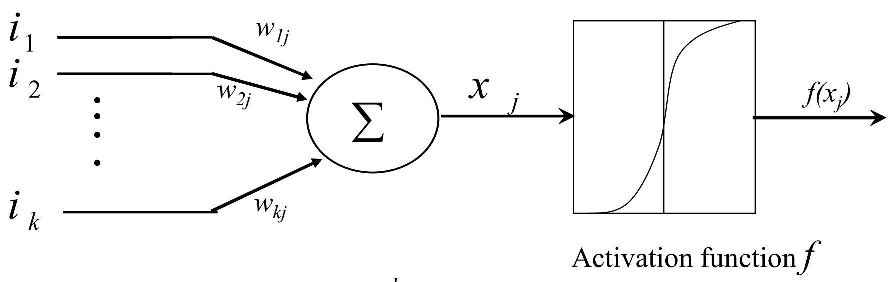
2.5 Basics of Artificial Neural Networks

As the terminology of ANNs has developed from biological model of the human brain, a neural network consists of a set of connected cell: the neuron. The neurons receive impulses from either input cells or other neurons and perform some kind of transformation of the input and transmit the outcome to other neurons or to output cells. The neural networks are built from layers of neurons connected so that one layer receives input from the proceeding layer of neurons and passes the output on to the next layer.

The activation function (or transformation function) in a neuron is a function of the input vector $\mathbf{x}(x_1, ..., x_n)$. The output is obtained as:

$$f(x_j) = f\left(\alpha_j + \sum_{i=1}^{k} w_{ij} y_j\right). \tag{2.1}$$

where f is the activation function, typically a sigmoid (logistic or tangent hyperbolic) function or a step function and α_j is a bias input to the neuron. A graphical presentation of an artificial neuron is given in Figure 2.2, where $i_1, i_2, .., i_k$ are inputs to the neuron and $w_{1j}, w_{2j}, ..., w_{kj}$ are synaptic weights associated with each input. Mathematically a multi-layer perceptron network is a function consisting of composition of weighted sums of the function corresponding to the neuron.



$$x_j = \sum_{m=1}^{k} w_{mj} i_m$$

Figure 2.2: An artificial neuron and its components.

2.6 Architectures of Neural Networks

An ANN is defined as a data processing system consisting of a large number of highly inter connected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain. Though there are several types of architectures of ANNs, the two most widely used architectures described below.

2.6.1 Feed-Forward Neural Networks

Feed-forward neural networks (FNN), also known as associative networks, are the most popular and most widely used models in many practical applications. They are known by many different names, such as "multi-layer perceptron." FNNs allow signals to travel one way only; from input to output. There is no loops (feedbacks), i.e. the output of any layer does not affect that same layer. FNNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down structure.

Figure 2.3 illustrates a single hidden layer FNN with 3 inputs, 2 outputs and and 4 hidden 4 neurons. Each arrow in the figure symbolizes a synaptic weight parameter in the network. The network is divided into layers. The input layer consists of just the inputs to the network. Then follows a hidden layer, which consists of any number of neurons, or hidden units placed in parallel. Each neuron performs a weighted summation of the inputs, which then passes a nonlinear activation function, also called the neuron function.

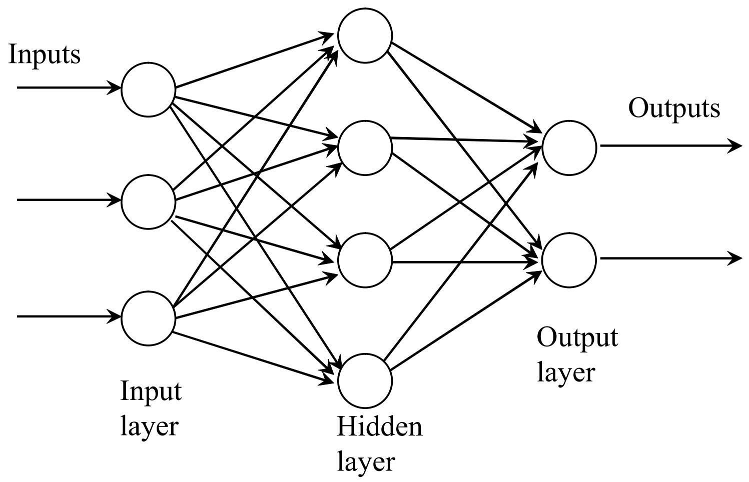


Figure 2.3: The structure of a feed-forward neural network.

2.6.2 Feedback Neural Networks

Feedback neural networks, also knows as auto-associative networks or recurrent neural networks, can have signals traveling in both directions by introducing loops in the network. They are very powerful and can get extremely complicated during developing the network and training. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Figure 2.4 illustrates a structure of a feedback neural network, where there are two feedback connections or loops, one from output layer neuron to first neuron in the hidden layer and the other from the third neuron of the hidden layer onto itself.

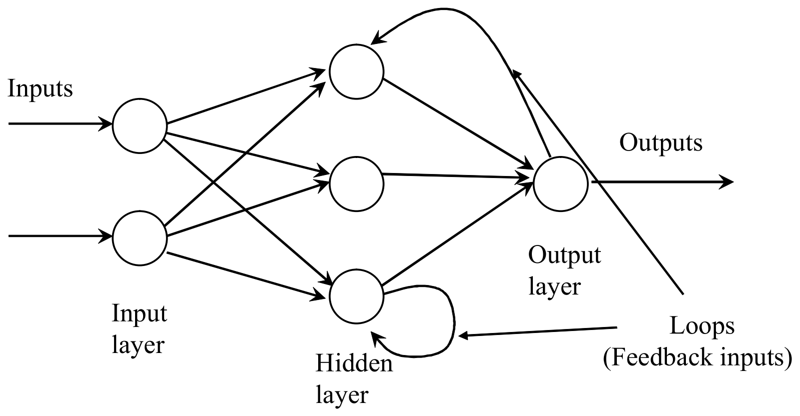


Figure 2.4: The structure of a feedback neural network.

2.7 Types of Neural Networks

This classification, the types of neural networks, is based on the behavior of neurons and directions of data they pass between input, processing and output neurons. In other words, the way neurons semantically communicate within the networks, is different in each type of neural network.

2.7.1 Multi-Layer Perceptron

The most popular form of neural network architecture is the multi-layer perceptron (MLP), and a multi-layer perceptron:

- has any number of inputs.
- has one or more hidden layers with any number of units.
- uses linear combination functions in the input layers.
- uses generally sigmoid activation function in hidden layers.
- has nay number of outputs with any activation function.
- has connections between the input layer and the first hidden layer, between

the hidden layers, and between the last hidden layer and the output layer.

2.7.2 Radial Basis Function Networks

Radial basis function (RBF) networks are also feed-forward, but have only one hidden layer. An RBF network:

- has any number of inputs.
- typically has only one hidden layer with any number of units.
- uses radial combination functions in the hidden layer, based on the squared Euclidean distance between the input vector and the weight vector.
- typically uses exponential activation function in the hidden layer. in which case the network is Gaussian RBF network.
- has any number of outputs with any activation function.
- has connections between the input layer and the hidden layer, and between the hidden layer and the output layer.

2.7.3 Kohonen Networks

Kohonen networks (or Kohonen self-organizing feature maps, or SOMs for short) are another type of artificial neural networks developed in 1982 by Tuevo Kohonen[43]. As its name suggests, SOMs learn on their own through unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representations of the input space, called a map. SOMs are different from other ANNs in the sense that they use a neighborhood function to preserve the topological properties of the input space.

The principal goal of a SOM is to transform an incoming signal pattern of arbitrary dimension into one or two dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion. Therefore a SOM is a set up placing neurons at the nodes of a one or two dimensional lattice. Higher dimensional maps are also possible, but not so common. The neurons become selectively tuned to various input patterns (stimuli) or classes of input patterns during the course of the competitive learning. The locations of the neurons so tuned (i.e. the winning neurons) become ordered and a meaningful coordinate system for the input features is created on the lattice. The SOM thus forms the required topographic map of the input patterns.

Therefore SOMs are useful for multi-dimensional scaling or visualizing high-dimensional data in low-dimensional views. An SOM consists of components called nodes or neurons. Each node is associated with a weight vector and a position in the map space.The usual arrangement of nodes is a two-dimensional regular spacing in a hexagonal or rectangular grid. Figure 2.5 shows a mapping of a high dimensional input space into a low dimensional space by a self-organizing map.

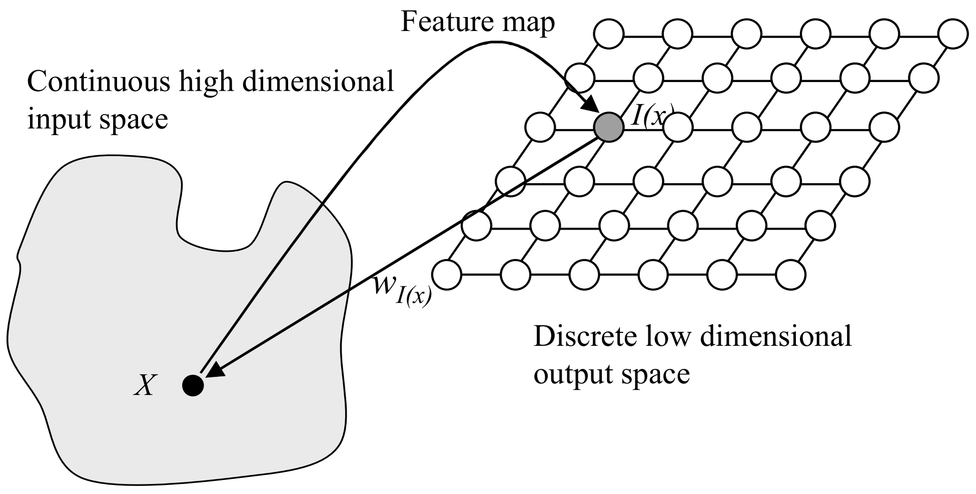

Feature map
Continuous high dimensional input space
I(x)
w_{I(x)}
X
Discrete low dimensional output space

Figure 2.5: A self-organizing map.

2.8 Learning Methods of Neural Networks

Learning is essential to most of the neural network architectures and hence the choice of a learning algorithm is a central issue in neural network development. What is really meant by saying that a processing element learns is that a processing unit is capable of changing its input/output behavior as a result of changes in the environment. Since the activation rule is generally fixed when the network is constructed and since the input vector cannot be changed, to change the input/output behavior the weights corresponding to inputs need to be adjusted. A method is thus needed by which, at least during the training stage, weights can be modified in response to the input/output process. A number of such learning rules are available for neural network models.

2.8.1 Supervised Learning

Supervised learning is fairly common in classification problems because the goal is often to get the computer to learn a classification system that we have created. More generally, classification learning is appropriate for any problem where deducing a classification is useful and the classification is easy to determine. In some cases, it might not even be necessary to give pre-determined classification to every instance of a problem if the agent can work out the classification for itself.

Supervised learning entails learning a mapping between a set of input variables and an output variable and applying that mapping to predict the output for unseen data. Supervised learning is the most important methodology in machine learning and it also has central importance in the processing of multimedia data. Every input variable that is used to train the network is associated with an output pattern, which is the target or the desired pattern. A teacher is assumed to be present during learning process, when a comparison is made between the network's computed output and the desired or expected output to determine the error. The error is used to change network parameters, which result in improving the performance.

2.8.2 Unsupervised Learning

Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. By contrast with supervised learning or reinforcement learning, there are no explicit target output or environmental evaluations associated with each input; rather the unsupervised learner brings to bear prior biases as to what aspects of the structure of the input should be captured in the output.

Unsupervised learning is important since it is likely to be much more common in the brain than supervised learning. For example there are around 10^6 photoreceptors in each eye whose activities are constantly changing with the visual world and which provide all the information that is available to indicate what objects are in the world, how they are presented, what the lighting conditions are, etc. Developmental and adult plasticity are critical in animal vision ; indeed structural and physiological properties of synapses in the neocortex are known to be substantially influenced by the patterns of activity in sensory neurons that occur. This makes unsupervised methods essential, and equally, allows them to be used as computational models for synaptic adaptation.

2.8.3 Reinforced Learning

Reinforced learning is learning what to, how to map situations to actions, so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediately reward but also the next situation and, through that, all subsequent rewards. These two characteristics , trial and error search and delayed reward, are the two most important distinguishing features of reinforcement learning.

In this method, a teacher though available, does not present the expected answer but only indicates its network output is correct or incorrect. The information provided helps the network in its learning process. A reward is given for a correct answer and penalty for wrong answer. But, reinforced learning is not a popular form of training, where a comparison is made between the network's computed output and the desired or expected output patterns. Hence, the system learns of its own.

One of the challenges that arises in reinforced learning and not in other kinds of learning is the trade-off between exploration and exploitation. To obtain a lot of reward, a reinforced learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. Another key feature of reinforced learning is that it explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment.

2.9 Training with Back-Propagation

The Back-propagation algorithm was originally introduced in the 1970s, but its importance was not fully appreciated until that famous publication by Rumelhart et al.[69] in 1986. That paper together with the book by Rumelhart and McClelland [70] describes several neural networks where back-propagation works far faster than earlier approaches to learning, making it possible to use neural networks to solve problems which had previously been insoluble. Today the back-propagation is the workhorse of learning in many neural networks.

Back-propagation requires a known, desired output for each input value in order to calculate the gradient of the loss function. It is therefore usually considered to be a supervised learning method. It is a generalization of the delta rule to multi-layered feed-forward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Back-propagation requires that the activation function used by the artificial neurons to be differentiable.

2.9.1 The Learning Problem

A feed-forward neural network is a computational graph whose nodes are computing units and whose directed edges transmit numerical information from node to node. Each computing unit is capable of evaluating a single primitive function of its input. In fact the network represents a chain of function compositions which transform an input to an output vector (called a pattern). The network is a particular implementation of a composite function from input to output space, which is called the network function. The learning problem consists of finding the optimal combination of weights so that the network function φ approximates a given function f as closely as possible. However, the function f is not given explicitly but only implicitly through some examples.

Consider a fee-forward neural network with n input and m output units. It can consist of any number of hidden units and can exhibit any desired feed-forward connection pattern. A training set $\{(x_1, t_1), (x_2, t_2), ..., (x_p, t_p)\}$ is also given consisting of p ordered pairs of $n-$ and $m-$dimensional vectors, which are called the input and output patterns. The primitive functions at each node of the network is considered to be continuous and differentiable. The weights of the edges are real numbers selected at random. When the input pattern x_i from the training set is presented to this network, it produces an output o_i different in general from the target t_i. What is needed is to make o_i and t_i identical for $i = 1, ..., p$ by using a learning algorithm. More precisely, it is needed to minimize the error function of the network, defined as;

$$E = \frac{1}{2} \sum_{i=1}^{p} \parallel o_i - t_i \parallel^2 \tag{2.2}$$

After minimizing this function for the training set, new unknown input patterns are presented to the network and expected to be interpolated. The network must recognize whether a new input vector is similar to learned patterns and produce a similar input.

The back-propagation algorithm is used to find a local minimum of the error function. The network is initialized with randomly chosen weights. The gradient of the error function is computed and used to correct the initial weights. The first step of the minimization process consists of extending the network, so that it computes the error function automatically as shown in Figure 2.6.

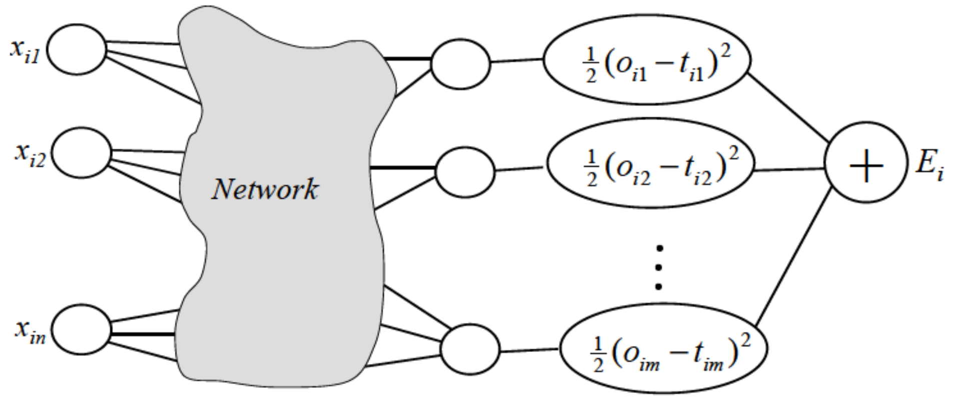

x_{i1}
x_{i2}
x_{in}
Network
$\frac{1}{2}(o_{i1}-t_{i1})^2$
$\frac{1}{2}(o_{i2}-t_{i2})^2$
$\frac{1}{2}(o_{im}-t_{im})^2$
+
E_i

Figure 2.6: Extended network for the computation of error function.

2.9.2 Back-Propagation Algorithm

In order to train a neural network to perform some task, the weights of each unit must be adjusted in such a way that the error between the desired output and the actual output is reduced. This process requires that the network compute the error derivative of weights(**EW**), in other words, it must be calculated how the error changes as each weight is increased or decreased slightly. The back-propagation algorithm is the most widely used method for determining the **EW**.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each **EW** by first computing the **EA**, the rate at which the error changes as the activity level of a unit is changed. For output units, the **EA** is simply the difference between the actual and desired output. To compute the **EA** for a hidden unit in the layer just before the output layer, first identify all the weights between that hidden unit and the output units to which it is connected. Then those weights are multiplied by the **EA**s of those output units and products are added up. This sum is equal to **EA** for the chosen hidden unit. Once all the **EA**s in the hidden layer just before the output layer are calculated, the **EA**s for other layers can be computed in like fashion, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back-propagation its name. Once the **EA** has been computed for a unit, it is straight forward to compute the **EW**s for each incoming connection of the unit. The **EW** is the product of **EA** and the activity through the incoming connection

2.9.2.1 Learning with Back-propagation

Since the error function E needs to be minimized, which depends on the network weights, it is needed to deal with all weights in the network one at a time. The feed-forward step is computed in the usual way, but the output of each unit is stored in its right side. The back-propagation step is performed in the extended network that computes the error function and then fix the attention on one of the weights, say w_{ij} whose associated edge points from the i^{th} to the j^{th} node in the network. This weight can be treated as an input channel into the subnetwork made of all paths starting at w_{ij} and ending in the single output unit of the network. The information fed into the subnetwork in the feed-forward step was $o_i w_{ij}$, where o_i is the stored output of the unit i. The back-propagation step computes the gradient of E with respect to this input, i.e., $\partial E / \partial o_i w_{ij}$. Since in the back-propagation step o_i is treated as a constant, we have

$$\frac{\partial E}{\partial w_{ij}} = o_i \frac{\partial E}{\partial o_i w_{ij}}. \tag{2.3}$$

In other words, the back-propagation is performed in the usual way. All subnetworks defined by each weight of the network can be handled simultaneously, but it stores additionally at each node i:

- The output o_i of the node in the feed-forward step.

- The cumulative result of the backward computation in the back-propagation step up to this node, and this quantity is called the back-propagated error.

If the back propagated error at the j^{th} node is denoted by δ_j, then the partial derivative of E with respect w_{ij} can be expressed as:

$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j. \tag{2.4}$$

Once all the partial derivatives have been computed, the gradient descent can be performed by adding to each weight w_{ij} the increment

$$\Delta w_{ij} = -\gamma o_i \delta_j. \tag{2.5}$$

This correction is needed to transform the back-propagation algorithm into a learning method for neural networks.

2.9.3 The Case of Layered Networks

The most important special case of feed-forward networks is that of layered networks with one or more hidden layers. The following describes explicit formulas for weight updates and show how they can be calculated using linear algebraic operations.

2.9.3.1 Extended Network

Consider a network with n input sites, k hidden, and m output units.The weight between input site i and hidden unit j will be called $w_{ij}^{(1)}$. The weight between hidden unit i and output unit j will be called $w_{ij}^{(2)}$. The bias $-\theta$ of each unit is implemented as the weight of and additional edge. Input vectors are thus extended with a 1 component, and the same is done with the output vector from the hidden layer as shown in Figure 2.7. The weight between the constant 1 and the hidden unit j is called $w_{n+1,j}^{(1)}$ and the weight between the constant 1 and the output unit j is denoted by $w_{k+1,j}^{(2)}$. There are $(n+1) \times k$ weights between input sites and hidden

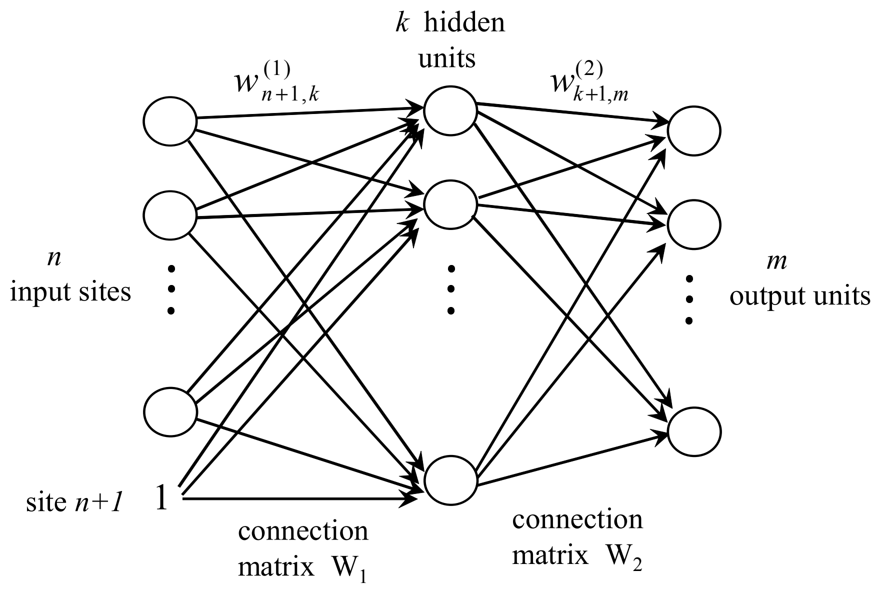


Figure 2.7: Notation for the 3-layered network.

units and $(k+1) \times m$ weights between hidden and output units. Let $\overline{\boldsymbol{W}}_{\mathbf{1}}$ be the $(n+1) \times k$ matrix with components $w^{(1)}_{ij}$ at the i^{th} row and j^{th} column. Similarly let $\overline{\boldsymbol{W}}_{\mathbf{2}}$ be the $(k+1) \times m$ matrix with components $w^{(2)}_{ij}$. An over lined notation is used to emphasize that the last row of both matrices correspond to the biases of computing units. The matrix of weights without this last row will be needed in the back-propagation step. The $n-$dimensional input vector $\boldsymbol{o} = (o_1, ..., o_n)$ is extended, transforming it to $\hat{\boldsymbol{o}} = (o_1, ..., o_n, 1)$. The excitation net_j of the j^{th} hidden unit is given by;

$$net_j = \sum_{i=1}^{n+1} w^{(1)}_{ij} \hat{o}_i. \tag{2.6}$$

The activation function is a sigmoid and the output $o^{(1)}_j$ of this unit is thus;

$$o^{(1)}_j = s\left(\sum_{i=1}^{n+1} w^{(1)}_{ij} \hat{o}_i\right). \tag{2.7}$$

The excitation of all units in the hidden layer can be computed with the vector-matrix multiplication $\hat{\boldsymbol{o}}\overline{\boldsymbol{W}}_{\mathbf{1}}$. The vector $\boldsymbol{o}^{(1)}$ whose components are the outputs of the hidden units is given by;

$$\boldsymbol{o}^{(1)} = s(\hat{\boldsymbol{o}}\overline{\boldsymbol{W}}_{\mathbf{1}}), \tag{2.8}$$

using the convention of applying the sigmoid to each component of the argument vector. The excitation of the units in the output layer is computed using the extended vector $\hat{\boldsymbol{o}}^{(1)} = (o_1^{(1)}, ..., o_k^{(1)}, 1)$. The output of the network is the $m-$dimensional vector $\boldsymbol{o}^{(2)}$, where;

$$\boldsymbol{o}^{(2)} = s(\hat{\boldsymbol{o}}^{(1)}\overline{\boldsymbol{W}}_2). \tag{2.9}$$

These formulas can be generalized for any number of layers and allow direct computation of the flow of data in the network with simple matrix operations.

2.9.3.2 Steps of the Algorithm

Figure 2.8 shows the extended network for computation of the error function. In order to simplify, a single input-output pair (o, t) is considered. And the network has been extended with an additional layer of units. The right sides compute the quadratic deviation $\frac{1}{2}(o_i^2 - t_i)$ for the i^{th} component of the output vector and the left sides store $(o_i^2 - t_i)$. Each output unit i in the original network computes the sigmoid s and produces the output o_i^2. Addition of the quadratic deviations gives the error E. The error function for p input-output examples can be computed by creating p networks like the one shown, one for each training pair, and adding the outputs of all of them to produce the total error of the training set.

After choosing the weights of the network randomly, the back-propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

1. Feed-forward computation
2. Back-propagation to the output layer
3. Back-propagation to the hidden layer

4. Weight updates

The algorithm is stopped when the value of the error function has become sufficiently small.

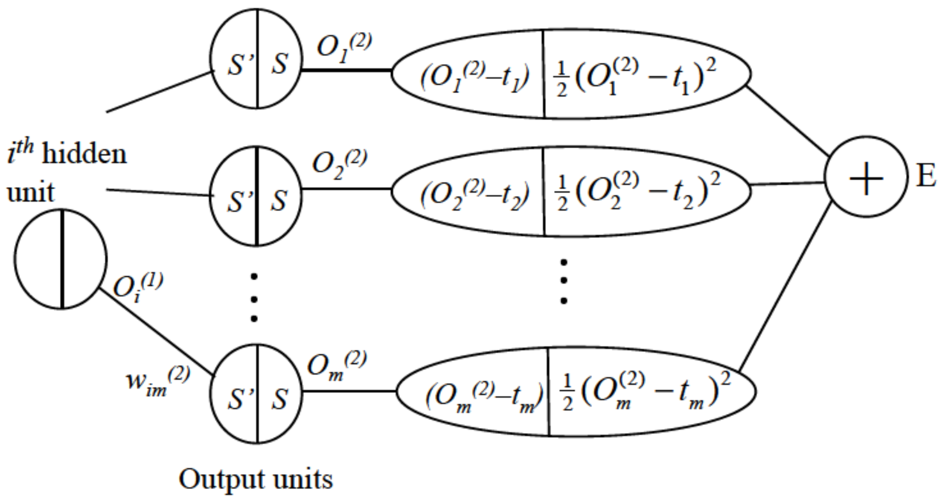

S' | S
$O_1^{(2)}$
$(O_1^{(2)}-t_1)$ | $\frac{1}{2}(O_1^{(2)}-t_1)^2$
i^{th} hidden unit
$O_2^{(2)}$
$(O_2^{(2)}-t_2)$ | $\frac{1}{2}(O_2^{(2)}-t_2)^2$
$+$
E
$O_i^{(1)}$
$w_{im}^{(2)}$
$O_m^{(2)}$
$(O_m^{(2)}-t_m)$ | $\frac{1}{2}(O_m^{(2)}-t_m)^2$
Output units

Figure 2.8: Extended multilayer network for computation of error.

First Step: Feed-forward computation

The vector **o** is presented to the network. The vectors $\mathbf{o^{(1)}}$ and $\mathbf{o^{(2)}}$ are computed and stored. The evaluated derivatives of the activation functions are also stored at each unit.

Second Step: Back-propagation to the output layer

The back-propagation path from the output of the network up to the output unit j is shown in Figure 2.9.
From this path all the multiplicative terms can be collected by simple inspection which define the back-propagated error $\delta_j^{(2)}$. Therefore

$$\delta_j^{(2)} = o_j^{(2)}(1 - o_j^{(2)})(o_j^{(2)} - t_j), \tag{2.10}$$

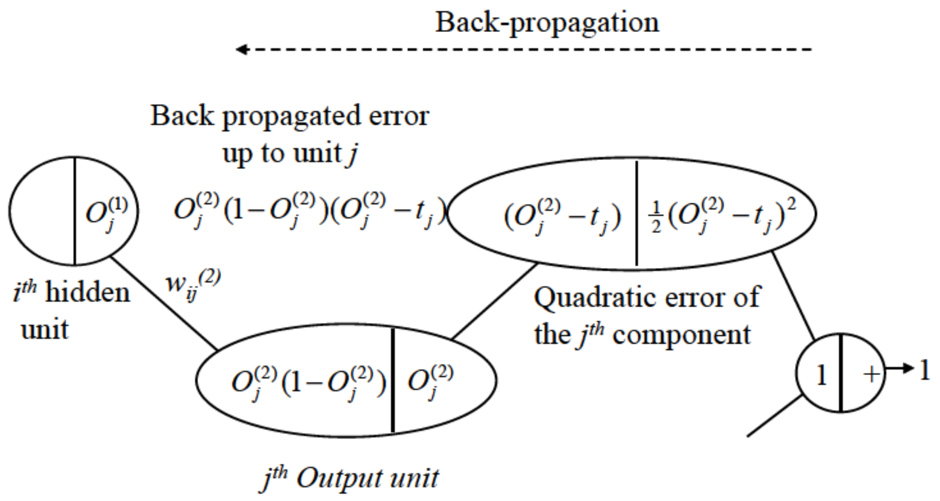

Back-propagation
Back propagated error up to unit j
$O_j^{(2)}(1-O_j^{(2)})(O_j^{(2)}-t_j)$
$O_j^{(1)}$
i^{th} hidden unit
$w_{ij}^{(2)}$
$(O_j^{(2)}-t_j)$
$\frac{1}{2}(O_j^{(2)}-t_j)^2$
Quadratic error of the j^{th} component
$O_j^{(2)}(1-O_j^{(2)})$
$O_j^{(2)}$
1
+
1
j^{th} Output unit

Figure 2.9: Back-propagation path upto output unit j.

and the partial derivative is

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \left(o_j^{(2)}(1-o_j^{(2)})(o_j^{(2)}-t_j)\right) o_j^{(1)} = \delta_j^{(2)} o_i^{(1)}. \tag{2.11}$$

where the weight $w_{ij}^{(2)}$ is a variable and its output $o_i^{(1)}$ is a constant.

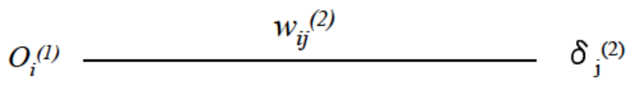


Figure 2.10: Input and back-propagation error at an edge.

Figure 2.10 shows the general situation during the back-propagation algorithm. At the input side of the edge with weight w_{ij} there is $o_i^{(1)}$ and at the output side there is back-propagated error $\delta_j^{(2)}$

Third Step: Back-propagation to the hidden layer

Now it is needed to compute the partial derivatives $\frac{\partial E}{\partial w_{ij}^{(1)}}$. Each unit j in the hidden layer is connected to each unit q in the output layer with an edge of weight $w_{jq}^{(2)}$, for

$q = 1, ..., m$. The back-propagated error up to unit j in the hidden layer must be computed taking into account all possible backward paths, as shown in Figure 2.11. The back-propagated error is then

$$\delta_j^{(1)} = o_j^{(1)}(1 - o_j^{(1)}) \sum_{q=1}^{m} w_{jq}^{(2)} \delta_q^{(2)}. \tag{2.12}$$

Therefore the partial derivative is

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_j^{(1)} o_i. \tag{2.13}$$

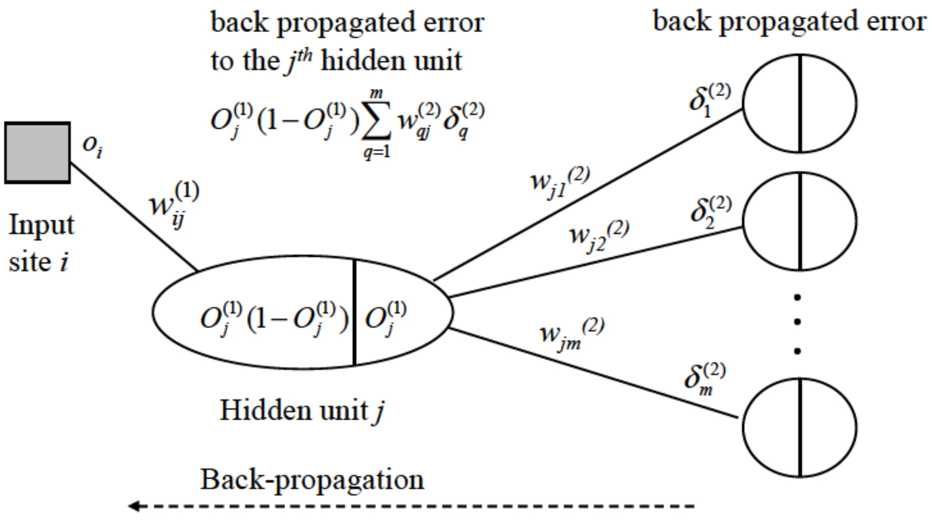


Figure 2.11: All paths up to input site i.

The back-propagated error can be computed in the same way for any number of hidden layers and the expression for the partial derivatives of E keeps the same analytic form.

Fourth Step: Weight updates

After computing all partial derivatives the network weights are updated in the negative gradient direction. A learning constant γ defines the step length of the correction. The corrections for the weights are given by

$$\Delta w_{ij}^{(2)} = -\gamma o_i^{(1)} \delta_j^{(2)}, \quad for \;\; i = 1, ..., k+1; j = 1, ..., m, \tag{2.14}$$

and

$$\Delta w_{ij}^{(1)} = -\gamma o_i \delta_j^{(1)}, \quad for \;\; i = 1, ..., n+1; j = 1, ..., k, \tag{2.15}$$

where $o_{n+1} = o_{k+1}^{(1)} = 1$. It is very important to make the corrections to the weights only after the back-propagated error has been computed for all units in the network. Otherwise the corrections become intertwined with the back-propagation of the error and the computed corrections do not correspond any more to the negative gradient direction though some authors fall in this trap. Note also that some books define the back-propagated error as the negative traversing value in the network. In that case the update equations for the network weights do not have a negative sign (which is absorbed by the deltas), but this is a matter of pure convention.

More than one training pattern

In the case of more than 1 input-output patterns ($p > 1$ in equation 2.2), an extended network is used to compute the error function for each of them separately. The weight corrections are computed for each pattern, for example, for weight $w_{ij}^{(1)}$ the corrections are

$$\Delta_1 w_{ij}^{(1)}, \Delta_2 w_{ij}^{(1)}, ..., \Delta_p w_{ij}^{(1)}. \tag{2.16}$$

The necessary update in the gradient direction is then

$$\Delta w_{ij}^{(1)} = \Delta_1 w_{ij}^{(1)} + \Delta_2 w_{ij}^{(1)} + ... + \Delta_p w_{ij}^{(1)}. \tag{2.17}$$

Error Convergence

Figure 2.12 shows the typical pattern of convergence of the total error during training of a feed-forward neural network. After some iterations the algorithm finds a solution to the learning problem. In the figure the error falls fast at the beginning and end of training. Between these two zones lies a region in which the error function seems to be almost flat and where progress is slow. This corresponds to a region which would be totally flat if step functions were used as activation functions of the units. However, using the sigmoid, this region presents a small slope in the direction of the global minimum.

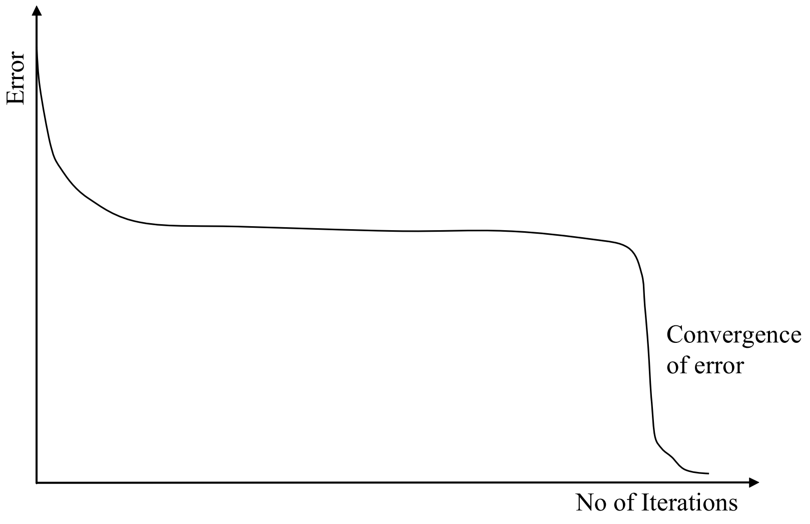


Figure 2.12: Convergence pattern of error function during back-propagation.

2.9.4 Back-Propagation Through Time

The back-propagation algorithm can also be extended to the case of recurrent neural networks. To deal with this kind of systems a discrete time variable t is introduced. At time t all units in the network recompute their outputs, which are then transmitted at time $t + 1$. Continuing in this step-by-step fashion, the system produces a sequence of output values when a constant or time varying input is fed into the network. A a recurrent neural network behaves like a finite automaton and hence the problem is how to train such an automaton to produce a desired sequence of output values.

The simplest way to deal with a recurrent neural network is to consider a finite number of iterations only. Assume for generality that a network of n computing units is fully connected and that w_{ij} is the weight associated with the edge from node i to node j. By unfolding the network at the time steps $1, 2, ..., T$, it can be considered that this recurrent neural network as a feed-forward network with T stages of computation. At each time step t an external input $x(t)$ is fed into the network and the outputs $(o_1^{(t)}, ..., o_n^{(t)})$ of all computing units are recorded. The $n-$dimensional vector of the unit's outputs at time t is called the network state $o^{(t)}$. It is assumed that initial values of all unit's outputs are zero at $t = 0$, but the external input $x(0)$ can be different from zero. Figure 2.13 shows a diagram of the unfolded network. This unfolding strategy which converts a recurrent network into a feed-forward network in order to apply the back-propagation algorithm is called back-propagation through time or just BPTT.

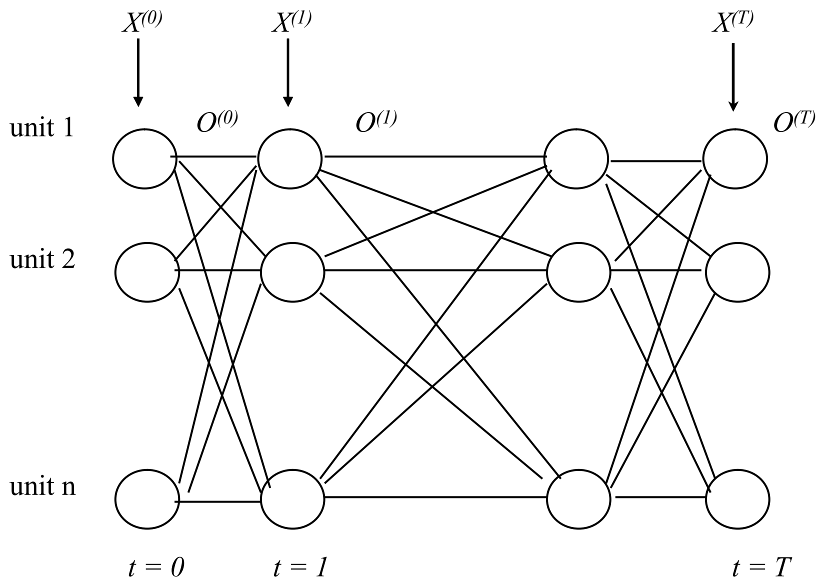


Figure 2.13: Back-propagation through time.

Let W stand for the $n \times n$ matrix of network weights w_{ij} . Let W_0 stand for the $m \times n$ matrix of interconnections between m input sites and n units. The feed-forward step is computed in the usual manner, starting with an initial m-dimensional external input $x^{(0)}$. At each time step t the network state $o^{(t)}$ (an n-dimensional row vector) and the vector of derivatives of the activation function at each node $o'(t)$ are stored. The error of the network can be measured after each time step if a sequence of values is to be produced, or just after the final step T if only the final output is of importance. For a more general case, denote the difference between the n-dimensional target $y^{(t)}$ at time t and the output of the network by $e^{(t)} = (o^{(t)} - y^{(t)})^T$. This is an n-dimensional column vector, but in most cases we are only interested in the outputs of some units in the network. In that case define $e_i(t) = 0$ for each unit i, whose precise state is unimportant and which can remain hidden from view.

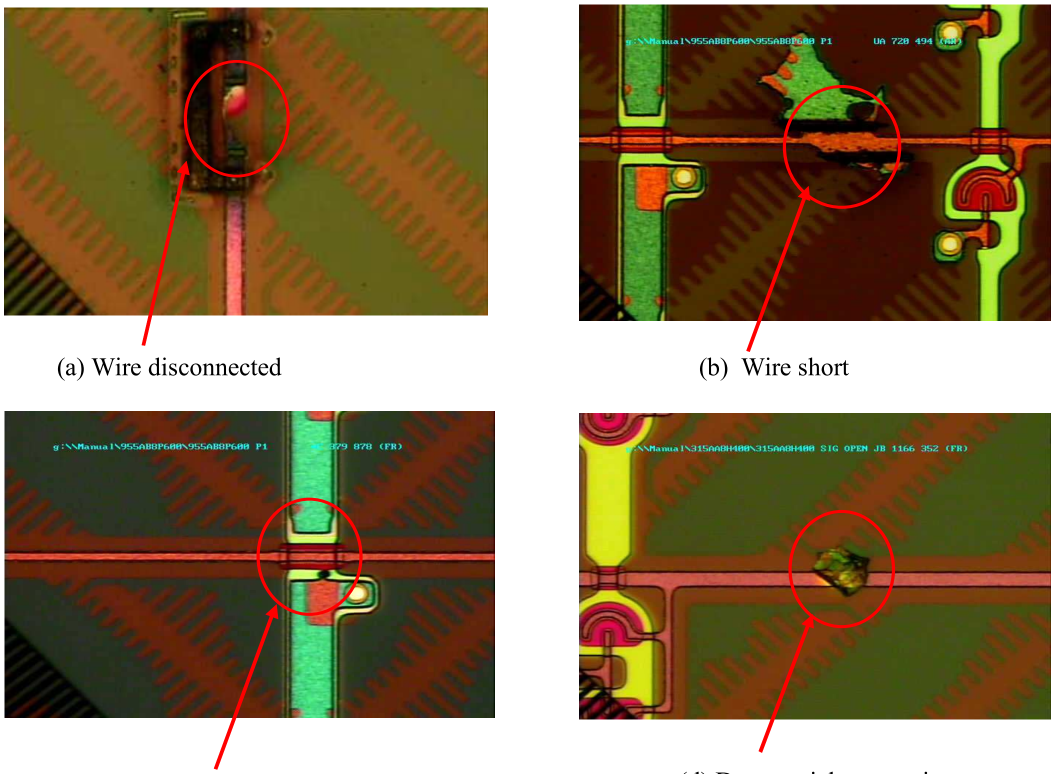
Chapter 3

Non-Contact Defect Inspection on Flat Panel Displays

3.1 Defects on Flat Panel Displays

Flat panel displays often produce surface defects due to highly automated fabrication process in highly controlled factory environments. Those defects are mainly electrical defects such as open circuits (open NGs) and short circuits (short NGs) within inside circuitry of the TFT array and sometimes dust particles and cracks. These dust particles and cracks can also be effective as open NGs or short NGs according to their size and existing location. Figure 3.1 shows some photographs of real NG situations taken by a micro camera, where Figure 3.1 (a) to (d) show an open circuit defect on a wire, a short circuit defect on a wire, a short circuit defect in between two layers and a dust particle on a wire which can effectively be an open circuit.

The existence of these NGs are costly for manufactures and must be detected and repaired in early stages of fabrication to minimize wastage and maximize their productivity. There are various methods of such defect inspections such as optical inspection method, direct pin probe method and the most advanced non-contact method proposed by Hamori et al.[29, 30, 31, 32] as described in chapter 1. The latter, Hamori et al. method, is described in detail in this chapter where its drawbacks and possible algorithmic improvements were the main inspiration based upon for the research in this thesis.



(a) Wire disconnected

(b) Wire short

(c) Short circuit between layers

(d) Dust particle on a wire

Figure 3.1: Various types NGs on flat panel display circuitry.

3.2 Non-Contact Inspection Method

As the demand for larger sizes of mother glasses, such as Generation 10,11 and 12, have been increasing, the demand for FPDs with high density pitch pattering of thin film transistor (TFT) arrays have also been increasing. Particularly, with the emergence of ultra-high definition 4K and 8K TVs and their expected growth in coming years, large scale FPDs with high pitch TFT arrays would be in mainstream [58, 59, 80]. However with increasing pitch pattern density there is a tendency for having defects, such as inter-layer short circuits between TFT lines, to increase. Under these circumstances, detection and repair of defects in early manufacturing stages have become significantly important [64]. As a result, the speed and the precision of defects detection have been major issues for the manufactures of FPDs and researchers alike.

The Non-contact defects inspection method for FPDs proposed by Hamori et al.[29, 30, 31, 32] is the most promising technique of defects detection in FPDs to-date, which is totally non-contact, utilizing a capacitor based sensor that scans over the TFT lines of mother glass panels of FPDs. The capacitor based non-contact sensor utilizing two electrodes, a feeding electrode and a receiving electrode, that scan parallel to each other across TFT lines over the mother glass of FPD panel. Figure 3.2 is an illustration of a non-contact FPD inspection system proposed by Hamori et al. During scanning, a known voltage is applied into TFT lines on the panel surface through the feeding electrode and is received through the receiving electrode capturing the voltage signal through an analogue to digital converter, which is sent to the host computer as a digitized waveform for analysis.

Figure 3.3 shows a typical waveform pattern of a captured voltage waveform through a non-contact sensor. The electrical defects, short circuits (short NG) or open circuits (open NG), on TFT lines will manifest themselves as peaks and toughs on the captured waveform, detection of which in effect produces the basis for detection of defects. Generally such waveforms are mixed with lot of random noises, external vibrations and other artifacts as shown in the Figure 3.3. The deviation at point a may be a random electrical noise, at point b may be a deviation caused due to a real electrical defects and at point c may be a vibration caused by an external force. Due to practical reasons in real production environments the gap between the surfaces of the scanning electrodes and the surface of flat panel are not uniformly even. This unevenness causes low frequency swinging or baseline fluctuations on the captured voltage waveform as visible in the figure.

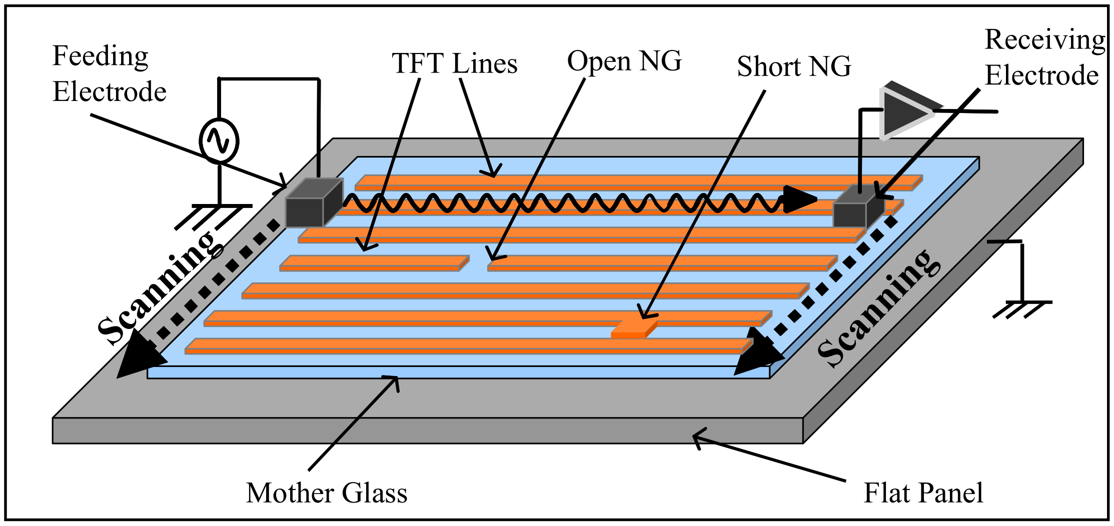


Figure 3.2: Non-contact inspection system of flat panel displays.

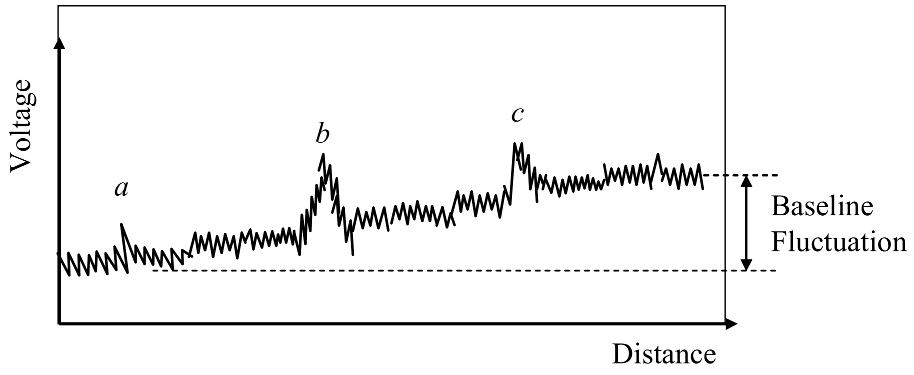


Figure 3.3: A typical pattern of a waveform captured by a non-contact sensor.

3.3 Detection of Defects by Thresholding

3.3.1 Single Channel Inspection Method

This method is basically to determine a TFT line as defective (having Open NG or short NG), where its voltage value exceeds a certain threshold level. This threshold level is a global value to the entire waveform and it is applied after some noise reduction filters are applied. The following Figure 3.4 explains the steps in thresholding processes used in the Hamori et al. method. The original waveform captured by a non-contact sensor consists of lot of noise including random noise and various environmental effects and vibrational effects as shown in Figure 3.4 (a). A series of filtering and other operations are applied to the waveform before trying to detect defect points on it. Initially, a moving average filter is applied to reduce high frequency random noises (Figure 3.4 (b)). Then low frequency swinging and baseline fluctuations of the waveform due to the unevenness of the gap between the panel surface and the sensor surface are neutralized by applying a derivative operator with a pre-determined step length (Figure 3.4 (c)). The resulting waveform is undergone again a moving average operator to remove remaining spike noises. Finally magnitude values of the waveform are compared with a pre-determined threshold value and the points that exceed the threshold level are considered as defect points as shown in Figure 3.4 (c).

However, the single channel inspection was not stable due to external electrical noises as well as mechanical vibrations. In addition, it was difficult to detect incomplete defects between defective wirings and normal wirings. Therefore for the purpose of decreasing of error margin due to the influence of noise and decreasing detection ratio of imperfect wirings or short circuits, a dual channel method was proposed by combining additional sensor as mentioned in the following section.

3.3.2 Dual Channel Inspection Method

In the dual channel method an additional sensor, located in a slightly shifted position, scans simultaneously. As shown in Figure 3.5, the dual channel system measures the voltage not only by a pair of the receiving electrode and the feeding electrode but also by another pair of same electrodes slightly shifted from the other pair to the scanning direction. In the system, two data sets of micro voltage signals with physical distance offsets are simultaneously recorded through two pairs

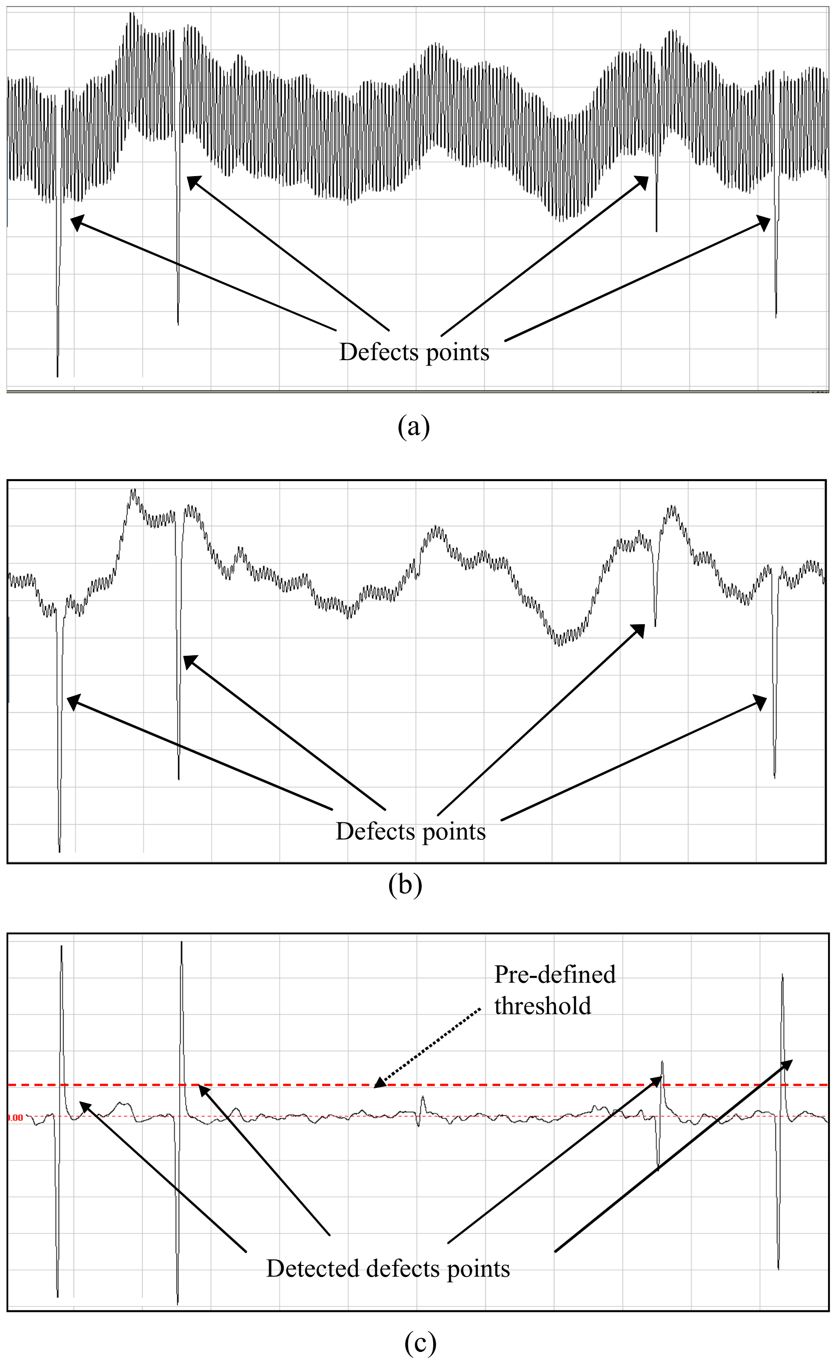


Figure 3.4: Defects detection by thresholding method; (a) Original waveform; (b) After noise suppression; (c) Thresholding on differential waveform.

of electrodes. By combining these two data sets measured simultaneously, missed detections can be reduced caused by noises and/or external forces.

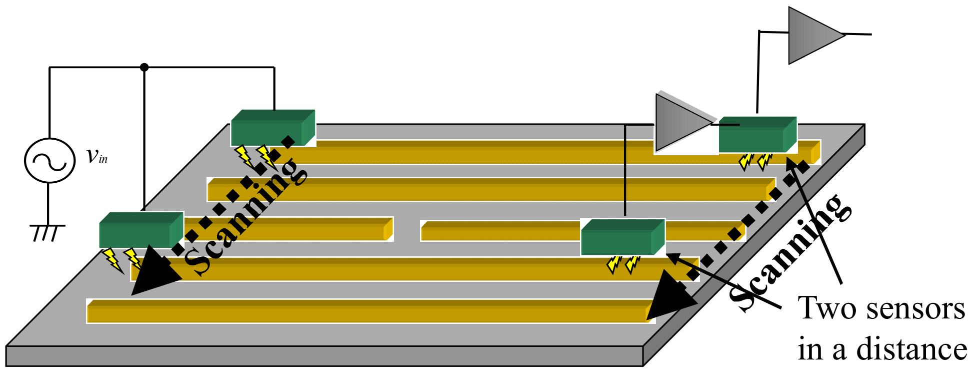


Figure 3.5: Configuration of a dual channel measurement system.

For example, waveforms of two data sets recorded simultaneously by two pairs of electrodes are shown in Figure 3.6, where the horizontal axis represents the time (the origin is where the scanning is started) and the vertical axis represents the measured voltage. In the waveform for the electrode Set-1 (CH1) in Figure 3.6, point A shows a change stemming from an electrical noise, point B shows a change stemming from a real electrical defect and point C shows a change stemming from a vibration caused by the external force. In the single channel method with only one pair of electrodes only the data set for CH1 is available. Therefore, the inspection system cannot discriminate between the change caused by defect (point B) and ones caused by other factors (points A and C), and judges all the changes as defects. In the dual channel method with two pairs of electrodes, it can discriminate among those changes because the data set for CH2 is also available. More specifically, since electrical noises, observed at point A, generally occur at random in both time domain and spatial domain, changes by electrical noises like point A are recorded by one electrode set (CH1) but not recorded by the other electrode set (CH2) like point A. Meanwhile, changes by defects as point B are recorded by both electrode sets (points B andB'), where point B' is later than point B by the time corresponding to the physical distance offset. In addition, since the mechanical vibration, as in point C, occurs in the total system at the same timing, changes caused by the vibration appear at the same time in the data recorded by two electrode sets. Figure 3.7 shows waveforms where the horizontal axis represents the distance from the origin of scanning and the vertical axis does the measured voltage. In Figure 3.7 where both waveforms have the same origin, only the changes caused by the defect (points B andB') occur at the same position.

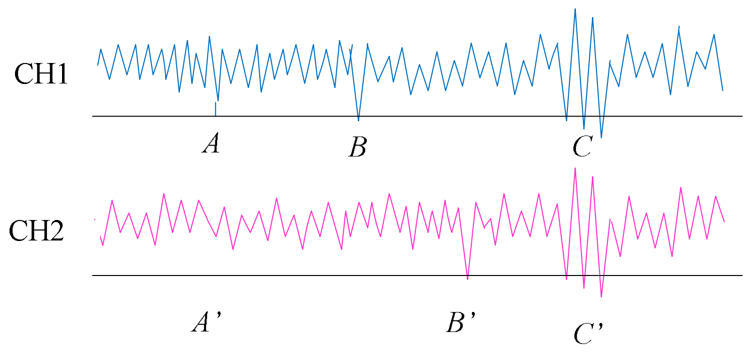


Figure 3.6: Comparison of measurement data on time axis.

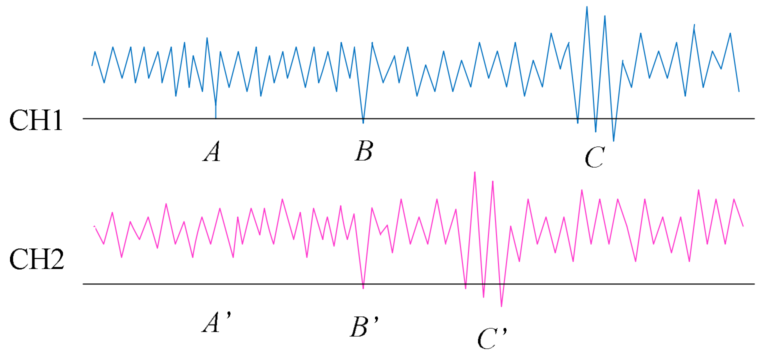


Figure 3.7: Comparison of measurement data on distance axis.

Taking into consideration of these features, the dual channel inspection system was able to discriminate wave features between the changes caused by defects and those caused by other factors. Their algorithm, of detection by reducing not only missed detection but also non-detection utilizing these features, can be described as follows. In the algorithm, f_n is the data series measured by CH1 and g_n is one measured by CH2, which are recorded in a given constant sampling interval (for example, in $10 \mu m$ interval) from the origin of scanning i is half the length (number of data points) of an interval where peculiar changes by defects are observed.

Detection Algorithm

The defects detection algorithm using the dual channel waveform data is described by the following 6 steps. In the algorithm, f_n is the data series measured by CH1 and g_n is one measured by CH2, which are recorded in a given constant sampling interval (for example, in 10 μm interval) from the origin of scanning i is of half the length (number of data points) of an interval where peculiar changes by defects are observed.

Step 1: Data Smoothing

Captured waveform data are always included with lots of noises due to various reasons such as random noise, vibration noise and other external effects etc. The random noise are always scattered in high frequency. For the purpose of removing such high frequency noises of measured data series f_n and g_n, first a moving average filter is applied for a given step size.

Step 2: Matching on Distance Axis

Since electrodes for CH1 and CH2 are placed at different positions, or with a distance offset, having different distances from the origin of scanning, both measured data series f_n and g_n involve physical distance offsets (a gap in distance, which corresponds to the distance between electrode sets). When f_n is recorded by CH1 at a position, g_{n+j} is also recorded by CH2 at the same position, which means that the distance offset is equal to the length of an interval containing j points of the measured data series. Considering a new data series $g'_n = g_{n+j}$ generated by shifting g_n with the distance offset, g'_n is a data recorded at the same position where f_n is recorded.

Step 3: Removal of Baseline Fluctuation

Due to practical reasons in real production environments the gap between the surfaces of the scanning electrodes and the panel surface are not uniformly even. This unevenness causes low frequency swinging or baseline fluctuations on the captured voltage waveform. Compute differential series of f_n and g'_n as $df_n = f_{n+1} - f_n$ and $dg'_n = g'_{n+1} - g'_n$ to remove baseline fluctuations.

Step 4: Emphasis of Peculiar Changes

By getting the product of the differences of f_n and g'_n such that $h_n = (f_{n+i} - f_n) * (g'_{n+i} - g'_n)$, steep changes on waveforms are emphasized which are the characteristics of defect points on the waveform data.

Step 5: Smoothing of Spike Noises

In order to smoothen spike noises, compute the sum of $2i$ points of h_n since the length of the defective part (number of data points) is assumed to be $2i$ such as;

$$h'_n = \sum_{k=1}^{2i} h_{n+k}. \tag{3.1}$$

Step 6: Judgment of Defect

The final defect judgment is determined by using a pre-determined threshold value by comparing the value of h'_n with the threshold. If the value of above h'_n at any given point in the waveform exceeds the given threshold level it is considered to be a defect point.

Defects inspection machines, adopting this technique proposed by Hamori et al.[29, 30, 31, 32] for flat panel displays gained a high recognition from the manufacturers immediately after its inception. Currently a large number of manufacturers, market share-wise a much bigger portion, have installed this machine as it seems to be the future of electrical defects inspection of TFT arrays of FPDs whether it is LCD or Plasma or LED.

3.4 Drawbacks in the Thresholding Method

Though the above mentioned non-contact inspection method based on capacitor based sensor is the most advanced technique to-date, it has still some drawbacks which are beyond the control of the sensor circuitry as describes below. The thresholding method described above is quiet appropriate as long as environmental effects such as the machine temperature and external vibrations and the gap between surfaces of the sensor and the panel remain firmly stationary, which minimizes extra noises and unnecessary deviations on the voltage waveform. However in real production lines the temperature and the external vibrations can vary from time to time and from machine to machine. It can also be varied from location to location of machines even inside the same factory.

Keeping the gap fixed between the sensor and the panel, during scanning, is also not an easy task, so that the voltage signal shows fluctuations or swinging. All of these factors severely affect the pattern and the noise level of the captured voltage waveform and determining threshold parameters for the software program is a difficult task as shown in Figure 3.8. Whenever an inspection machine is changed or its location is changed, operators have to look carefully several waveforms and set threshold parameters manually. On the other hand, the threshold parameter set by the operator is a global value, which applies to the entire waveform. However most of the above effects to the voltage signal are local effects, in which, taking a global threshold value as the deciding factor in a highly locally dependent feature space is lacking appropriateness.

Figure 3.8 illustrates the difficulty of determining of a proper threshold value in thresholding method in detail , as shown in Figure 3.8 (a), points B, C and E are real defect points and points A and D are not defects but sudden fluctuations due to change of gaps between the sensor and the panel. The corresponding points for real defects on the differentiated waveform (Figure 3.8 (b)) are $B^{'}$, $C^{'}$ and $B^{'}$. Out of these 3 defect points only $B^{'}$ and $E^{'}$ can be detected using the given threshold and point $C^{'}$ is difficult to be detected by the threshold although it is actually a defect

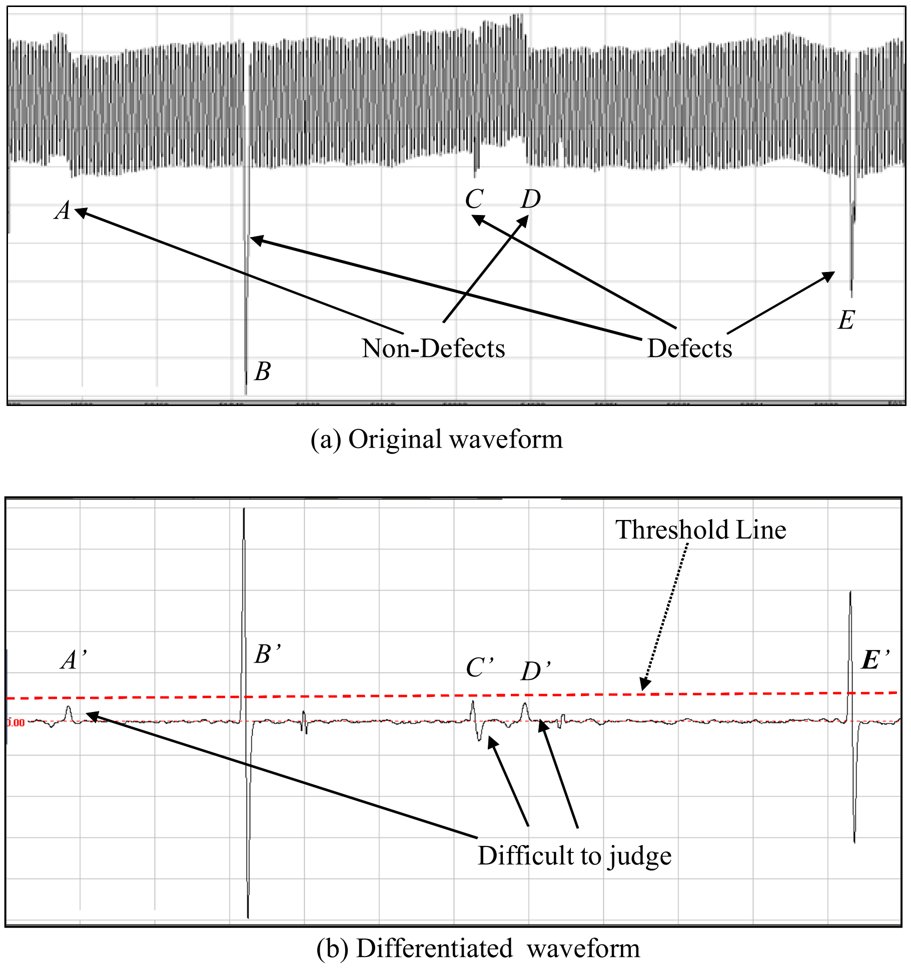


Figure 3.8: Difficulty of determining a proper thresholding value (a) Original waveform (b) Differential waveform.

point. If the threshold level is further lowered the missed detected point C' can be detected but in the same time points A' and D' can also be detected as defects, which will be false detections. In manufacturer's standpoint both missed detections and false detections are costly.

This hurdle of determining a proper threshold value is difficult to leap over as long as the threshold value is global and the features around real defect points are local. The lower the threshold value is set the higher the ratio of false detections appear whereas the higher the threshold value is set the higher the ratio of missed detections occur. Moreover, the captured waveform data always consists of uncertain amount of noise which are difficult to judge as from what conditions they arise. Therefore parameter determination is always not easy, such as the step size of the moving average in step 1 and the gap size for differentiation in step 3 in the above algorithm.

3.5 Existing Line Scan Method and its Drawbacks

In the above mentioned non-contact FPD inspection method, the defective TFT lines (or NG lines), which are detected using thresholding or feed-forward neural network or recurrent neural network methods, have to be scanned again using a micro camera based sensor (called NG sensor) to determine exact locations of NGs (open circuits or short circuits) on defective lines. This process is necessary for operators in order to determine if NGs are to repair or not. Once defective lines are observed by the non-contact sensor, this line scanning by a NG sensor must obviously be a shortest possible path. The currently used motion path of the NG sensor is a simple top-down unidirectional method (Figure 3.9) which is heavily time consuming particularly on larger glass panels such as 3m x 3m panels. The scan time may be negligible for a few number of NG lines or for small sizes of mother glasses. However due to current trend of high demand of bigger sizes of flat panels it is no longer negligible. Moreover with the emergence of finer pitch patterns for 4K and 8K TVs, the number of lines for a panel becomes bigger and hence the possibility of more defective lines on panels increases. Therefore, it is necessary to find a more realistic and intelligent approach of the possible shortest path of NG scanning.

3.6 Proposed Solutions

To address the drawbacks mentioned in the section 3.4, from next chapter , by formulating the problem of detecting NGs on a noisy waveform to a data driven

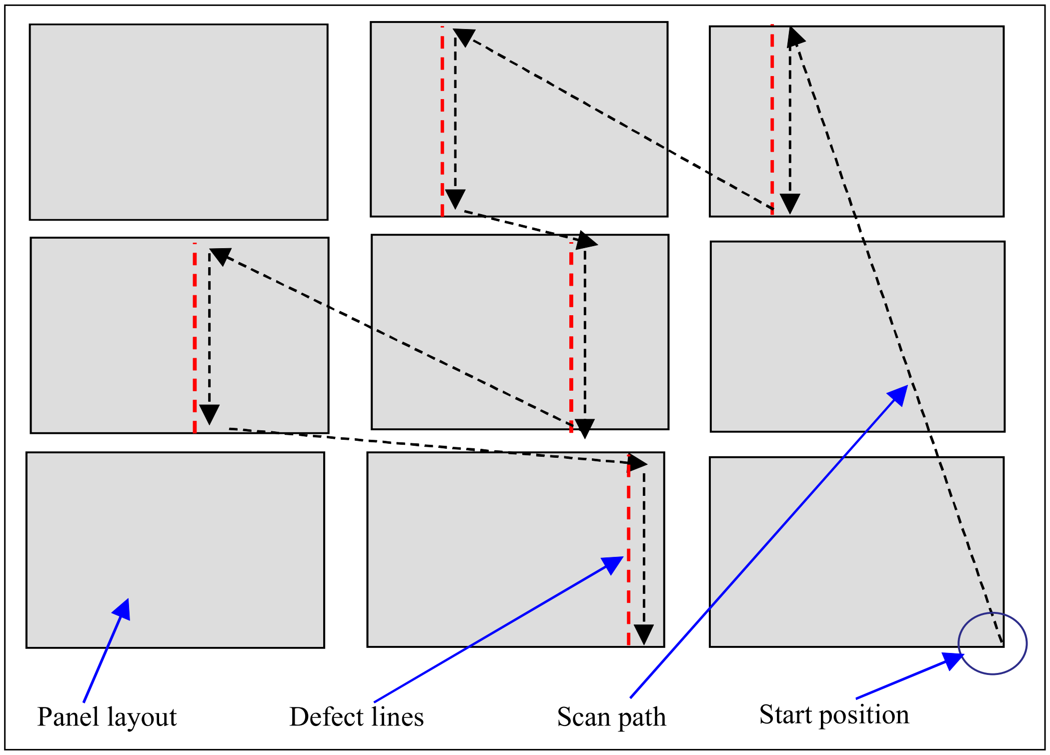


Figure 3.9: Existing unidirectional top-down NG scan method.

non-linear classification problem we attempt to solve it by a feed-forward neural network. The single hidden layer feed-forward neural network is trained by error back-propagation algorithm. There are three inputs to the network which are characteristics on and around candidate points on the waveform, which are Signal to noise (SNR) at the candidate point, residual difference in a neighborhood of the candidate point and the change of wave length in the same neighborhood in the differential waveform.

Then in the next chapter, the method extends to a more realistic and noise resilient recurrent neural network which overcomes some drawbacks in feed-forward network. The topology of the recurrent network and its training by back-propagation through time are evolved by a genetic algorithm based multi-objective optimization algorithm. The inputs for the recurrent neural network are same as those used in the feed-forward neural network. This evolutionary multi-objective optimization algorithm optimizes all topological parameters of the recurrent neural network including the number of layers and number of neurons in each layer and each network in the population trains during the evolution process using back-propagation through time to determine if they can be trained well.

Then in the following chapter, for the purpose of addressing the drawbacks mentioned in the section 3.5, the problem of finding a shortest scan path of the distance traversed by the NG sensor to locate exact locations of NG points, is also addressed by a Kohonen's self-organizing map based solution. First, the problem is approximated to an asymmetric traveling salesmen problem with precedence constraints, and then, an algorithm with a combination of a modified self-organizing map, a 2-Opt algorithm and a repair algorithm for node assignment is proposed for the optimization of path of NG sensor.

Both of defect detection method by feed-forward and recurrent neural network approaches and the NG sensors scan path optimization method by kohonen's network were proved much superior than the existing thresholding method and top-down unidirectional scan method.

Chapter 4

Defects Inspection Using a Feed-Forward Neural Network

Feed-forward neural network (FNN) or a multi-layer perceptron is a well-known and widely used class of neural networks. The popularity of feed-forward networks derives from the fact that they have been applied successfully to a wide range of information processing tasks in such diverse fields as speech recognition, financial prediction, image compression, medical diagnosis and classifications. Feed-forward neural networks are trained, rather than programmed, to carry out the chosen information processing tasks. Training a feed-forward neural network involves adjusting the network so that it is able to produce a specific output for each of a given set of input patterns. Since the desired inputs are known in advance, training a feed-forward neural network is an example of supervised learning. A good analysis on neural network modules and training algorithms can be seen in Auda and Kamel [8]. Zhang [83] have also presented another good survey of feed-forward neural networks in classification. When looking at literature on applications of feed-forward neural networks it can be seen that a majority of work done in classification theme. They are bio-engineering applications such as feature classifications of electrocardiogram (ECG)[37, 51] and Electroencephalography (EEG) [17, 62, 66]. Chambayil et al. [17] have proposed EEG eye blink classification using a feed-forward neural network. Though the purpose is different from the work in this thesis the basic analysis is based on a wave signal captured from an electrical signal which also include noise.

Other types of wave feature classification using feed-forward neural networks includes the work done by Du et al. [23], which extracts features from a sine wave, and Slazar et al. [71] proposed a feed-forward neural networks for defect detection in non-destructive evaluation by sonic signals. Sathiya et al. [72] proposed another interesting work by using a feed-forward neural network, which extracts ocean wave height data in highly noisy environment. There are more promising works presented in waveform feature extraction area using feed-forward neural networks [11, 38, 65, 75].

As described in Chapter 1, the focus in this chapter is to classify voltage waveform data captured from a non-contact defect inspection sensor proposed by Hamori et al. [29, 30, 31, 32], using a feed-forward neural network. Other applications of FNN in similar patterns of data can be identified as ECG and EEG feature detection [17, 62, 66], wave feature detection [23, 71, 72] and financial and stock related time series analysis.

4.1 Formulation of the Problem

In the thresholding method used in Hamori method [29, 30, 31, 32] for classification of waveform data, the criteria of determining a threshold value lacks appropriateness since the feature variations on and around defect points are largely local features whereas the threshold is a global value. Hence it requires an adaptive algorithm with a high degree of accuracy and efficiency since the system is largely data driven and the patterns of defect points on waveforms are highly non-linear. The cross correlation methods can be used to detect such patterns on waveform data, which is a measure of goodness of fit with a pre-selected pattern. The level of fitness of about 80% or more corresponds to patterns in data that are easily discerned as good matches by the human eye. However having various patterns of waveforms with various patterns of defect points on them it requires to prepare and store hundreds of patterns if not thousands, which would be a huge time consuming exercise. Moreover, the features around defect points on waveforms are varied from channel to channel, from location to location etc. So that a technique such as a neural network that can learn the environmental effects and memorize, must be a much suitable way of addressing the problem. Since artificial neural networks are intelligent agents that can handle effectively a large amount of dynamic, non-linear and noisy data [8, 39, 67]. It also can observe, learn and memorize from the experience before performing a particular task. Therefore the most reliable approach must be an intelligent approach such as using a feed-forward neural network since a neural network can be trained by feeding known data before actually put into perform and can keep the adaptability.

4.2 Inputs Selection

Feed-forward neural networks are trained by giving a known set of input data having all variations in the input space. When classifying input patterns by the network all the characteristics of the input space that are influential for the classification must be considered. In our case too, any feature of the input pattern or (input point on the input waveform) that can be considered as influential to the output must be considered as an input to the network. Therefore it is necessary to throughly observe and analyze a large set of input data before determining what the the input parameters. After such analysis of a large set of waveform data consisting all kinds of defect points and normal regular patterns including the neighborhoods of defect points, following three features were identified as inputs to the network (input vector $\mathbf{x}$), namely signal to noise ration (SNR), residual difference, and change of wave length on the differential waveform. All of these input parameters $\mathbf{x}(x_1, x_2, x_3)$ are picked within a pre-determined length of neighborhoods of possible candidate points on the waveform whereas candidate points are selected by a simple low level threshold such that it may include many false detection but not to miss any of them
.

4.2.1 SNR

If the characteristics within a neighborhood of a defect point on a waveform are observed (Figure 4.1) it is understood that there is a sharp deviation of magnitude. In other words the level of the signal at a particular point shows a considerable deviation against the level of background noise, which means a change of signal to noise ratio (SNR). SNR is a measure of signal strength relative to background noise. Therefore SNR is considered as the first input (x_1) of the input vector $\mathbf{x}$ and is taken as:

$$x_1 = SNR = \frac{\mu}{\sigma}, \tag{4.1}$$

where μ is the mean value and σ is the standard deviation of the waveform within the selected neighborhood around a candidate point.

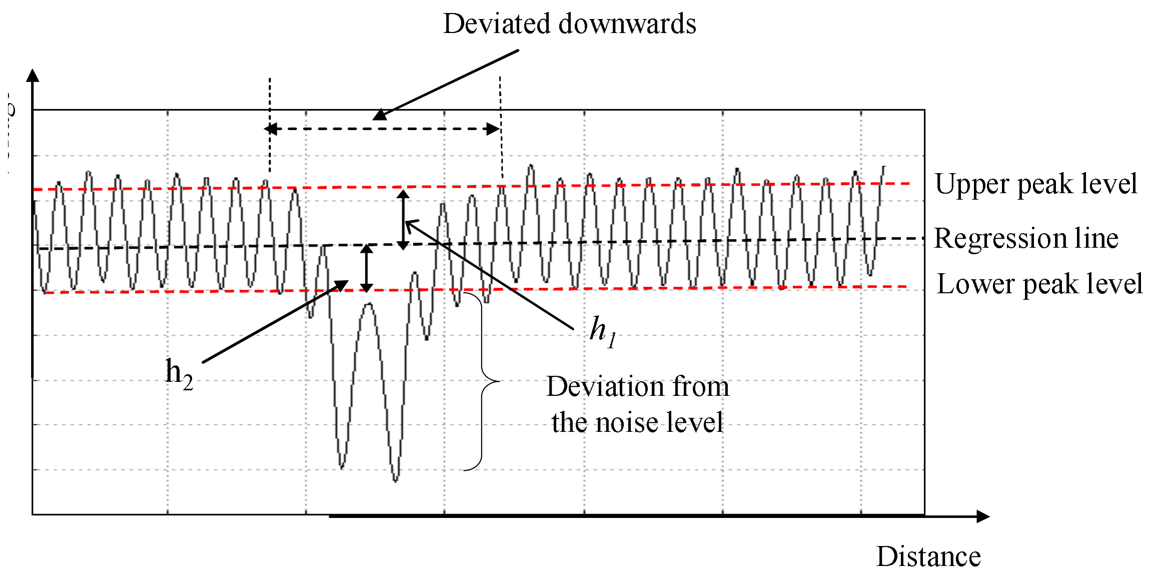


Figure 4.1: Picking up SNR and residual difference from a neighborhood of a candidate point.

4.2.2 Residual Difference

Besides the sharp deviation at the defect point, the neighboring area consisting of a few wave lengths can also be seen deviated towards the same direction as main deviated point (Figure 4.1). This particular feature of the waveform within the neighborhood is measured as the difference of average upper peak level with the regression line (h_1) and the difference of average lower peak level with the regression line (h_2). In other words the residual difference of upper and lower peek levels in the neighborhood is taken as the second input (x_2) of the input vector $\mathbf{x}$ and is taken as:

$$x_2 = |h_1 - h_2|. \tag{4.2}$$

4.2.3 Change of Wave Length

In the original waveform it shows a considerable change of wave length at a defect point (Figure 4.1) and is taken as the next input to the network. Though this change of wave length appears in the normal waveform, it is easier to measure on the differentiated waveform as seen in Figure 4.2. So that the rate of change of wave length of a defect point from that of average wave length in the normal area is taken as the third input (x_3) of the input vector $\mathbf{x}$ and is taken as:

$$x_3 = \frac{|D - d|}{d}, \tag{4.3}$$

where D is the wave length at the input point and d is the average wave length at neighborhood.

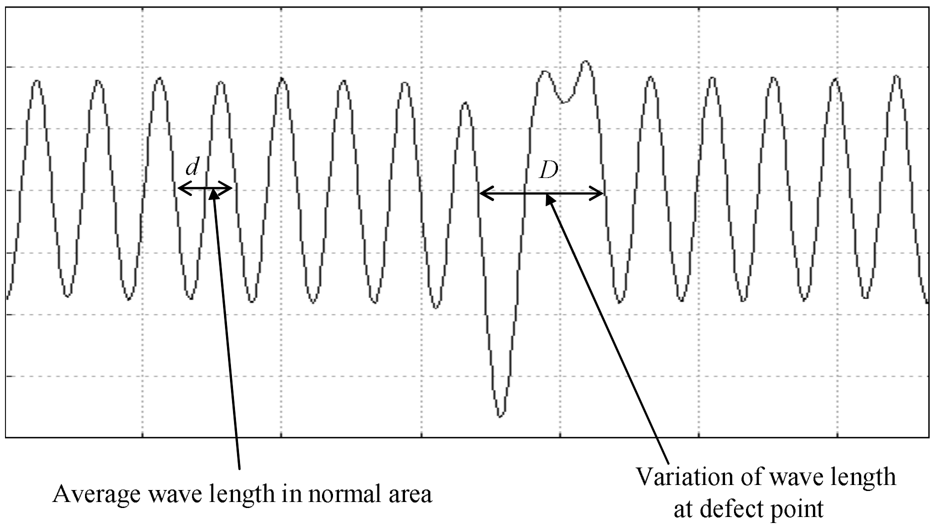


Figure 4.2: Picking up the change of wave length on the differential waveform.

4.3 Architecture of the Feed-Forward Neural Network.

4.3.1 Topology of the Network

Determining the best topology of a feed-forward neural network for a certain problem is one of the most important tasks. Therefore, various topologies with various number of hidden layers with various number nodes were tested using a selected good set of input data. Convergence pattern and the convergence limit of the error graphs during error back-propagation of many of them were not as expected as described and shown in chapter 2 and Figure 2.12. Most of their error convergence levels were much higher or the error curve fell rapidly but converged to a higher level. However After testing numerous topologies of neural networks, we have found that the most reliable network for our problem is a 4-layer feed-forward neural network such that an input layer with two units, two hidden layers with 2 units and 3 units respectively and an output layer with one unit (Figure 4.3). The input feature vector $\mathbf{x}(x_1, x_2, x_3)$ consists of 3 components (above mentioned 3 input parameters), the weight space consists of 27 weights which is inclusive of weights associated with bias inputs to each unit which are not depicted in the figure and the output vector **y** is single component. Then the total network function Net can be represented as:

$$\mathbf{y} = Net(\mathbf{x}). \tag{4.4}$$

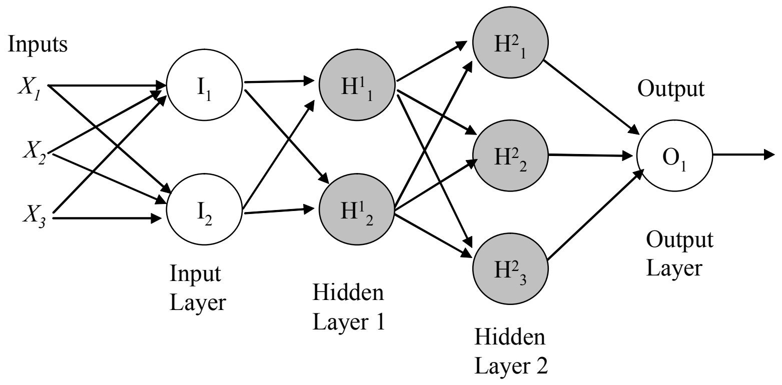


Figure 4.3: Topology of the feed-forward neural network.

4.3.2 Activation Function

The activation function of each computing unit in the network is in the form of a sigmoid function (logistic function) because training of the network is to be done by error back-propagation algorithm, which requires the activation function to be a continuous function [24, 69].

Since the back-propagation requires computation of gradients of the error function at each iteration step, the continuity and differentiability of the error function must be guaranteed. Therefore the activation function f of each unit in the network is taken as:

$$f(X) = \frac{1}{1+e^{-X}}\ , \tag{4.5}$$

and

$$X = \sum_{i=1}^{n} w_i x_i \ ,$$

where $x_i (i = 1, .., n)$ are the inputs to the unit (in case of input layer those are the components of the input vector x, and in case of a hidden or output layer those are outputs from the previous layer) and w_i are the weights associated with each such input.

4.3.3 Network Function

The network function (4.4) of the above FNN (Figure 4.3) can then be explicitly expressed as:

$$y = f\left(\sum_{i=1}^{3} w_{ol}.(f(\sum_{k=1}^{2} w_{lk}.(f(\sum_{j=1}^{2} w_{kj}.(f(\sum_{i=1}^{3} w_{ji}xi + B_j^I)) + B_k^{H1})) + B_l^{H2})) + B^o \right) , \tag{4.6}$$

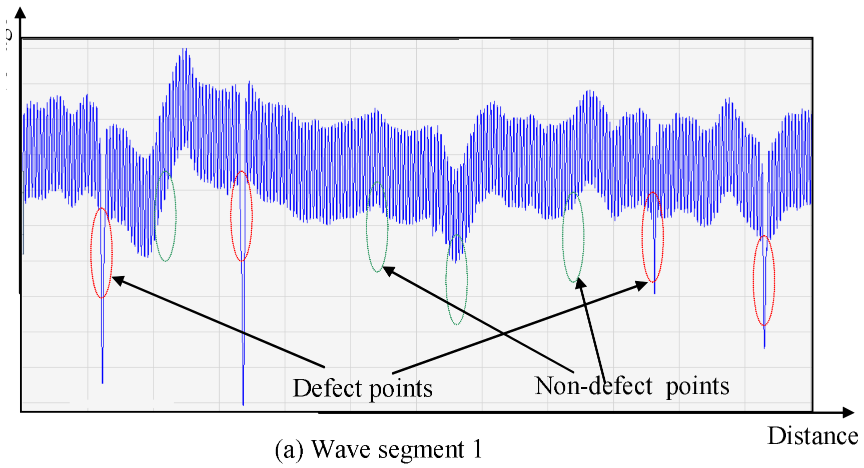
where w_{ij} is a weight in input layer connecting i^{th} input and j^{th} neuron, w_{kj} is a weight in hidden layer 1 connecting j^{th} neuron in the input layer and k^{th} is a neuron in the hidden layer 1, w_{lk} is a weight in hidden layer 2 connecting k^{th} neuron in the hidden layer 1 and l^{th} neuron in the hidden layer 2 and w_{ol} is a weight in the output layer connecting l^{th} neuron in the hidden layer 2 and the output neuron. B_j^I is the bias for the j^{th} neuron in the input layer, B_k^{H1} is the bias for the k^{th} neuron in the hidden layer 1, B_l^{H2} is the bias for the l^{th} neuron in the hidden layer 2 and B^O is the bias for the output neuron. x_i is the i^{th} component of the input vector $\mathbf{x}$ and f is the common activation function (or the sigmoid function described in equation 4.5) of the network.

4.4 Training by Back-Propagation

A feed-forward neural network is a computational graph whose nodes are computing units and whose directed edges transmit numerical information from node to node. In the network each arrow from left side to right side (Figure 4.3) is associated with a synaptic weight and those weight values must be optimized before use in a real situation task. The combination of those weights, which minimizes the error, is said to be the optimal solution to the learning problem. Since there are many weights in the network associating each input in a layer to the each unit in the next layer, we don't know how much each of those weights is to blame for the final error and divvy up the adjustment among these weights proportionately. Therefore this problem can be called a blame assignment problem or a credit assignment problem. The back-propagation solves this problem, as its name depicts it looks for the minimum of the error function in weight space using the method of gradient descent. A set of known inputs comprising both NG points and OK points in all sorts of patterns of input data will be used as the training data set. We also have adopted an adaptation technique [67] in order to accelerate the convergence of the error function. It is an adaptive step algorithm that the step size is increased whenever the algorithm proceeds down the error function over several iterations. When the algorithm moves over a valley of the error function the step size is decreased.

Training of the network was carried out with the error back-propagation algorithm by using a set of hand picked data containing 50 defect points (NG) and 50 non-defect points (OK) covering every possible pattern of defect points and non-defect points as shown in Figure 4.4. The data were captured by OHTs three different GX-3 High speed LCD/PDP testers in different factories. The set of 3 input parameters was picked from those selected data points and used as the input data set for back-propagation training of the neural network.

As described in section 4.3.1, numerous topologies of feed-forward neural networks with both one hidden layer and two hidden layers were tested to determine the best one for this problem. Error graphs of most of them were either moved a little bit



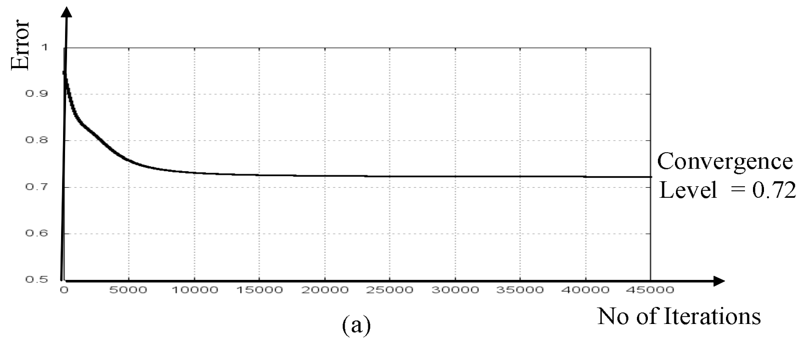
(a) Wave segment 1

Figure 4.4: Picking up input data (both NG points and OK points) for network training.

horizontally or fell suddenly and then in both cases converged to a higher level than the excepted level. Figure 4.5 (a) and Figure 4.5 (a) are two of such unacceptable error graphs with a single hidden layer consisting 3 units and 2 units respectively. As shown, their convergence levels are much higher than the level in our selected network with 2 hidden layers as shown in Figure 4.6, which converges until the given limit of 0.01. Therefore a 4 layer network with 2 hidden layers was judged as the best topology for this problem as its error graph converged until the given small convergence limit.

4.5 Training Results

Initially all the weight values of the above selected network were randomly set to small values between 0 and 0.1 and the learning constant γ was set to 0.1. The error back-propagation was performed iteratively until the error function converged to a level smaller than a pre-determined small value as shown in Figure 4.6. The convergence time of the error graph was considerably reduced by using an adaptive technique that adjusts the step length according to flow of the error function. Table



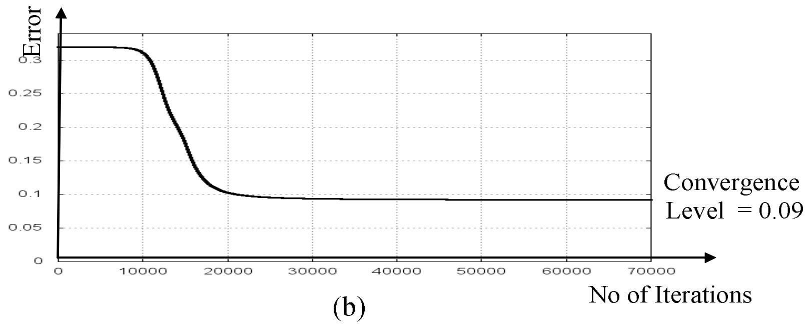


Figure 4.5: Error convergence in a single hidden layer network: (a) 3 neuron hidden layer, (b) 2 neuron hidden layer.

4.1 and Table 4.2 show the training results.

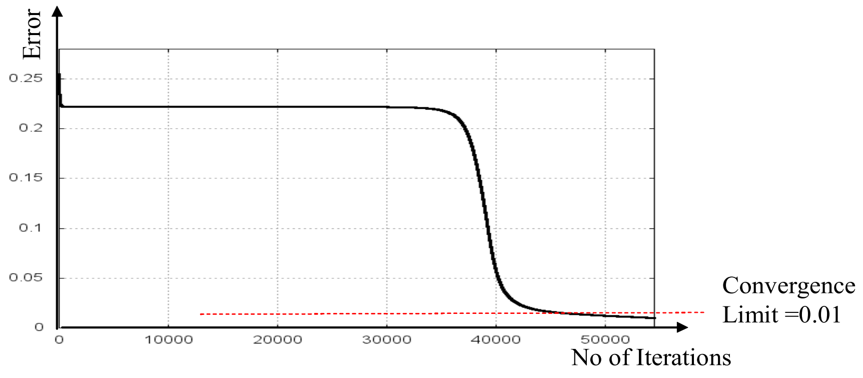


Figure 4.6: Convergence of error graph in back-propagation.

As seen in Table 4.2, although the initial weight values were set between 0 and

Table 4.1: Parameters of training by back-propagation.

No of training data	100
Initial weight values (Random)	$0 \sim 0.1$
Initial step size (γ)	0.1
Error convergence limit	0.01
No of iterations	54454

0.1 the final optimized weight values, after error back-propagation, range from even smaller values to even bigger values. A positive weight represents an excitatory connection whereas a negative weight represents an inhibitory connection.

Table 4.2: Network weights after error back-propagation training.

Input layer	Hidden layer 1	Hidden layer 2	Output layer
0.4299	−0.0869	0.4477	−0.56339
−0.2330	−1.5049	0.153	−1.1827
−0.1928	3.7513	−0.5832	3.6198
0.0507	−3.1422	−0.5832	3.7065
1.7997	0.4497	−1.6563	
−0.5134	−0.2282	3.8409	
0.2399		−0.5956	
0.1892		−1.6760	
		2.2174	

4.6 Defects Detection Using the Trained Neural Network

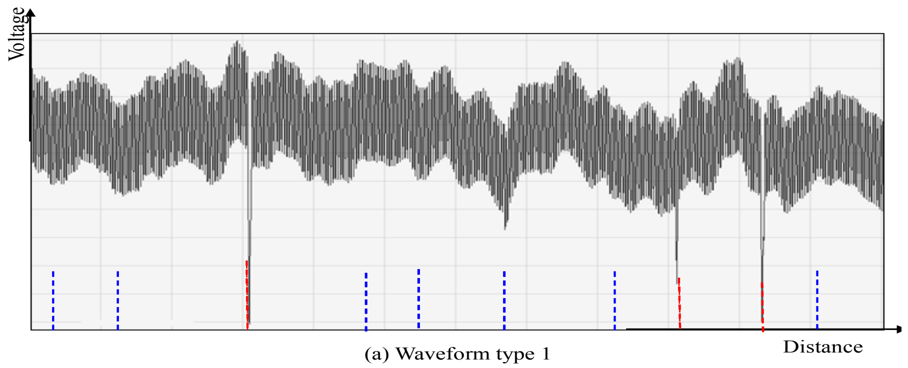
The feed-forward neural network (with parameters shown in Table 4.2), trained by back-propagation algorithm, was used to detect defect points on waveform data captured by a non-contact sensor scanned over TFT lines of flat panels displays. A simple moving average filter was initially applied to smooth waveform data before picking candidate points to the network. Then candidate points were selected using a lower threshold value than the threshold value used in Hamori et al. method [29, 30, 31, 32] and the three input parameters to the network were picked from those candidate points and entered into the network. Figure 4.7 shows some detection results on 3 different waveforms captured from 3 different machines in different locations. All of those voltage waveforms consist of different levels of random noises, external vibrations and baseline fluctuations on them but were able to detect using this method correctly. The points marked at dashed lines in Figure 4.7 (a) (both red and blue) are candidates that were selected as inputs to the network, and there were 3 defects (at red dashed lines) detected correctly out of 10 candidates. Similarly Figure 4.7 (b) shows 3 defects detected out of 8 candidates and Figure 4.7 (c) shows 2 defects detected out of 26 candidates correctly.

Table 4.3: Comparison of missed detections and false detections between FNN based method and thresholding method

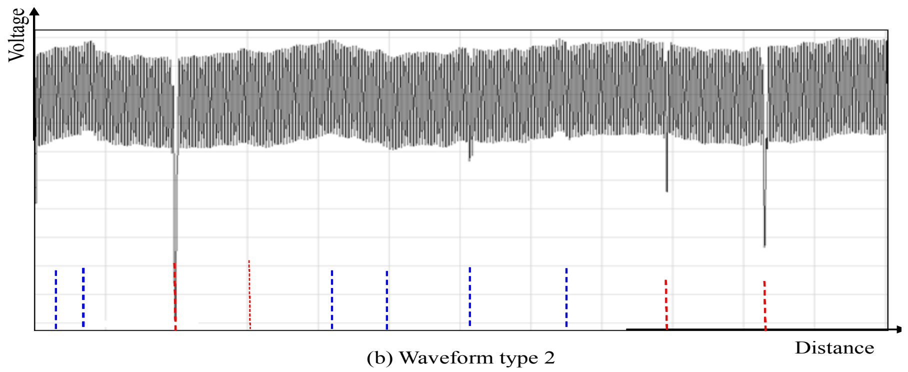
	Total defects	Missed detections		False-detections	
		Old method	New method	Old method	New method
Data set 1	40	11 (27.5%)	3 (7.5%)	10 (25.0%)	6 (15.0%)
Data set 2	55	13 (23.6%)	3 (5.4%)	16 (29.0%)	7 (12.7%)
Data set 3	36	7 (19.4%)	1 (2.7%)	11 (30.5%)	5 (13.8%)
Total	131	31 (23.6%)	7 (5.3%)	37 (28.2%)	18 (13.7%)

Table 4.3 shows a comparison of missed detections and false detections between this method and existing thresholding method. It shows comparison results for 3 different data sets captured from 3 different machines. The ratio of missed detections was dropped to around 5% from existing range of 20% to 30% whereas the ratio of false detections wa dropped to below 15% from existing range of 20% to 30%.

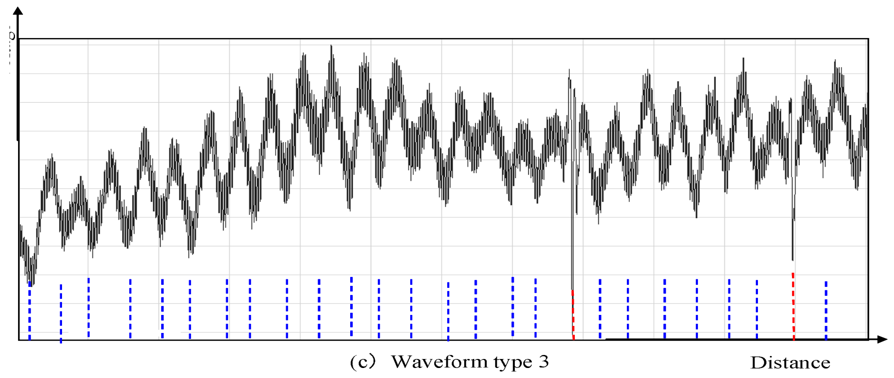
In addition, due to the reduction of ratio of false detections, the number of repairs were able slash drastically and due to the reduction of missed detections, manual



(a) Waveform type 1



(b) Waveform type 2



(c) Waveform type 3

Figure 4.7: Detection results on 3 different waveforms captured by 3 different machines in 3 different locations.

check ups were reduced and customer satisfaction was increased. Moreover, manual workload for threshold setting for operators, whenever the machine or the location or the product type to be tested is changed, was also decreased with the use of already trained network. As a result the overall testing time was improved and hence this method can be considered as fast and adaptive.

Chapter 5

Extension to an Optimized Recurrent Neural Network

Recurrent neural networks (RNN) are another type artificial neural networks that are fundamentally different from feed-forward neural network architecture in the sense that they not only operate on an input space but also on an internal state space, a trace of what already has been processed by the network. In other words, an internal feedback can be processed together with external inputs at a RNN [67]. RNNs behave like biological recurrent networks that are found in the brain. Since feedback is ubiquitous in the brain, this task in general could include most of the brain's dynamics.

There are many types of formal RNN models. Discrete-time models are mathematically cast as maps iterated over discrete time steps (such as $n = 1, 2, 3, ...$). Continuous-time models are defined through differential equations whose solutions are defined over continuous time t. Especially for purposes of biological modeling, continuous dynamical models can be quite involved as activation signals on the level of individual action potentials. On the other hand engineering applications often use discrete-time models of RNNs.

In the settings of reinforcement learning, there is no teacher providing target signals for the RNN, instead a fitness function or reward function is occasionally used to evaluate the RNN's performance, which is influencing its input stream through output units connected to actuators affecting the environment.

In literature, there is a great extent of discussions on RNNs, and good surveys can be found in [8, 9, 77]. Connor and Martin [19] and Huang et al. [35] also presented interesting RNN approaches for time series predictions. Dolinsky and Takagi [22] proposed an RNN based method for learning of naturalness of a system. Some RNN approaches have been proposed [16, 75] for classification problems as well.

The focus in this chapter is to address the drawbacks in feed-forward neural network based method for defect inspection on flat panel displays discussed in chapter 4. The FNN based approach has some drawbacks such as difficulties in topology determination process and poor performance in noisy data environment. One of the major reasons, why an RNN is brought into this problem, is because it is more resilient on noisy and imperfect inputs.

5.1 Proposed Method Using Multi-Objective Evolutionary Optimization

After careful observation of the variation of patterns of waveform data from various environments with varying levels of noise, it was clear that a recurrent neural network would be the best approach to the problem. RNNs are fundamentally different from feed-forward architecture in the sense that they not only operate on an input space but also on an internal state space, a trace of which has already been processed by the network. In other words, an internal feedback can be processed together with external inputs in an RNN. One of the major reasons, why an RNN is brought into this problem, is because it is more resilient on noisy and imperfect inputs.

Since choosing an appropriate RNN is also not an easy task, initially a RNN with one input layer and one hidden layer with unknown number of units was considered. Making the network recurrent there must be one or more feed-back connections, so that, the hidden layer itself in number of past iterations keep as inputs to hidden layer itself as well as to the output layer. Moreover, the input layer in number of past iterations also keep as input to the hidden layer as shown in Figure 5.1. The determination of these unknown numbers of units in input layer and hidden layer and the unknown numbers of hidden layers and input layers in past iterations is the major task in this chapter. As it is not easy by trial and error approach, an evolutionary optimization approach is the best way. Therefore a genetic algorithm based multi-objective evolutionary optimization approach was used to determine those parameters. During the optimization process the relevant RNN is trained by back-propagation through time. This process optimizes all the parameters of the RNN and trains that RNN by back-propagation through time and hence it is a multi-objective optimization algorithm. The optimization algorithm is described in the following sections in detail.

Choosing an appropriate topology of an RNN was again a major hurdle as it was even difficult than the FNN method by using the trial and error approach. Therefore a genetic algorithm based evolutionary optimization approach to determine the topology of the RNN is employed. It is clear that topologies generated by a genetic algorithm may contain many superfluous [67, 70] components such as single nodes or even whole clusters of nodes isolated from the network input. Such units, called passive nodes, do not process the signals applied to the network sensors and produce only constant responses that are caused by the relevant biases. A simplification procedure or a list of constraints can be used to remove passive nodes and links so that the input/output mapping of the network remains unchanged.

In the literature, there are various types of approaches of RNN for feature classification in waveform type data and training [8, 9, 77]. Haseken and Stagge [36] presented a recurrent neural network for noisy time series data classification and various techniques have been used for training of RNN [22, 55].

Among various methods of topology optimization in the literature, Delgado et al. [21] proposed a technique for simultaneous training and topology optimization of RNNs using a multi-objective hybrid process based on SPEA2 [85] and NSGA2 [20]. Katagiri et al. [40] introduced some improvements to the Delgado et al.

[21] method by introducing an elite preservation strategy, a self-adaptive mutation probability and preservation of local optimal solutions and their efficiency have been verified with benchmark time series data. For this reason, multi-objective evolutionary optimization of training and topology of RNN is adopted for optimizing an appropriate topology of an RNN with the capability of addressing our problem.

5.2 Initial Topology of the Recurrent Neural Network

The initial topology of the RNN to be optimized will be constructed with one input layer of n units, one hidden layer of m units and a single unit output layer. And a p number of consecutive previous input layers, called past input layers, are copied and kept and are considered as inputs to the hidden layer during each epoch. Similarly a q number of consecutive hidden layers, called feedback layers, are copied and kept and are considered as inputs to both the hidden layer itself and the output layer. As depicted in Figure 5.1, black arrow lines indicate the inputs to respective layers whereas dashed arrow lines indicate copying of layers to past input layers and feedback layers. $I_1, I_2, .., I_n$ are input layer neurons, $H_1, H_2, ..., H_m$ are hidden layer neurons and X_1, X_2, X_3 are inputs to the network (SNR, residual difference and change of wave length at a candidate point). Moreover, the upper shady area shows the p number of past input layers and the lower shady area shows the q number of feed-back layers.

With this RNN, if the output of the network is $Y(t)$ for a given set of inputs X at any given time t, then $Y(t)$ can be explicitly expressed as:

$$Y(t) = f\left(\sum_{a=1}^{m} w_{oa}^{H}(t).H_a(t) + \sum_{b=1}^{q}\sum_{c=1}^{m} w_{obc}^{F}(t).H_c(t-1)\right), \tag{5.1}$$

where

$$H_a(t) = f\left(\sum_{d=1}^{n} w_{ad}^{IH}(t).I_d(t) + \sum_{e=1}^{q}\sum_{g=1}^{m} w_{aeg}^{FH}(t).H_g(t-e) + \sum_{h=1}^{p}\sum_{i=1}^{n} w_{ahi}^{PH}(t).I_i(t-h)\right), \tag{5.2}$$

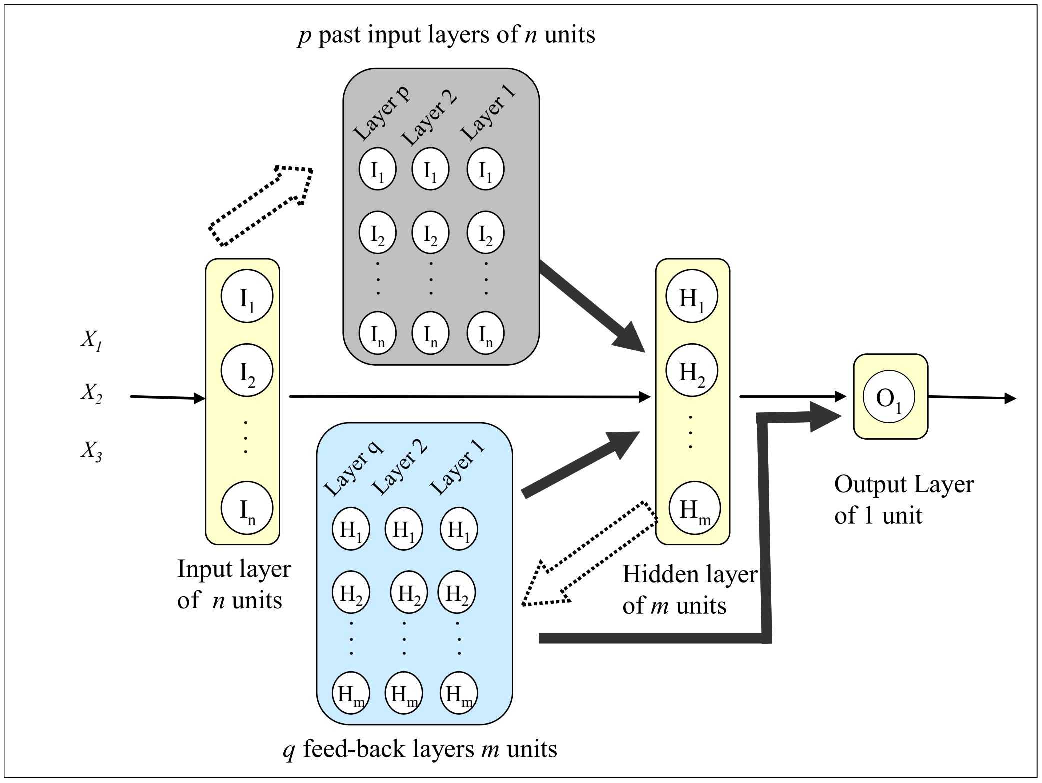


Figure 5.1: Initial topology of the recurrent neural network.

$$where \quad I_d(t) = f\left(\sum_{j=1}^{3} w_{dj}^{XI}(t).x_i\right) \tag{5.3}$$

$$and \quad f(x) = \frac{1}{1+e^{-x}}. \tag{5.4}$$

$H_g(t)$ and $I_i(t)$ in equaition 5.2 can also be expressed in similar pattern.

In equations 5.1 and 5.2, $H_a(t), a = 1, \ldots, m$, $H_c(t), c = 1, \ldots, m$ and $H_g(t), g = 1, \ldots, m$ are outputs from the hidden layer and feedback layers at time t. In equation 5.2 $I_d(t), d = 1, \ldots, n$ and $I_i(t), i = 1, \ldots, n$ are outputs from the input layer and past input layers at time t. w_{oa}^{H} is the weight associated with connection between a^{th} unit of the hidden layer and the output layer and w_{obc}^{F} is the weight associated with connections between c^{th} unit of the b^{th} feedback layer and the output layer. Similarly w_{ad}^{IH} is weight the weight associated with the connection between d^{th} unit of the input layer and the a^{th} unit of the hidden layer, w^{FH}_{aeg} is the weight associated with the connection between g^{th} unit of the e^{th} feedback layer and a^{th} unit of the hidden layer and w^{PH}_{ahi} is the weight associated with the connection between i^{th} unit of the h^{th} past input layer and a^{th} unit of the hidden layer. In equation 5.3 w^{XI}_{dj} is the weight associated with the connection between j^{th} input and the d^{th} unit of the input layer.

5.3 Multi-Objective Topology Optimization

With the above topology of an initial recurrent neural network and with a set of initial training data set, the objective functions for the multi-objective optimization algorithm can be formulated as follows.

- Minimize error: $\frac{1}{N}\sum_{i=1}^{N}(O_i(t) - Y_i(t))^2$
- Minimize number of neurons in the input layer
- Minimize number of neurons in the hidden layer
- Minimize number of past input layers
- Minimize number of feed-back layers
- Minimize Converged error value
- Minimize the difference between the error curve and standard back-propagation error curve

where $Y_i(t)$ is the network function mentioned in equation 5.1 and $O_i(t)$ is the target value at the t^{th} iteration step.

5.4 Optimization Algorithm

This optimization is a genetic algorithm based multi-objective evolutionary optimization algorithm, that optimizes the best topology of the recurrent neural network suited for the problem in the form of Figure 5.1. The recurrent neural network is also trained during evolution by back-propagation through time. A selected set of input data, in the form of $\mathbf{x}(x_1, x_2, x_3)$, is picked from actual voltage waveform data captured from flat panel display inspection machine utilizing a non-contact sensor. The genetic algorithm consists of N number of populations for the evolution process. Each member of the population is a chromosome where a chromosome consists of RNN with the initial topology mentioned in the above section. The mathematical representation of a chromosome in the algorithm is in floating point form which allows both integer and floating point parameters to be included. Figure 5.2 shows a chromosome used in this algorithm. The genetic algorithm based multi-objective evolutionary optimization algorithm is described in the following 8 steps.

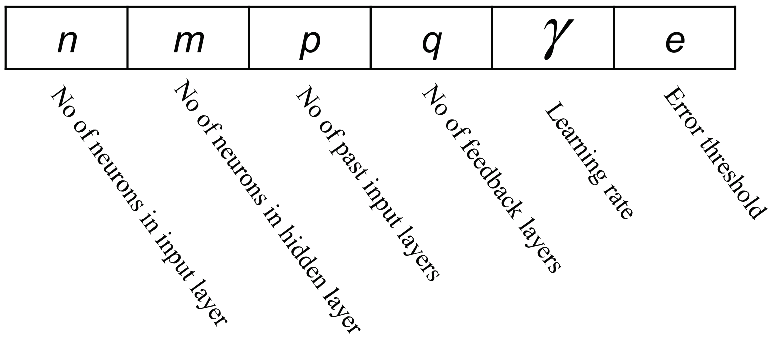


Figure 5.2: Format of the chromosome of the genetic algorithm.

Step 1: Initialization of population

Create a population of chromosomes of size N such that each chromosome contains an RNN with the topology as shown in Figure 5.1. All of the topology parameters in all RNNs of chromosomes are set randomly within their respective ranges. The selected set of training data is also assigned to the system initially.

Step 2: Evaluation of Solutions

Each RNN of each chromosome in the population is trained with back propagation through time (BPTT) algorithm using the given training data set. Since this is the most time consuming step, the patterns of error graphs of each RNN is checked frequently during training. If the error graph of any RNN goes out of shape from the expected convergence pattern, the training process of that particular RNN will be immediately terminated without continuing for the rest of the preset number of iterations. Such terminated chromosomes will be assigned zero marks of fitness level, which will be discarded and not allowed for cross overs or mutations.

Step 3: Measurement of fitness

A fitness value is assigned to each chromosome according to a pre-set marking scheme. The marking scheme assigns percentage of marks to each chromosome with the following criteria.

1. Converged error value.
2. Convergence pattern of error graph of its RNN (the better the convergence the higher the marks it earns) .
3. Number of neurons in input layer (the fewer the better)
4. Number of neurons in hidden layer (the fewer the better)
5. Number of past input layers (the fewer the better)
6. Number of feedback layers (the fewer the better)

Step 4: Checking for the pass mark limit

If the total fitness level of a chromosome exceeds the pass mark (a pre-set value, i.e. 90%), the evolution process is terminated, and the recurrent neural network associated with that chromosome will be considered as the optimized recurrent neural network. Since the network is already trained by back-propagation through time during this optimization process, it will be directly applied to the system.

Step 5: Discarding week chromosomes

If the fitness level of a chromosome is lower than a pre-set level, for example 20%, it is discarded from the populations for not to allow it to mutate or cross over with others members. The fitness level drops that low means that recurrent neural network is far below the expected level of the system and hence it should not be allowed to mutate or cross-over with other chromosomes for the next generation.

Step 6: Preservation

If the fitness level of a chromosome is bigger than a pre-set level but lower than pass mark, it is considered to be good enough to carry forward to the next generation without mutation or cross over, and that chromosome is flagged and kept. This step is to preserve considerably good chromosomes without any change in the current generation and allow them to improve in the next generation by mutation or crossing over with others.

Step 7: Mutation

Remaining chromosomes are allocated a mutation probability based on a roulette wheel based selection procedure. Accordingly the chromosomes that mutates will select mutation points on the chromosome randomly and performs the mutation.

Step 8: Cross Over

Remaining chromosomes are allocated a cross over probability based on a roulette wheel based selection procedure. Accordingly the chromosomes pairs that crosses over will select their cross over points randomly and performs cross over operation.

Step 9: Evolution

Create new chromosomes for discarded chromosomes in Step 2 and Step 5 and return to Step 2. For maintaining a constant number of population size, new members are created and added to the population. When create new chromosomes, the initial parameters are set as in the same criteria with the Step 1 where initial population is created. The population in this step is mixed with newly created members, mutated members from the previous generation, siblings from cross overs in previous generation and members who are preserved from previous generation. All of these newly fresh generation is passed again into Step 2 for evaluation.

5.5 Optimization Results

The multi-objective evolutionary optimization process, mentioned above, was performed with a population of size 40. The program has to be executed in a multi threaded environment with a visual interface indicating all the status of each chromosome. This process needs a huge system speed and capacity and as a result the number of population was needed to be limited to 40 with limitations on currently used PC. Since one chromosome means one recurrent neural network is associated with it, the size of population always matters the evolution time, as every single recurrent neural network must be trained parallel. We observed that the average fitness level of generations was gradually increasing during the evolution, as in the graph shown in the Figure 5.3, and after about 100 generations a chromosome with a fitness level over 90% was found. Though the graph had some uncertain fluctuations during initial few generations, as shown in the figure, it became stable and quite smooth in the end.

The topology of the RNN of optimized chromosome was consisting of one input layer with 2 units and one hidden layer with 6 units. The number of past input layers were optimized to 3, and the number of previous hidden layers were optimized to 1. Figure 5.4 shows the topology of the optimized recurrent neural network. The black arrows indicate inputs to a certain layer from a certain layer whereas big dashed arrows indicate the copying of a layer in time steps during back-propagation through

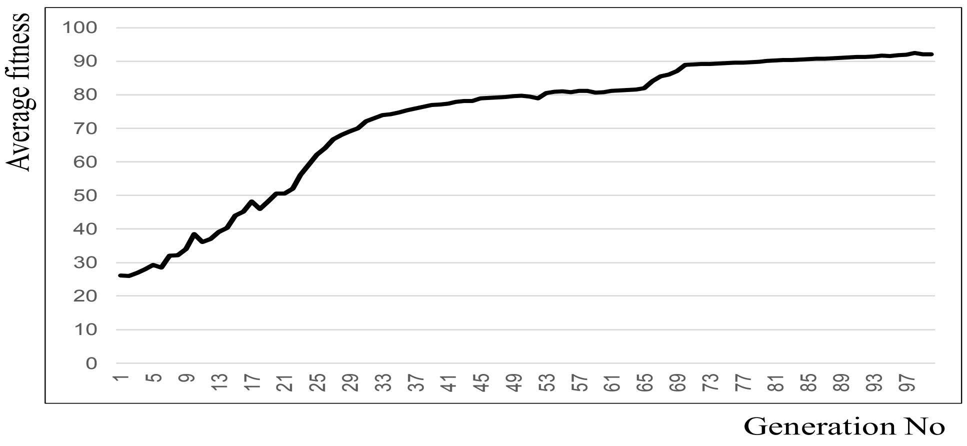


Figure 5.3: Change of average finesses during evolution.

time. With this RNN, the equations of explicit expressions; equations 5.1, 5.2 and 5.3 will be simplified to:

$$Y(t) = f\left(\sum_{l=1}^{6} w_{ol}^{H}(t).H_l(t) + \sum_{m=1}^{6} w_{om}^{F}(t).H_m(t-1)\right), \tag{5.5}$$

where

$$H_l(t) = f\left(\sum_{i=1}^{2} w_{li}^{IH}(t).I_i(t) + \sum_{j=1}^{6} w_{lj}^{FH}(t).H_j(t-1) + \sum_{r=1}^{3}\sum_{k=1}^{2} w_{lkr}^{PH}(t).I_k(t-r)\right), \tag{5.6}$$

$$where \qquad I_i(t) = f\left(\sum_{s=1}^{3} w_{is}^{XI}(t).X_s\right) \tag{5.7}$$

In equations 5.5 and 5.6, $H_l(t), l = 1, \ldots, 6$ are outputs from the hidden layer and $I_i(t), i = 1, 2$ are outputs from the input layer at time t. w_{ol}^{H} and w_{om}^{F} in equation 5.5 are weights associated with connections between l^{th} unit of the hidden layer and the output unit and between m^{th} unit of the feedback layer and the output layer respectively. Similarly w_{li}^{IH}, w_{lj}^{FH} and w_{lkr}^{PH} in equation 5.6 are weights associated with connections between the i^{th} unit in input layer and the l^{th} unit in hidden layer, between the j^{th} unit in feedback layer and the l^{th} unit in hidden layer, and between k^{th} unit in the r^{th} past input layer and the l^{th} unit in the hidden layer respectively.

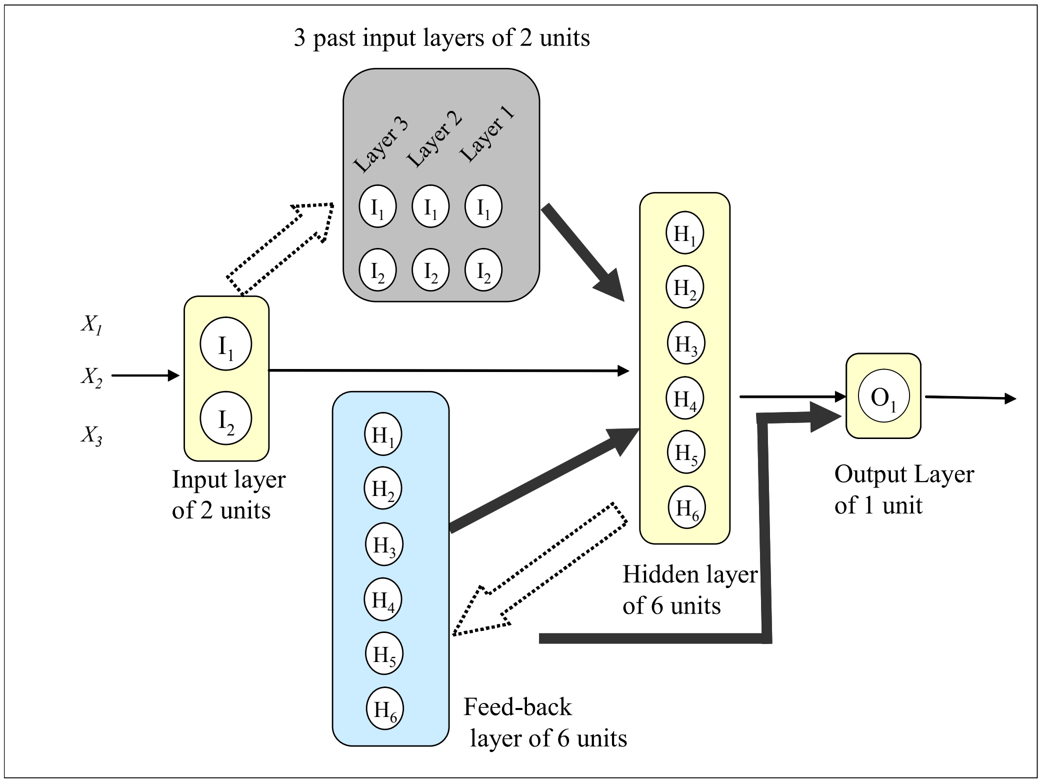


Figure 5.4: Optimized topology of the recurrent neural network.

In equation 5.7 w_{is}^{XI} is the weight associated with the connection between s^{th} input and the i^{th} unit in the input layer.

During the optimization, a high rate of discarded chromosomes was witnessed in early stages but was gradually reduced in the latter stages. The reason is that almost all chromosomes in initial stages are randomly created new chromosomes and do not possess any genetic feature brought from previous generation by crossing over or mutating. Therefore, the tenancy to discarded them in fitness test and even before the checking of error graph features during back-propagation through time is high.

However with the number of generations grows the number of randomly created new chromosomes gets smaller and the number of chromosomes possessing good genetic features gets bigger. Therefore the number of discarded chromosomes will get smaller in latter parts, which is a good indication that the algorithm gets going smoothly.

Furthermore, the number of preserved chromosomes in initial stages was zero and it started to appear a few numbers in the latter part. Once a preserved chromosome appears it will appear until the end, as the algorithm preserves them while allowing it to be crossed over and mutated. Therefore the possibility of one of those preserved chromosomes to cross the final pass mark is always high as they always keeps improving but never gets discarded.

5.6 Detection Results

The above optimized recurrent neural network was used for detection of NGs on various types waveform data, captured from many different machines in different locations, and produced very good results. Figures 5.5 and 5.6 show some detection results on 2 different waveforms captured from 2 different machines in different locations. Both of those voltage waveforms consist of different levels of random noises, external vibrations and baseline fluctuations on them but were able to detect using this method correctly.

The red circles in Figures 5.5 and 5.6 show real defect points while blue dashed lines are candidates and red dashed lines are detected defect points by the RNN among candidates. The upper graph of both figures 5.5 and 5.6 show detection in FNN based method while the lower graph of both figures shows detection in RNN based method. It shows in both cases that there are some defects points that cannot be detected in FNN method but were possible in RNN method.

Several sets of input data, selected from different types of voltage waveform data captured from different machines in different locations, were tested and the results were compared with both thresholding method and our previous feed-forward neural network based method.

Table 5.1 shows the rates of missed detections and false detections in existing

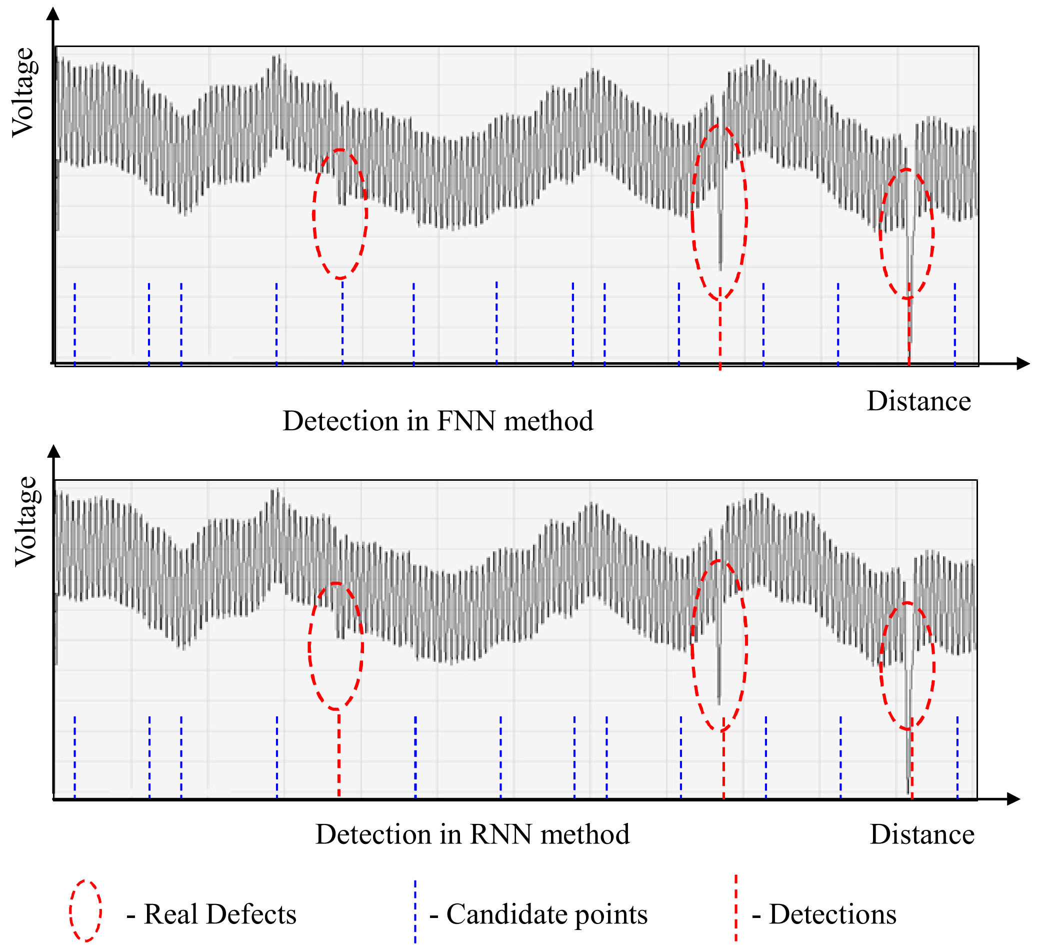


Figure 5.5: Detection results on a wave segment.

method, FNN method and RNN method over a selected data set. Existing rate of missed detections of about 25% has dropped to 9% in FNN method and further dropped to 5% in RNN method, whereas existing rate of false detections of about 28% has dropped to 16% in FNN method and further dropped to 7.5% in RNN method. Moreover, it was also realized that the huge manual workload, such as determining threshold values manually whenever a new machine or product is introduced, can be drastically dropped. The time taken for optimization of the topology and training of the network, which was around 20 minutes, can be negligible as it is once and for all comparative with huge time taken by operators in each time when

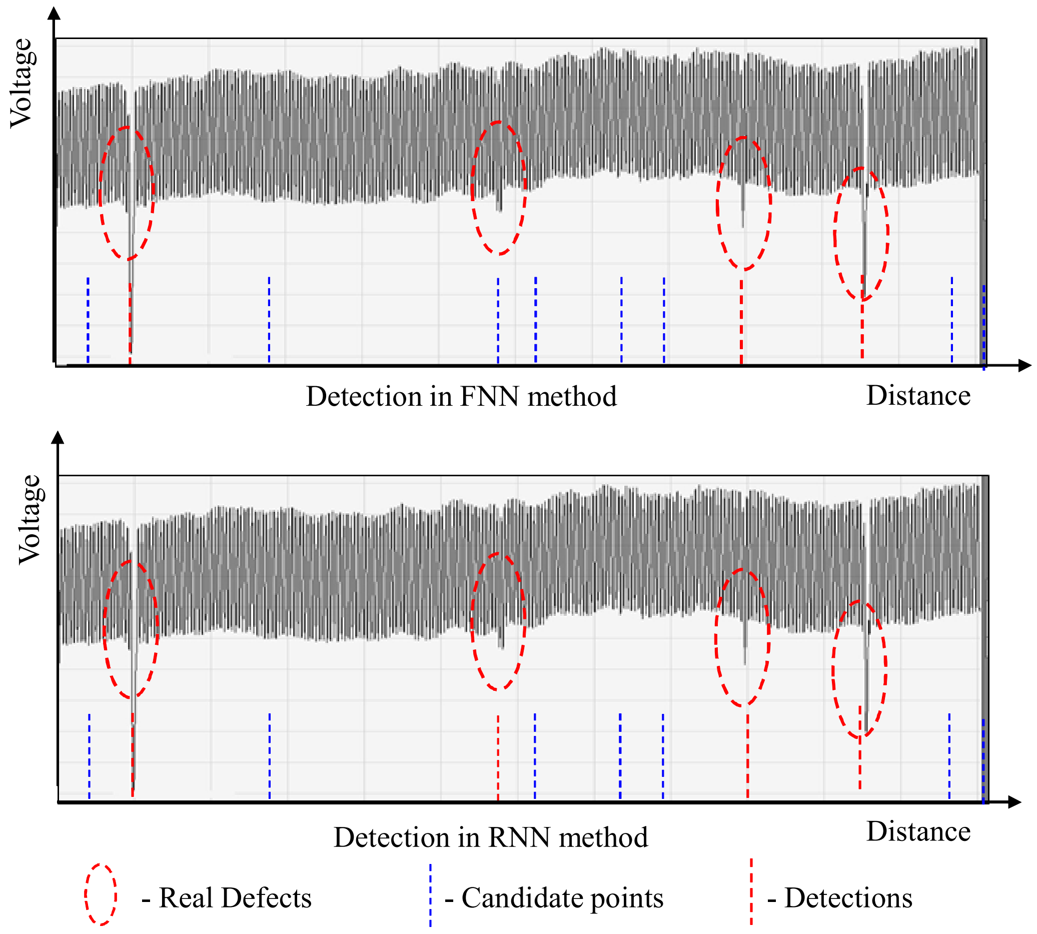


Figure 5.6: Detection results on another wave segment.

a product or machine is changed.

Among these two types of detections, missed detections and false detections, missed detections are the most crucial and never liked to be accepted by any manufacturer. If it allows assuming the judgment is correct it will continue to flow the production line until up to a finished product. However, it is highly unlikely to get 100% correct inspection and detection by any system, so that manufacturers always tend to have a manual inspection step sometime for all and sometime randomly. However by reducing the rate of missed detection drastically, that rate of manual inspections will also be reduced and the manufacturer's confidence will also increase.

Reducing the rate of false detection will also reduce the manual inspection time and the relation is always one to one since each and every panel that has been shown as defective has to be inspected manually.

Table 5.1: Comparison of missed detections and false detections between FNN method, RNN method and thresholding method.

Total defects		Missed detection			False detection		
		Threshold method	FNN method	RNN method	Threshold method	FNN method	RNN method
Data set 1	48	14 (29.1%)	6(12.5%)	3(6.25%)	13(27.0%)	8(16.6%)	5(10.4%)
Data set 2	60	16(26.5%)	5(8.3%)	3(5.0%)	18(30.0%)	9(15.0%)	4(6.6%)
Data set 3	50	11(22.0%)	4(8.0%)	2(4.0%)	14(28.0%)	8(16.0%)	3(6.0%)
Total	158	41(25.9%)	15(9.45%)	8(5.0%)	45(28.4%)	25(15.8%)	12(7.5%)

Chapter 6

Scan Path Optimization Using a Self-Organizing Map

The best known and most popular model of self-organizing map (SOM) is the topology preserving map proposed by Teuvo Kohonen [42, 43]. SOMs, also known as Kohonen networks, are an embodiment of some of the ideas developed by Rosenblatt [68], von der Malsburg [53], and other researchers. If an input space is to be processed by a neural network, the first issue of importance is the structure of this space. A neural network with real inputs computes a function f defined from an input space A to an output space B. The region where f is defined can be covered by a Kohonen network in such a way that when, for example, an input vector a_1 is selected from the region A, only one unit in the network fires. Such a tiling in which input space is classified in sub regions is also called a chart or a map of input space. Kohonen networks learn to create maps of the input space in a self-organizing way.

Kohonen network model works with elements not very different from the ones used by other researchers. More relevant is the definition of the neighborhood of a computing unit. Kohonen networks are arrangements of computing nodes in single or multi-dimensional lattices (Figure 6.1). During learning, the weights of computing units and their neighbors are updated. The objective of such a learning approach is that neighboring units learn to react to closely related signals.

Kohonen learning uses a neighborhood function ϕ, whose value $\phi(i, k)$ represents the strength of the coupling between unit i and unit k during the training process.

A simple choice is defining $\phi(i,k) = 1$ for all units i in a neighborhood of radius r of unit k and $\phi(i,k) = 0$ for all other units [42, 43].

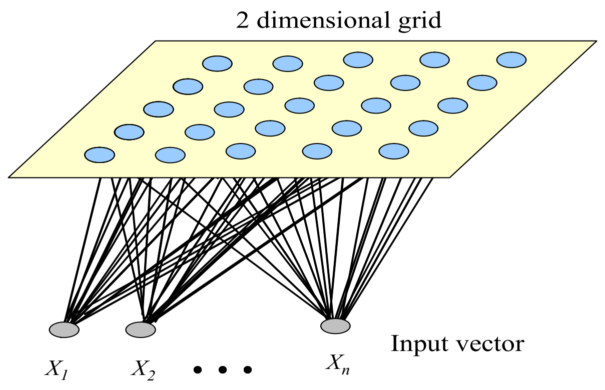


Figure 6.1: A typical self-organizing map.

6.1 Scan Path of Defective TFT Lines

In the non-contact FPD inspection method proposed by Hamori et al.[29, 30, 31, 32], the defective TFT lines (or NG lines), which are detected using thresholding or feed-forward neural network or recurrent neural network methods (proposed in chapters 4 and 5), have to be scanned again using a micro camera based sensor (called NG sensor) to determine exact locations of NGs (open circuits or short circuits) on defective lines. This process is necessary for operators in order to determine if those NGs are to be repaired or not. Once defective lines are observed by the non-contact sensor, this line scanning by the NG sensor must obviously be a shortest possible path. However, currently used motion path of the NG sensor is a simple top-down unidirectional method which is heavily time consuming particularly on larger glass panels such as 3m x 3m panels. It may be negligible for a few number of lines or small sizes of mother glasses. However due to current trend of high demand of bigger sizes of flat panels it is no longer negligible. Moreover with the emergence of finer pitch patterns for 4K and 8K TVs, the number of lines for a panel becomes bigger and hence the possibility of more defective lines on panels increases. Therefore, it is necessary to find a more realistic and intelligent approach of the possible shortest path of NG scanning. Figure 6.2 depicts an existing top-down unidirectional motion path of NG sensor, where red dashed lines indicate detected defective TFT lines and blue arrow path depicts the top-down unidirectional motion path. As seen in the figure it is obvious that this top-down unidirectional scan path is a huge time waisting particularly for large panels such as $3m \times 3m$.

Therefore optimizing the detection of defective lines by feed-forward neural network and then improving with a multi-objective topology optimized recurrent neural network alone does not finish the task of defect detection on flat panel displays. The second stage where those detected defective line scanning by a NG scanner must also be optimized to overcome the huge time consuming problem in the current this primitive scanning method of top-down unidirectional method.

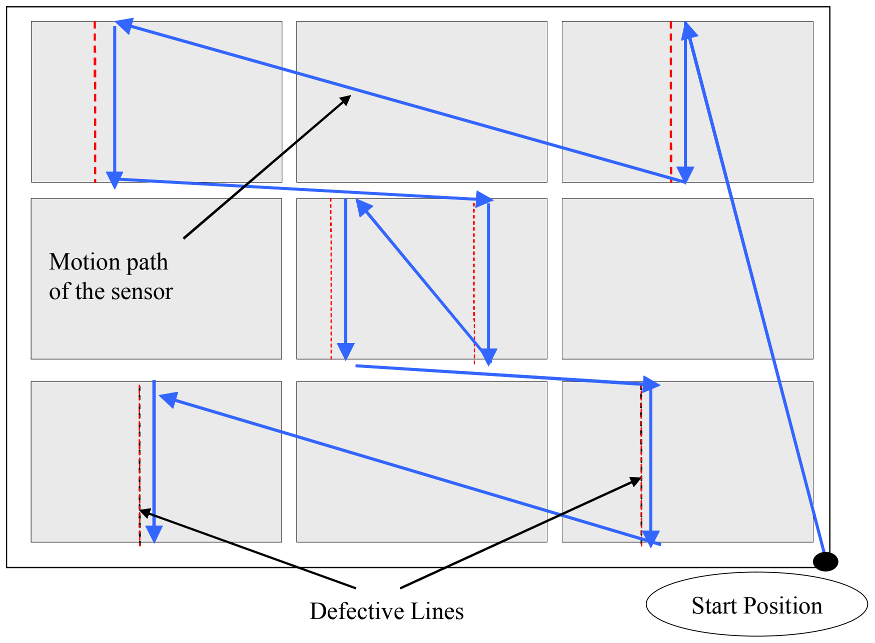


Figure 6.2: Existing top down unidirectional line scan path on a 3x3 panel lay out.

6.2 Greedy Approach

A greedy algorithm is a mathematical process that looks for simple, easy-to-implement solutions to complex, multi-step problems by deciding which next step will provide the most obvious benefit. Such algorithms are called greedy because while the optimal solution to each smaller instance will provide an immediate output, the algorithm doesn't consider the larger problem as a whole. Once a decision has been made, it is never reconsidered. At each step the value for one decision variable is assigned by making the best available decision. A heuristic is needed for making the decision at each step: what is the best next?. Therefore we cannot expect the greedy algorithm to obtain the overall optimum solution.

Nearest neighbor search is one such algorithm that can be applied to this kind of problems. Nearest neighbor search, also known as proximity search or closest point search, is an optimization problem for finding closest points. It is good for a few number of inputs (in our case defective lines) but when the size of input space grows the algorithm time increases exponentially. Since in each step it searches the next best step it obviously involves a huge time factor.

Practically, after testing the greedy nearest neighbor search, it was realized that our system cannot afford to loose such a huge time. It was well within the allowed time frame only for less than 7 lines. However the number defective lines can always be much more than that and therefore it was understood that greedy approaches are not appropriate for this system. Under these circumstance, again a self learning neural network approach was considered. How the problem of finding a closest scan path is approximated to a asymmetric traveling salesman problem and how it is solved by a self organization map or Kohonen network is described in following sections.

6.3 Approximation to a TSP Solver

The existing top-down unidirectional line scan method (Figure 6.2) is heavily time consuming for larger sized glass panels as well as for panels with higher number of defective TFT lines. This problem can be described as finding the shortest path of visiting each defective TFT line once and only once. Furthermore, without loss of generality, if a defective line is considered as a pair of points (two tips of the line) the path of the sensor would become a path connecting a set of pairs of points. However, once an either point on a defective line (a tip) is reached the next point must necessarily be the other point (other tip) of the same defective line. In other words there is a precedence of reaching the next point when one tip of a defective line is reached. Hence the problem of finding the shortest path within a set of points is restricted by precedence of those pairs of points. Therefore, this problem can be approximated to an asymmetric traveling salesman problem (TSP) with a precedence constraint [56].

The traveling salesman problem is one of the most intensively studied problems in the computational mathematics, and is commonly encountered in the areas of manufacturing, planning, scheduling, and transportation logistics, that involves the determination of a tour, which starts from a base city and then visits all other given cities once and only once and terminates at the base city while minimizing the total distance traveled. The goal in solving a TSP is to find the minimum cost tour, or the optimal tour, covering a n number of cities. A tour of vertices of a graph which visits each vertex (repeating no edge) once and only once is known as *Hamiltonian cycle*. No general method of solution is known, and the problem is *NP-hard*. Figure 6.3 shows a typical traveling salesman problem. TSP has been discussed extensively in the literature [10, 13, 18, 25]. The presence of precedence constraints within the asymmetric TSP framework is encountered quite often in practice. Examples include: disassembly optimization and scheduling of wafers/ ICs on automated testing equipments in a semiconductor manufacturing facility; among others.

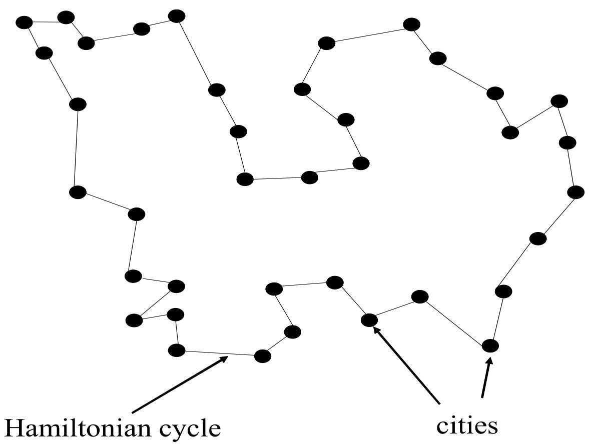


Figure 6.3: Solution of a typical traveling salesman problem.

Solving TSPs has many approaches in the literature and neural network based approaches are numerous and a promising survey of neural network applications for TSPs is given in the work done by Maire and Mladenov [52]. Takahashi [76], Mandziuk [54] and Feng and Douligeris [26] proposed Hopfield network based approaches for TSP solvers. Ghaziri and Osman [27] proposed a new heuristic based on a self-organizing map for TSP with back-hauls. Siqueira et al.[74] have proposed a TSP solver by using a recurrent neural network while a genetic algorithm based approach was proposed by Moon et al. [61]. And there is a recent work done by Zhong et al. [84] which proposed a dynamic Tabu artificial Bee colony algorithm for multiple TSPs with precedence constraints.

However determining what approach must be taken for a particular problem depends on many factors of the behavior of precedence involved, the size of input space, etc. Therefore in this work a combination of modified self-organizing map, a 2-Opt algorithm and a repair algorithm is proposed to find the shortest path of NG sensor over detected defective TFT lines on flat panel displays.

6.4 Scan Path Optimization by a Modified Self-Organizing Map

As described in the above section our problem of finding a shortest possible scan path of the NG sensor through NG lines on flat panel displays can be formulated as an asymmetric traveling salesman problem with precedence constraints. Our focus in this chapter is to solve this problem using a modified self-organizing map together with a 2-Opt algorithm and a repair algorithm. Self-organizing map based TSP solvers have also been discussed in the literature. Budinich [14, 15] presented a self-organizing map very early in 1996 instead of traditional neural networks which produced approximate solutions for TSP. Schabauer [73] proposed a parallelized SOM cluster architecture to solve very large TSP. Incorporating efficient initialization algorithm and a parameter adaptation algorithm to achieve faster convergence for non-constraint TSP was proposed by Bai et al.[10] and Brocki and Korzinek [13] also discussed an SOM based method.

6.4.1 Formulation of the SOM

As described in above sections, the shortest path of the NG sensor through defective TFT lines can be approximated to an asymmetric traveling salesman problem with precedence constraints. And our focus is to find a solution using a self-organizing map. Though a single line is considered as a pair of points, that pair of points is again considered as a single point (the middle point) in input space to the SOM. In other words a visit of the sensor to a defective line is considered as a visit to the middle of the line, and therefore the number of points in input space is equal to the number of lines.

Initially, a random map of neurons (nodes) is created with the same number of inputs and, without loss of generality, the random neurons are placed on a circular chain with same distance between them as shown in Figure 6.4. When mapping input points to nodes, the distance from a point to a node is taken as the synaptic

weight of the node (neuron). There are two approaches of mapping input points to nodes, namely Winner Takes All (WTA) and Winner Takes Most (WTM), and here WTM approach is adopted since it gives a chance not only to the best matched node but also to neighboring nodes to be adjusted. This is more appropriate particularly in cases with larger sizes of input spaces.
In the learning process of the SOM, the synaptic weight of the best matching node (BMN) for an input will be adjusted as in the following equation and as shown in Figure 6.5.

$$W_i \longrightarrow W_i - \lambda W_i \tag{6.1}$$

where W_i is the synaptic weight of i^{th} node and $\lambda(0 < \lambda < 1)$ is the learning rate. Synaptic weights of the neighboring nodes of i^{th} node will also be adjusted as:

$$W_{ij} \longrightarrow W_{ij} - \alpha\lambda W_{ij} \tag{6.2}$$

where W_{ij} is the synaptic weight of j^{th} neighboring node of the i^{th} node and $\alpha(0 < \alpha < 1)$ is a neighborhood function. The learning rate is decayed linearly during each iteration such as:

$$\lambda = \lambda_0 \frac{1-r}{N} \tag{6.3}$$

where λ_0 is the initial learning rate, r is the current iteration number and N is the total number of iterations. The neighborhood function (update length) is also decayed linearly in the learning process as:

$$\gamma_{r+1} = \gamma_r - \frac{\gamma_r}{0.5r} \tag{6.4}$$

Where r is the iteration number and γ_r is the update length in the r^{th} iteration.

6.4.2 SOM Learning Algorithm

The learning of the SOM is performed, adhering to above mentioned protocols of change of synaptic weights, for a pre-determined n number of iterations as follows:

1. Create a set of nodes (the number of nodes is equal to the number of input points) and place them randomly on a circular chain with same distance between them. Initial positioning of nodes on a circular chain is more general than keeping them randomly scattered or on any other format since the input patterns are on the surface of a rectangular FPD panel lay out (Figure 6.4).

2. Take one input point randomly and find the BMN according to their synaptic weights, i.e. the nearest node (Figure 6.5).

3. Update synaptic weight of BMN as in Equation (6.1). The synaptic weight is the distance between the node and the input point, therefore, it actually moves the node towards the input point as shown in Figure 6.5.

4. Find the neighboring nodes to the BMN and update their weights according to Equation (6.2). Again the neighboring nodes are also moved towards the input point.

5. Repeat the step (2) to step (4) for all remaining input points.

6. Adjust the learning rate according to the Equation (6.3). In Equation 6.3 the learning rate λ decays linearly in each step with respect to the initial learning rate λ_0, therefore the amount or the distance of moving nodes towards input points are getting smaller in each iteration.

7. Adjust the neighborhood function according to Equation (6.4). By adjusting neighborhood function the number of neighboring nodes to be updated are getting decayed with iterations. Therefore only the BMN is getting moved during latter parts of iterations.

8. Repeat step 2 to step 7 for N number of iterations.

Figure 6.5 shows how a synaptic weight of a node is updated during training. Black dots are original positions of the input points whereas orange color dots are nodes. Yellow color dots indicate weight adjustment (distance) of neighboring nodes for one input point (start point) in one step of the training.

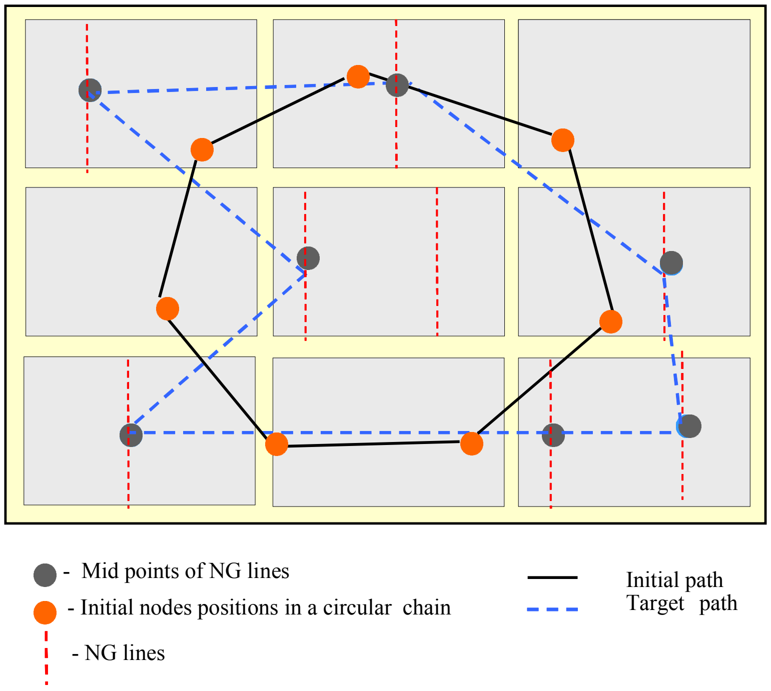


Figure 6.4: Initial positions of nodes.

6.4.3 Node Assignment Algorithm

After the learning algorithm of the SOM is over it can be observed that nodes are not yet perfectly mapped onto input points as shown in Figure 6.6. This may lead to confusions due to two reasons, one is when multiple nodes are closer to a single point and the other is when some nodes are isolated. Therefore we follow three steps when assigning nodes to points as:

1. Assign the nearest node for each point. When assigning nodes to points in this way, multiple nodes can be assigned to a single point and there may be some isolated points without any node assignment (Figure 6.7).
2. In multiple assignments, delete all nodes other than the nearest node.
3. Create a new node per each node deleted in step 2 and assigned them to points where there are no assignments.

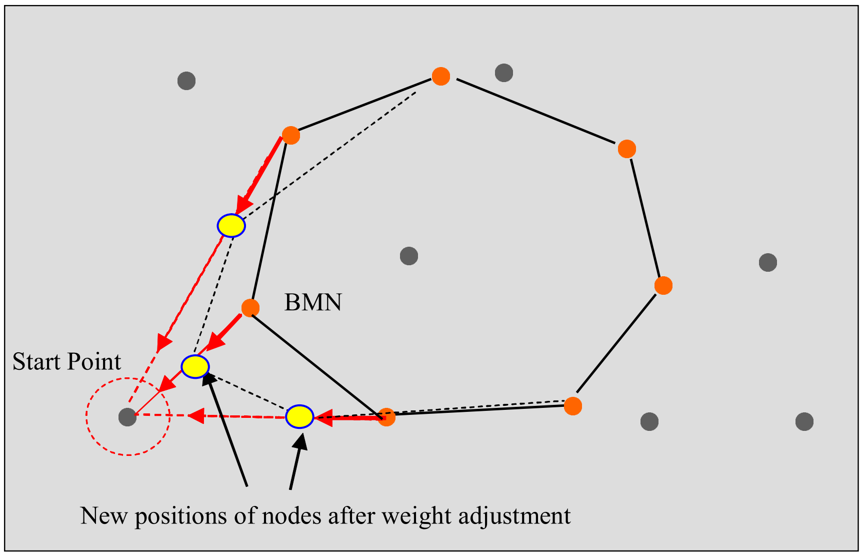


Figure 6.5: Updating synaptic weights of a node.

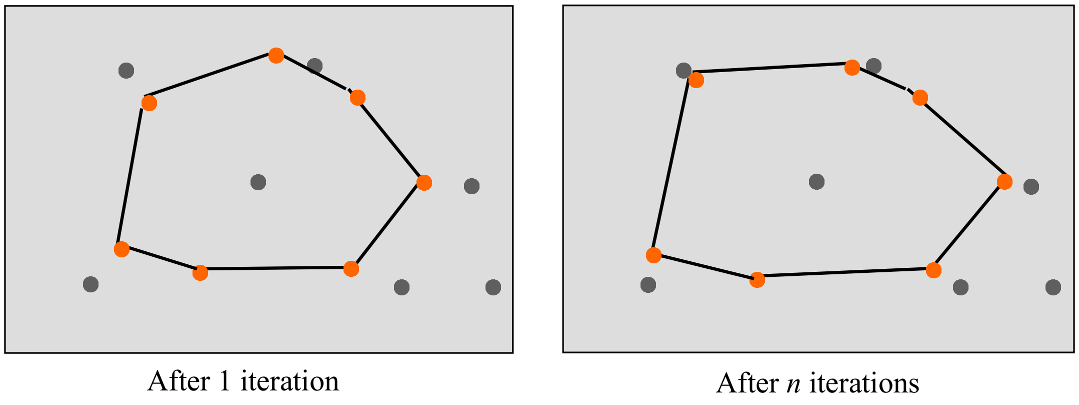


Figure 6.6: Node mapping after 1 and n iterations.

6.4.4 Using 2-Opt Algorithm

The points, assigned by nodes, are replaced by their original pairs of points (the two tips of the defect lines). However the path distance, after replacing by original pairs of points, may have huge differences on the way on which direction the line of one point connects to the line of next one. In order to overcome this problem

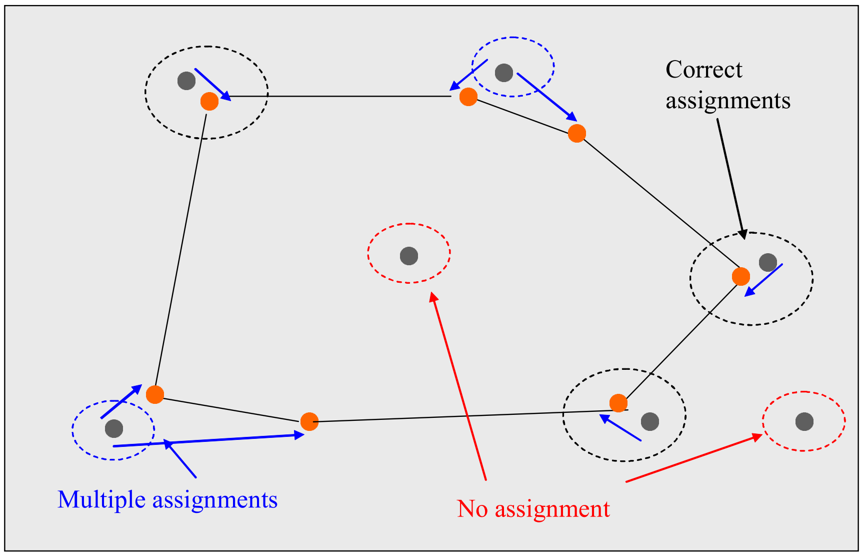


Figure 6.7: Assignment of nodes to points.

and also to avoid any possible loop (Figure 6.8) within the path, a greedy 2-opt algorithm for each node was employed when restoring lines.

In the area of optimization theory, 2-opt is a simple local search algorithm first proposed by Croes in 1958 for solving the traveling salesman problem. The main idea behind it is to take a route that crosses over itself and reorder it so that it does not. 2-opt algorithm consists of eliminating two edges and reconnecting the two resulting paths in a different way to obtain a new tour. There is only one way to reconnect the paths that yield a different tour. Among all pairs of edges whose 2-opt exchange decreases the length we choose the pair that gives the shortest tour. This procedure is then iterated until no such pair of edges is found. The pairwise exchange or 2-opt technique involves iteratively removing two edges and replacing these with two different edges that reconnect the fragments created by edge removal into a new and shorter tour for a traveling salesman problem or graph traversing problem. The resulting tour is called 2-optimal.

3-opt (3-optimal) or 4-opt (4-optimal) may be better but only the 2-Opt is adopted in this work due to the time factor. Blazinskas and Misevicius [12] have

studied 2-opt, 3-opt, 4-opt and their combination in TSP. In Figure 6.8 $ABCD$ is a loop and 2-Opt algorithm searches and replace them as follows;
If $path\ ABCD > path\ ACBD$ then Set the path to $ACBD$

Figure 6.9 shows the final line restored path created by this SOM method for the input lines in Figure 6.4.

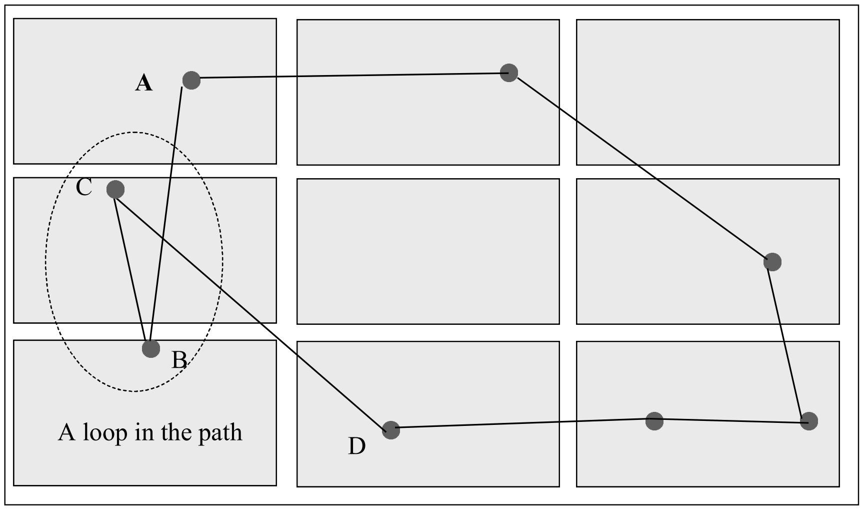


Figure 6.8: Loops on the optimized path.

6.5 Simulation Results

Numerous patterns of NG lines on different layouts of flat panel displays were simulated using simulator software. It was observed that this method improved the distance of line scan sensor by around 40%. The rate of improvement ranged from 20% to 60% and the algorithm time was also greatly impressive as it was below 10 ms even for 100 lines. Figures 6.10 to 6.14 are some selected randomly simulated results for a 3x3 panel layout. The path distances showed in figures are the distances by GUI coordinates. The path depicted in blue color is the new SOM based path

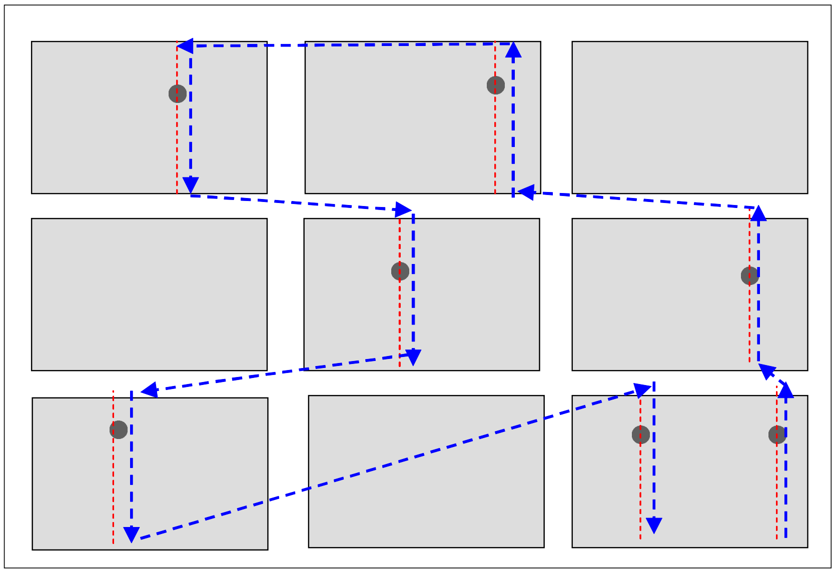


Figure 6.9: Final scan path optimized by the SOM.

and the path in black dashed lines is the existing top down unidirectional method in each figure. The numbers indicate the sequence on the path.

Table 6.1 shows a comparison of performance between this SOM based method and the existing top-down unidirectional method for some selected number of lines. Furthermore these figures are average distances traversed calculated of 10 random locations of lines using our simulator software.

Practically, when applying this method, the minimum number lines for the 2-Opt algorithm must be 4 and the minimum number of lines for the SOM learning must be 5 for a minimum of 2 neighbor learning. Therefore it was decided that this method is applied only for the cases where there are more that 5 number of lines, and otherwise the greedy nearest neighbor search method mentioned in section 6.2 was used, and the search time was within the allowed time for such a small number of defective lines. Moreover the simulation examples like in Figure 6.14, it is easy to understand even by human eye that the existing top-down unidirectional scan path is a careless time waste and it shows a huge improvement by our method which is nearly 60%. In another case, like shown in Figure 6.11, though it's improvement using our method is 30% it is still very much better from manufactures stand point. And overall the average improvement of our SOM based method, which is nearly 40%, against the existing unidirectional method, was a tremendous achievement and very much within the acceptance ratio by any manufacturer. Furthermore, The algorithm time does not matter if the inputs are from simulator data or real factory data as the path optimization is purely a system task dependent only on numerical input array (coordinates and orientation) of defective lines. Therefore factory environments or locations never matter which is also another factor view by manufactures stand point.

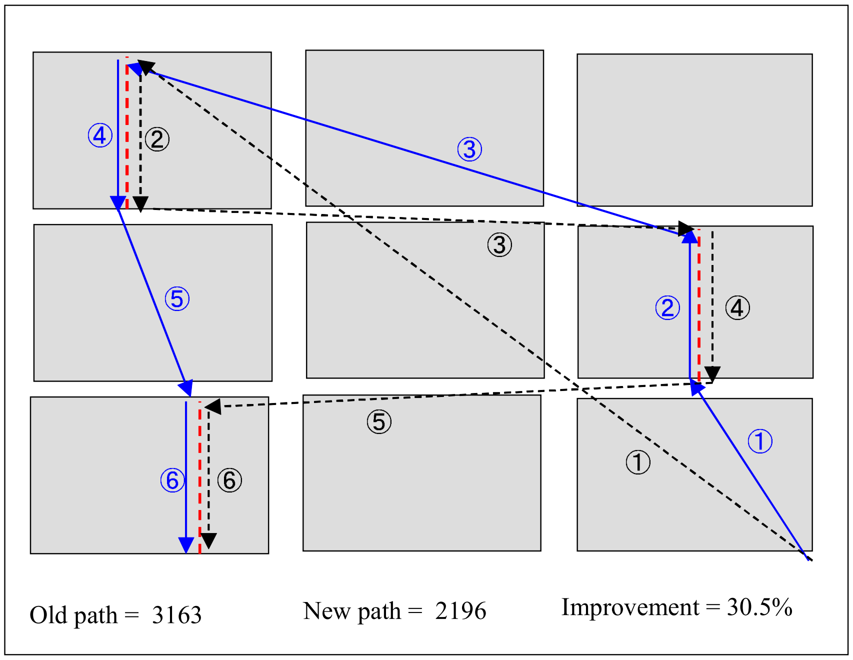


Figure 6.10: Optimized scan path for 3 random NG lines.

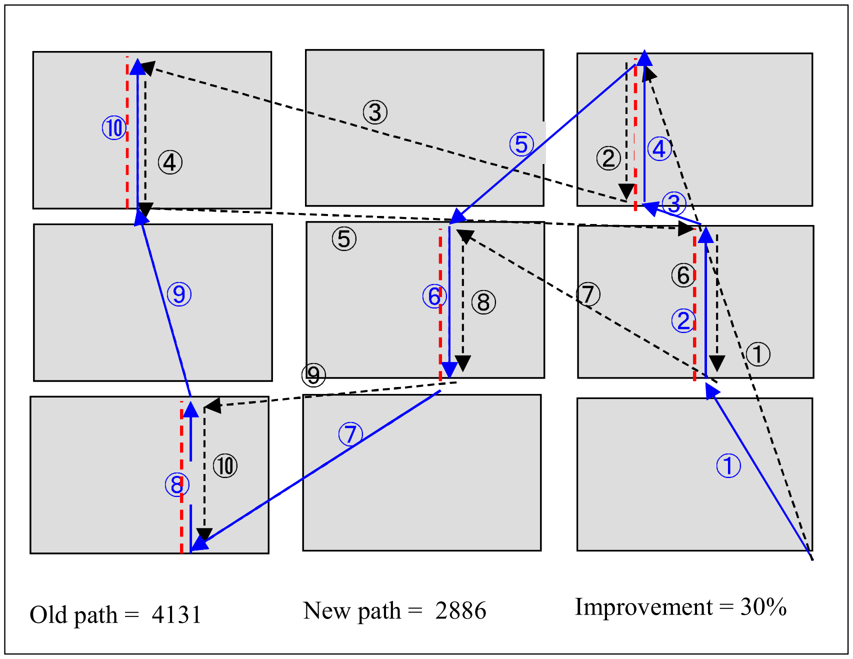


Figure 6.11: Optimized scan path for 5 random NG lines.

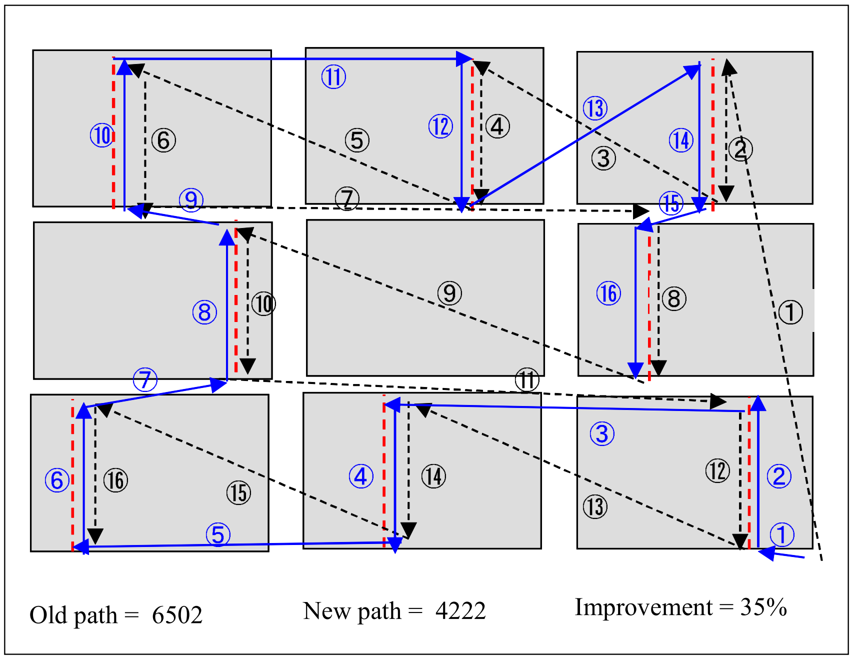


Figure 6.12: Optimized scan path for 8 random NG lines.

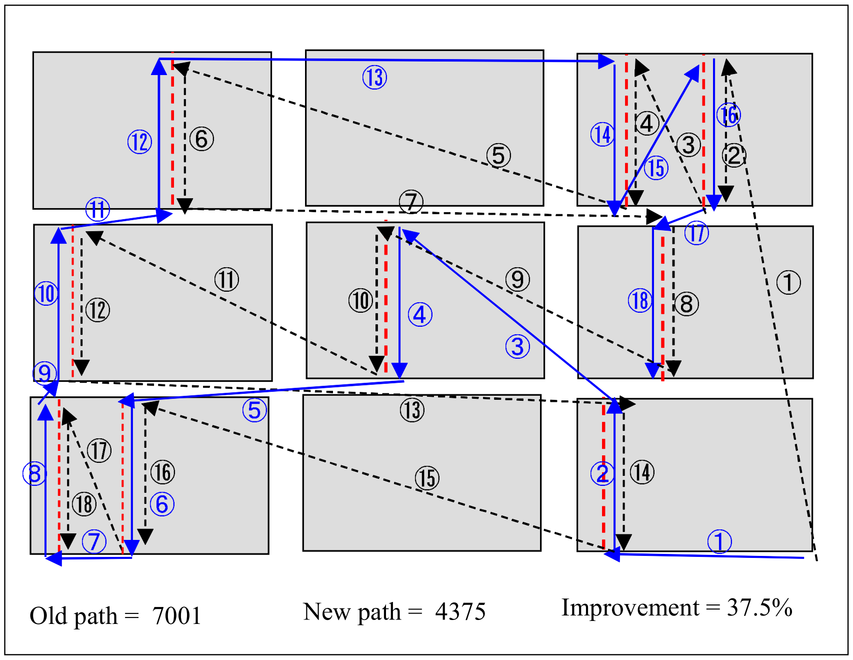


Figure 6.13: Optimized scan path for 9 random NG lines.

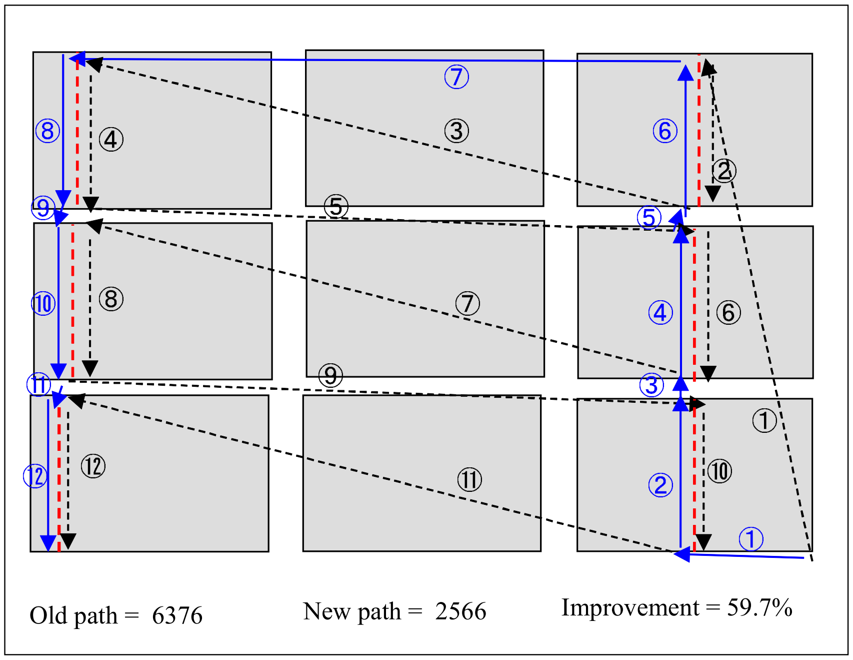


Figure 6.14: Optimized scan path for 6 random NG lines.

Table 6.1: Comparison of path distances between existing method and SOM based method.

No of lines	Distance traversed in existing method	Distance traversed in SOM method	Improvement (%)
	2593	1602	38.18
6	2881	1639	43.11
7	3105	1990	35.91
8	4000	2236	44.10
9	4214	2406	42.09
10	4808	2869	40.33
15	6401	3842	39.98
20	8088	4813	40.49
25	9468	5988	39.76
30	10784	6578	39.00
50	17098	9866	42.3
Average Improvement			40.28

Chapter 7

Conclusion

In this dissertation defects inspection process of flat panel display fabrication through neural networks was presented. First chapter is devoted to an introduction and the explanations of the flat panel display industry, its prospects and defect inspection process of thin film transistor (TFT) array of flat panel displays and the second chapter presented basic definitions and methods in neural networks. Chapter three presented a brief explanation of non-contact defect inspection of flat panel displays together with currently used thresholding method .

In chapter 4, the problem was identified as a largely data driven and highly non-linear classification problem of waveform data captured through a non-contact sensor by scanning over a TFT array of flat panel display. Then the problem was attempted to solve by using a feed-forward neural network. The approach was shown much better than the existing thresholding method through experimental results as the rates of missed detections and false detections were dropped below 15% from existing rate of around 25%.

In chapter 5, for the purpose of addressing the drawbacks that were encountered in feed-forward neural network method, such as difficulty of determination of a correct topology of the network and vulnerability to massively noisy input data, a multi-objective evolutionary optimized recurrent neural network was adopted to the problem. A genetic algorithm based multi-objective evolutionary optimization algorithm evolved the recurrent network and trained. Each chromosome of a generation consists of an initially assumed architecture of a recurrent neural network, and, during the evolution process each network of each chromosome is trained by back-propagation through time in every generation. The experimental results showed that the method is superior than both existing thresholding method and previous feed-forward neural network based method. The rate of missed detections dropped below 5% and the rate of false detections was dropped below 8%. Furthermore this method drastically dropped the work load of operators for manually determining thresholding values for each and every time when a machine change, product change, location change or recipe change

In chapter 6, the shortest scan path of the NG sensor over defective TFT lines was focused and a method, combining a modified self-organizing feature map (SOM) and 2-Opt algorithm, was adopted. First the problem was approximated to a asymmetric traveling salesman problem with precedence constraints. The SOM based TSP solver, together with a repair algorithm for node assigning and 2-Opt algorithm for avoiding any possible loops on the final path, produced highly acceptable results through the simulator. The average distance of the path traversed by this method was improved by around 45% against existing top-down unidirectional method. Moreover, the algorithm time did not exceed 10 ms even for 100 defective lines, which was very much impressive and acceptable from manufacturers perspective.

By the results of several experiments given in each chapter, the proposed optimized recurrent neural network based defect inspection method of TFT lines on flat panel displays and the self-organizing map based NG line scan method were found to be much efficient and effective for defect inspection process of flat panel displays.

As some future directions of this research work, firstly the multi-objective optimization algorithm of the recurrent neural network is to be improved for faster evolution by applying a better learning coefficient in back-propagation through time. Furthermore, the path optimization algorithm is to be improved by avoiding the repair algorithm, where repair algorithm can be failed if two nodes are overlapped or multiple nodes are positioned very closely during training.

References

[1] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, Defects Detection on TFT lines of Flat Panels using a Feed Forward Neural Network, *Artificial Intelligence Research*, Vol. 2, No. 4, pp. 1-12, 2013.

[2] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, Defects Detection on TFT lines of Flat Panel Displays using an Evolutionary Optimized Recurrent Neural Network, *American Journal of Operations Research*, Vol. 4, No. 3, pp. 113-123, May 2014.

[3] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, A Neural Network Approach for Non-Contact Defects Inspection of Flat Panel Displays, *17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems (KES 2013)*, Kita Kyushu, Japan, pp. 28-38, September 2013.

[4] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, A Multi-objective Evolutionary Optimized Recurrent Neural Network for Defects Detection on Flat Panel Displays, *11th International Conference on Modeling Decisions for Artificial Intelligence (MDAI 2014)*, Tokyo, Japan, pp. 170-181, October 2014.

[5] H. A. Abeysundara, H. Hamori, T. Matsui and M. Sakawa, Path Optimization for Line Scanning on Flat Panel Displays using a Self Organizing Map, *Computational Research*, Vol. 2, No. 4, pp. 63-68, 2014.

[6] J. Alam S.M. and H. Guoqing, Nonuniform Defect Detection of Cell Phone TFT-LCD Displays, *Indonesian Journal of Electrical Engineering*, Vol. 12, No. 8, pp. 6026-6046, 2014.

[7] J. H. Ang, C.K. Goh, E.J. Teoh and A.A. Mamum, Multi-Objective Evolution-

ary Recurrent Neural Network for System Identification, *Proceedings of IEEE Congress on Evolutionary Computations*, pp. 1586-1592, 2007.

[8] G. Auda and M. Kamel, Modular Neural Networks: A Survey, *International Journal of Neural Systems*, Vol. 9, No. 2, pp. 129-151, April 1999.

[9] O. Awodele and O. Jegede, Neural Networks and Its Application in Engineering, *Proceedings of Informing Science & IT Education Conference*, pp. 84-95, 2009.

[10] Y. Bai, W. Zhang and Z. Jin, A new self-organizing maps strategy for solving the traveling salesman problem, *Journal of Chaos, Solitons & Fractals*, Vol. 28, No. 4, pp. 1082-1089, May 2006.

[11] J. K. Basu, D. Bhattacharyya and T. H.Kim, Use of Artificial Neural Network in Pattern Recognition, *International Journal of Software Engineering and its Applications*, Vol. 4, No. 2,pp. 23-33, 2010.

[12] A. Blazinskas and A. Misevicius, Combining 2-opt, 3-Opt and 4-Opt with K-Swap-Kick-Perturbations for the Traveling Salesman Problem, *17th International Conference on Information and Software Technologies*, Kaunas, Lithuania, April 2011.

[13] L Brocki and D. Korzinek, Kohonen Self-Organizing Map for the Traveling Salesperson Problem, *International Journal of Recent Advances in Machatronics*, pp.116-119, 2007.

[14] M. Budinich, A Self-Organizing Neural Network for the Traveling Salesman Problem that is Competitive with Simulated Annealing, *Neural Computation*, Vol. 8, No. 2, pp. 1-8, 1996.

[15] M. Budinich, A Neural Network for the Traveling Salesman Problem with Well Behaved Energy Function, *Mathematics of Neural Networks, Operations research/ Computer Science Interfaces Series*, Vol. 8, pp. 134-139, 1997.

[16] T. L. Burrows and M. Niranjan, The Use of Recurrent Neural Networks for Classification, *Proceedings of the IEEE Workshop on Neural Networks for Sig-*

nal Processing pp. 117-125, 1994

[17] B. Chambayil, R. Singla and R. Jha, EEG Eye Blink Classification Using Neural Network, *Proceedings of the World Congress on Engineering*, Vol. 1, pp. 63-66, 2010.

[18] J. Cirasella, D. S. Johnson, L. A. McGeoch and W. Zhang, The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests, *Lecture Notes in Computer Science*, Vol. 2153, pp. 32-59, 2001.

[19] J. T. Connor and R. D. Martin, Recurrent Neural Networks and Robust Time Series Prediction, *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 240-254, March 1994.

[20] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182-197, 2002.

[21] M. Delgado, M. P. Cuellar and M. C. Pegalajar, Multiobjective Hybrid Optimization and Training of Recurrent Neural Networks, *IEEE Transactions on Systems, Man and Cybernetics: Part B Cybernetics*, Vol. 38, No. 2, pp. 381-403, 2008.

[22] J. Dolinsky and H. Takagi, RNN with Recurrent Output Layer for Learning of Naturalness, *Neural Information Processing: Letters and Reviews*, Vol. 12, No. 1-3, pp. 31-42, 2008.

[23] C. Du, J. Mou, L. Martua, S. Liu, B. Chen and F. L. Lewis, Time-domain Feature Extraction and Neural Network Identification of Structure crack based on surface-mounted active sensing network, *Fifth International Conference on Sensor Technologies and Applications*, pp. 365-370, 2011.

[24] W. Duch and N. Jankowski, Survey of Neural Transfer Functions, *Neural Computing Surveys*, Vol. 2, pp. 163-212, 1999.

[25] E. Duman and I. Or, Precedence constrained TSP arising in printed cir-

cuit board assembly, *International Journal of Production Research ISSN 00207543 print/ISSN 1366588X online # 2004 Taylor & Francis Ltd, http://www.tandf.co.uk/journals, DOI: 10.1080/00207540310001601073*, Vol. 42, No. 1, pp. 67-78, 2004.

[26] G. Feng and C. Douligeris, Using Hopfield Networks to Solve Traveling Salesman Problems Based on Stable State Analysis Technique, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks* Vol. 6, oo. 521-526, 2000.

[27] H. Ghaziri and I. H. Osman, A neural network algorithm for the traveling salesman problem with backhauls, *Computers & Industrial Engineering*, Vol. 44, pp. 267-281, 2003.

[28] M. Hagenbuchner, A. Sperduti and A. C. Tsoi, A Self-Organizing Map for Adaptive Processing of Structured Data, *IEEE Transactions on Neural Networks*, Vol. 14, No. 3, May 2003.

[29] H. Hamori, M. Sakawa, H. Katagiri and T. Matsui, A Fast Non-Contact Inspection System based on Dual Channel Measurement System, *Japan Institute of Electronics Packaging*, Vol. 13, No. 7, pp. 562-568, July 2010 (in Japanese).

[30] H. Hamori, M. Sakawa, H. Katagiri and T. Matsui, Fast Non-Contact flat Panel Inspection through a Duel Channel measurement System, *Proceedings of International Conference on Computers and Industrial Engineering* CD-ROM, July 2010.

[31] H. Hamori, M. Sakawa, H. Katagiri and T. Matsui, A Defect position Identification System based on a dual channel measurement system, *Journal of Japan Institute of Electronics, Information and Communication Engineers*, Vol. J94-C, No. 10, pp. 323-333, Oct 2011 (in Japanese).

[32] H. Hamori, M. Sakawa, H. Katagiri and T. Matsui, A Dual Channel Defect Position Identification Method for Touch Panel Manufacturing Process, *Proceedings of International Conference on Electronics Packaging 2011*, pp. 732-736, April

2011.

[33] F. Henley, Y. M. Liu and K. Otsuka, A High Speed Flat Panel In-Process Test System for TFT Array Using Electro-Optic Effects, *IEICE Transactions on ELECTRON* Vol. E76-C, No. 1, pp. 64-67, 1993.

[34] J. J. Hopfield, Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proceedings of the National Academy of Science of the USA*, Vol. 79, No. 8, pp. 2554-2558, 1982.

[35] B. Q. Huang, T. Rashid and M. T. Kechadi, Multi-Context Recurrent Neural Networks for Time Series Applications, *World Academy of Science, Engineering and Technology, International Journal of Computer, Information, Systems and Control Engineering*, Vol. 1, No. 10, pp. 3082-3091, 2007.

[36] M.Husken and P. Stagge, Recurrent Neural Networks for Time series Classification, *Neurocomputing*, 50(C), pp. 223-235, 2003.

[37] G. K. Jaiswal and R. Pal, Artificial Neural Network for ECG Classification, *Recent Research in Science and Technology*, 6(1), pp. 36-38, 2014.

[38] C. U. Joy, Comparing the Performance of Backpropagation Algorithm and Genetic Algorithms in Pattern Recognition Problems, *International Journal of Computer Information Systems*, Vol. 2, No. 5, pp. 7-12, 2011.

[39] N. K. Kasabov, *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*, MIT Press, 1998.

[40] H. Katagiri, I Nishizaki, T. Hayashida and T. Kadoma, Multiobjective Evolutionary Optimization of Training and Topology of Recurrent Neural Networks for Time Series Prediction, *The Computer Journal*, Vol. 55, No. 3, pp. 325-336, 2011.

[41] S. H. Kim, T. G. Kang and D. H. Jeong, Region Mura Detection using Efficient High Pass Filtering based on Flat Average Operation, *Proceedings of the 17th World Congress, The International Federation of Automatic Control*, Seol,

Korea, pp.8190-8195, 2008.

[42] T. Kohonen, Self-Organized Formation of Topologically Correct Feature Maps, *Biological Cybernetics*, Vol. 43, pp. 59-69, 1982.

[43] T. Kohonen, The Self-Organizing Maps, *Proceedings of the IEEE*, Vol. 78, No 9, pp. 1464-1480, September 1990.

[44] J. Y. Lee and S. I. Yoo, Automatic Detection of Region-Mura Defects in TFT-LCD, *IEICE Transactions on Information and Systems* Vol. 87-D, No. 10, pp. 2371-2378, 22004.

[45] W. C. Li and D. M. Tsai, Defect Inspection in Low-contrast LCD Images Using Hough Transform-based Non-stationary Line Detection, *The 10th Asia Pacific Industrial Engineering and Management Systems Conference*, Kitakyushu, Japan, pp. 62-73, 2009.

[46] Y. H. Liu and Y. J. Chen, Automatic Defect Detection for TFT-LCD Array Process Using Quasiconformal Kernel Support Vector Data Description, *International Journal of Molecular Science*, Vol. 12, pp. 5762-5781, 2011.

[47] Y.H. Liu, S.H. Lin, Y.L, Hsueh and M.J. Lee, Automatic target defect identification for TFT-LCD array process using kernel FCM based fuzzy SVDD ensemble, *International Journal of Expert Systems and Applications*, Vol 36, pp. 1978-1998, March 2009.

[48] Y.H. Liu, C. K. Wang, Y. Ting, W. Z. Lin, Z. H. Kang, C. S. Chen and J. S. Hwang, In TFT Array Process Micro Defect Inspection using Nonlinear Principal Component Analysis *International Journal of Molecular Sciences*, (Open Access) , pp. 4498-4514, 2009.

[49] C. J. Lu and D. M. Tsai, Defects Inspection of Patterned TFT-LCD Panels Using a fast Sub-Image base SVD, *Proceedings of the 5th Asia Pacific Industrial Engineering and Management Systems Conference*, pp. 3.3.1-3.3.16, 2004.

[50] C. J. Lu and D. M. Tsai, Automatic Defect Inspection for LCD using Sin-

gular Value Decomposition, *International Journal of Advanced Manufacturing Technology*, pp. 53-61, 2005.

[51] N. Maglaveras, T. StamKoposlos, K. Diamantaras, C. Pappas and M. Strintzis, ECG pattern recognition and classification using non-linear transformations and neutral networks: A review, *International Journal of Medical Informatics*, Vol 52, pp. 191-208, 1998.

[52] B. F. J. L. Maire and V. M. Mladenov, Comparison of Neural Networks for Solving the Traveling Salesman Problem, *IEEE 11th Symposium on Neural Network Applications in Electrical Engineering*, Belgrade, Serbia, 2012.

[53] C. von der Malsberg, Self-Organization of Orientation Sensitive Cells in the Striate Cortex, *Kybernetik*, Vol. 14, No. 2, pp. 85-100, 1973.

[54] J. Mandziuk, Solving the Traveling Salesman Problem with a Hopfield - type Neural Network, *Demonstratio Mathematica*, 29(1), pp. 219-231, 1996.

[55] J. Martens and I. Sutskever, Learning Recurrent Neural Networks with Hessian-Free Optimization, *Proceedings of the 28th International Conference on Machine Learning*, 2011 (`http://www.icml-2011.org/papers.php`).

[56] Per Mattsson, *The Asymetric Traveling Salesman Problem*, PhD thesis, Upsala University, Swedon, 2010.

[57] W. S. McCulloch and W. Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943.

[58] H. Minami, F. Matsumoto and S. Suzuki, Prospects of LCD Panel Fabrication and Inspection Equipment Amid Growing Demand for Increased Size, *Hitachi Review*, Vol. 56, No. 3, pp. 63-69, 2007.

[59] H. Minami, J. Mori, S. Iwai, H. Morida and N. Watanabe, Manufacturing and Inspection Equipments for Efficient Production of Large LCDs, *Hitachi Review*, Vol. 60, No. 5, pp. 228-232, 2011.

[60] M. Minsky, S. Papert, *Perceptrons: An Introduction to Computational Geome-*

try, MIT Press, 1969.

[61] C. Moon, J. Kim, G. Choi and Y. Seo, An efficient genetic algorithm for the traveling salesman problem with precedence constraints, *European Journal of Operational Research*, Vol. 140, pp. 606617, 2002.

[62] Nandish M. , S. Michahial, H.Kumar and F. Ahmed, Feature Extraction and Classification on EEG Signal Using Neural Network based Techniques, *International Journal of Engineering and Innovative Technology* , Vol. 2, No. 4, pp. 1-5, 2012.

[63] C. H. Noh, S. L. Lee, D. H. Kim and C. W. Chung, Effective Defects Classification for Flat Display Panel Film Images, *Proceedings of International Conference on Convergence and Hybrid Information technology*, pp. 264-267, 2009.

[64] H. Oba, Automatic Repair Technology for Fine Patterns in LCD manufacturing process, *NTN Technical Review*, No. 78, pp. 129-135, 2010.

[65] G. Ou and Y. L. Murphey, Multi-class pattern classification using neural networks, *Pattern Recognition*, Vol. 40, No. 1, pp. 4-18, 2007.

[66] L. M. Patnaik and O. K. Manyam, Epileptic EEG detection using neural networks and post-classification, *Journal of Computer Methods and Programs in Biomedicine*, Vol 91, pp. 100-109, 2008.

[67] Raul Rojas, Neural Networks A Systematic Introduction, *Springer-Verlag*, 1996.

[68] F. Rosenblatt, *Principles of Neurodynamics: Perceptons and the theory of brain mechanisms*, Spartan Books, 1962.

[69] D. E. Rumelhart, G. E. Hinton and R. J. Willioms, Learning representations by back-propagation error, *Nature*, pp. 533-536, 1986.

[70] D.E. Rumelhart and J.L. McClelland, Parallel distributed processing: Explorations in the micro structure of cognition - Volume 1. Foundations, *The MIT Press*, 1987.

[71] A. Salazar, J. M. Unio, A. Serrano and J. Gosalbez. Neural Networks for Defects Detection in Non-Destructive Evaluation by Sonic Signals, *IWANN'07 Proceedings of the 9th international work conference on Artificial neural networks*, pp. 638-645, 2007.

[72] R. D. Sathiya, V. Vaithiyanandana and G. B. Venkataraman, Extraction of the Ocean Wave Height Data using Neural Networks, *International Journal of Applied Engineering Research*, Vol. 8, No. 20, pp. 2625-2628, 2013.

[73] H. Schabauer, Solving Very Large Traveling Salesman Problem by SOM Parallelization on Cluster Architecture, *Proceedings on 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pp. 954-958, December. 2005.

[74] P. H. Siqueira, S. Scheer and M. T. A. Steiner, A Recurrent Neural Network to Traveling Salesman Problem, *Traveling Salesman Problem*, I-Tech, Austria, pp. 135-156, 2008.

[75] V. Stopjakova, P. Malosek and V. Nagy Neural Network-based Defect detection in analog and mixed IC using digital signal preprocessing, *Journal of Electrical Engineering*, Vol. 57, No. 5, pp. 249-257, 2006.

[76] Y. Takahashi, Solving a Dynamic Traveling Salesman Problem with an Adaptive Hopfield Network, *Complex Systems*, Vol. 10, pp.449-466, 1996.

[77] S. Timotheou, The Random Neural Network: A Survey, *The Computer Journal*, pp. 1-17, 2009.

[78] D. M. Tsai and C. Y. Hung, Automatic Defect Inspection of Patterned TFT-LCD Panels Using 1-D Fourier Reconstruction and Wavelet Decomposition, *International Journal of Production Research*, Vol. 43, No. 21, pp. 4589-4607, 2005.

[79] D. C. Tseng, Y. C. Lee and C. E. Shie, LCD Mura Detection with Multi-Image Accumulation and Multi-Resolution Background Subtraction, *International Journal of Innovative Computing, Information and Control* , Vol. 8, No.

7(A), pp. 4837-4850, 2012.

[80] S. Uchikoga. Future Trend of Flat Panel Displays and Comparison of its Driving Methods, *Proceedings of the 18th IEEE International Symposium on Power Semiconductor Devices and IC's*, Naples, Italy,pp.1-5, 2006,

[81] P. Werbos, *Beyond Regression; New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD Thesis, Dept of Applied Mathematics, Harvard University, 1974.

[82] Z. Yu and Z. Jian, Fuzzy recognition of the defect of TFT-LCD, *Proceedings of the International Conference of Optics and Photonics*, Vol. 5637, pp. 233-240, 2005.

[83] G. P. Zhang, Neural Networks for Classification: A Survey *IEEE Transactions on System, Man and Cybernetics - Part C: Applications and Reviews*, Vol 30, No 4, pp. 451-462, November 2000.

[84] W. Zhong, H, Shan, Z. Chen and L. Xia, Multiple Traveling Salesman Problem with Precedence Constraints based on Modified Dynamic Tabu Artificial Bee Colony Algorithm, *Journal of Information & Computational Science*, Vol 11.4, pp. 12251232, 2014.

[85] E. Zitzler, M. Laumanns and L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, *Proceedings of the Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pp. 95-100, 2001.