Automated Assessment Of Erythrocyte Parameters Using Artificial Neural Network

Md. Rejaul Karim Chowdhury

B.Sc. (Honors), Jahangirnagar University, Bangladesh, 1999

Thesis Submitted In Partial Fulfillment Of

The Requirements For The Degree Of

Master of Science

in

Mathematical, Computer, And Physical Sciences

(Computer Science)

The University Of Northern British Columbia

December 2006

Your file Votre référence
ISBN: 978-0-494-28419-3
Our file Notre référence
ISBN: 978-0-494-28419-3

Abstract

Artificial neural network techniques can be applied in conjunction with image analysis to determine erythrocyte parameters such as red blood cell count, hemoglobin value, and mean corpuscular hemoglobin. In this thesis, a new method to estimate the values of hemoglobin and red blood cell count is introduced. This method is based on a functional-link artificial neural network and microscopic image analysis. The neural network was trained using 800 blood sample images collected from Prince George Regional Hospital Laboratory. We used a variation of color properties as inputs to the functional-link artificial neural network, and the test results of the CELL-DYN 3200 System at the Hospital Laboratory were used as target values. Eleven variations of the backpropagation learning algorithms were applied during training. The trained network was tested against 200 blood samples. The output results were compared with those of the Hospital laboratory. The results of the simulation experiments are very promising. The proposed method achieved an overall 3.06% of simulation error for the hemoglobin value and that of 4.28% for the RBC count. In addition, a fast training time of less than 3 minutes was obtained.

The proposed technique is simple, fast, and accurate. Hence, the proposed technique will not only speed up the laboratory-reporting period for physicians but also can be a crucial and integral step in automating laboratory-reporting process. The proposed method can be implemented in hardware with minimal cost.

Contents

List of Tables

List of Figures

Publication from this thesis

1. Saif Zahir, Rejaul Chowdhury, Geoffrey W. Payne, "Automated Assessment of Erythrocyte Disorders Using Artificial Neural Network", The 6th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2006), pp. 776-780, Vancouver, August 2006.

Acknowledgements

I would like to express my deep gratitude to my Supervisor, Dr. Saif Zahir for his support throughout my master's study at UNBC. I am greatly indebted to him for his erudite supervision during the time of my research and writing of this thesis. His suggestions, comments, and encouragement helped me to complete this thesis.

My special thanks goes to Dr. Geoffrey W. Payne for his keen interest in this thesis project from the very beginning and his consent to serve as one of the members of my graduate committee. His great effort also helped us to get permission from the Northern Health Authority to access the Prince George Regional Hospital laboratory.

I extend my special thanks to Dr. Waqar Haque for serving on my graduate committee.

The work carried out in this thesis is financially partly supported by the Research Assistantship (RA) award from the Computer Science Program of UNBC. I am also very much thankful to the RA committee for granting me the award.

I am thankful to Dr. Jernej Polajnar and Dr. Alex Alagarsamy Arvind for what I have learnt from their courses. I also thank Dr. Siamak Rezaei for providing me valuable suggestions related to my research.

I am grateful to the Northern Health Authority for the permission to access the Prince George Regional Hospital Laboratory. I also express my deep gratitude to the Management and staffs of the Prince George Regional Hospital Laboratory, especially Mr. Ken Winning and Ms.Tamara, for providing me with all kinds of logistic support.

I would like to thank my friends at UNBC, Lele, Maruf, Subrata, Alim, Sharmin, Pruthvi, Srinivas, Bharat, and Khal.

I cannot thank my father, brothers and sisters enough for their love and support, which always act as a driving force behind me. Lastly and most importantly, I would like to thank my wife for her encouragement and patience during my extended absence.

Chapter 1

Introduction

Artificial neural networks are widely used in numerous fields including but not limited to medical applications. In this thesis, we attempt to employ artificial neural network to determine some blood parameters that enable the detection of certain types of disorders in human blood. In this introduction, we provide a short description on human blood components and artificial neural network.

1.1 Overview of Human Blood

Blood is the fluid of life, growth, and health that transports oxygen from the lungs to body tissue, carbon dioxide from body tissue to the lungs, nourishment from digestion and hormones from glands throughout the body, and also disease fighting substances to the tissue and waste to the kidneys [1]. It has different components. Four of the most important ones are erythrocytes or red cells, leukocytes or white cells, platelets, and plasma.

Red cells are relatively large microscopic cells without nuclei, which normally make up 40-50% of the total blood volume [2]. The red cells are rich in hemoglobin. Hemoglobin is the oxygen-carrying protein molecule that makes up 95% of a red cell and gives blood its red color. Each red cell has about 270,000,000 iron-rich hemoglobin molecules [3]. These cells are responsible for transporting oxygen throughout the body.

White blood cells are microscopic cells with nuclei, which are much less numerous than red [3]. These are cells of the immune system, responsible for protecting the body from infection and malignancy.

Platelets are a type of blood cell that play a key role in normal blood clotting at the site of a wound. During the clotting process, platelets clump together to plug small holes in damaged blood vessels. They stimulate the immune system. Individual platelets are about 1/3 of the size of red cells. Like the red and white blood cells, platelets are produced in bone marrow from stem cells.

Plasma is the liquid component of blood in which red cells, white cells, and platelets are suspended [4]. Normally, 55% of our blood's volume is made up of plasma, which is mainly composed of water, blood proteins, and inorganic electrolytes. Plasma serves as a transport medium for glucose, lipids, hormones, carbon dioxide and oxygen [4].

1.1.1 Complete Blood Count

Blood testing refers to the laboratory analysis of blood. A variety of different blood tests are available to provide information about the condition of the body. The most routine test is the complete blood count. A complete blood count is a series of tests used to evaluate the composition and concentration of the cellular components of blood [4]. It consists of the following tests:

White blood cell (WBC) count: WBC count refers to the number of white blood cells in a volume of blood. The normal range varies slightly between laboratories but is generally between 4.5 - 11.0 x 10^9 cells per liter [4].

Automated white cell differential: White blood cells usually split into neutrophils 50–70%, lymphocytes 25–35%, monocytes 4–6%, eosinophils 1–3%, basophils 0.4–1%, and bands 0–5% of total WBC [4].

Red blood cell (RBC) count: RBC count refers to the number of red blood cells per volume of blood. The normal range is generally between 4.3 - 5.5 x 10^{12} cells per liter for males and 4.0-5.0 x 10^{12} cells per liter for females measured using CELL-DYN 3200 system [2].

Hemoglobin value : Hemoglobin value refers to the amount of hemoglobin in a volume of blood. The normal range is approximately 132–159 g/L for males and 123–151 g/L for females measured using CELL-DYN 3200 system [2].

Mean cell volume (MCV): MCV is the average volume of the individual red blood cells. The normal range is 81-95 fL (femtoliters) for males and 82-97 fL for females measured using CELL-DYN 3200 system [2].

Hematocrit (Hct): Hct is the ratio of the volume of red cells to the volume of whole blood [4]. It is calculated from the red blood cell count and the mean cell volume: HCT= (RBC × MCV)/10. The normal range for hematocrit is approximately 40–48% for males and 37–44% for females measured using CELL-DYN 3200 system [2].

Mean corpuscular hemoglobin (MCH): MCH is the average amount of hemoglobin contained in the red blood cell [1]. This value is derived from the measurement of hemoglobin and the red cell count: MCH= (HGB/RBC). The normal range of MCH is 26.9-32.2 pg (picograms) for males and 26.6-32.9 pg for females measured using CELL-DYN 3200 system [2].

Mean cell hemoglobin concentration (MCHC): MCHC is the ratio of the weight of hemoglobin to the volume of the average red blood cell [3]. This is a calculated volume derived from the hemoglobin measurement and the hematocrit: MCHC= (HGB/HCT)×100. The normal range is 31.8-36 g/dL for males and 31.3-35 g/dL for females measured using CELL-DYN 3200 system [2].

Red cell distribution width (RDW): RDW is a measurement of the variability of red cell size [3]. The normal range is 11.4-15.4% for males and 11.5-15.4% for females measured using CELL-DYN 3200 system [2].

Platelet count: Platelet count refers to the number of platelets in a volume of blood. The normal range is in the range of 150,000 - 400,000/ cmm (150 - 400 x 10^9/liter) [4].

1.1.2 Methods for Testing Various Values of Complete Blood Count

In this section, we provide a brief introduction to several methods for testing the values of a complete blood count. There are two main types of procedures for testing the values of a complete blood count: (1) manual and (2) automated. The following are some examples of both types.

1. Manual procedure for determining hemoglobin concentration in whole blood:

This procedure is straight forward and requires the use of spectrophotometer. This instrument measures monochromatic light transmitted through a solution to determine the concentration of the light absorbing substance in that solution. Light from the lamp passes through the prism, which allows light of only a chosen wavelength to pass through the cuvette. The transmitted light strikes a detector, where it is converted into electrical energy and presented to the readout device [4]. This procedure is satisfactory but requires the attendance of labor as well as a significant amount of time.

2. Automated procedure for determining hemoglobin concentration in whole blood:

A modern automated complete blood count analyzer is the CELL-DYN 3200 System that combines spectrophotometry and a modified cyanmethemoglobin method for hemoglobin determination. In the cyanmethemoglobin method, whole blood is mixed with a solution of potassium ferrycyanide to convert the hemoglobin in the ferrous state, to methemoglobin in the ferric state, which then reacts with potassium cyanide to form cyanmethemoglobin, also called hemiglobincyanide (HiCN). HiCN is very stable and has a broad absorption maximum around 540 nm. The absorption of the solution at 540 nm is directly proportional to the amount of hemoglobin present [5]. The CELL-DYN system measures hemoglobin within a sample in a Hemoglobin Flow Cell. The hemoglobin dilution that is analyzed in the Flow Cell is a mixture of the sample +

reagent. The System takes five reference readings. The lowest and highest are eliminated, and the remaining three are averaged to give the final hemoglobin reading. The hemoglobin dilution is analyzed at 555 nm in the Hemoglobin Flow Cell; the System receives, then saves the results [6].

3. Automated procedure for counting different blood cells (RBC, WBC, platelet):

There are two general principles of instrument operations.

3.1 Electronic Impedance principle:

Cells suspended in an electrically conductive diluent increase the resistance between two electrodes when passing through a sensing aperture. The impedance of the direct current creates measurable voltage pulses. The size of the pulse generated by the cell is proportional to its volume. The cell count is determined by the number of pulses generated. Separate channels are used for counting WBC and RBC [5].

3.2 Light Scattering principle:

Cells are detected and counted as they pass through a focused beam of light. Light scattered by cells is the summation of three independent processes: diffraction (bending around corners), refraction, and reflection. Cells scatter light in all directions, with diffraction generally dominating at small angles relative to the incident light, reflection dominates at larger angles, and refraction generally dominates at intermediate angles. Light scattered by a particle in the sensing zone of the flow cell is detected by the appropriately placed photo detectors which are summed for the cell counts. Photo diodes and photo multiplier tubes (PMT) are commonly used for this purpose. Photo diodes are light detectors that are not very sensitive but are sufficient to detect forward scatter, which has a relatively strong light value. On the other hand, PMTs used to detect 90 degree scatter, are sensitive to weak light levels and multiply weak signals into stronger ones [5]. Forward low-angle light scatter correlates with cell volume primarily because of the diffraction of light (bending by the outside surface of the cell). Forward high degree scatter measurements depict the degree of structure inside the cell. Differential scatter is the combination of low and high angle forward scatter. This scatter analysis is used

to analyze RBC and WBC. The intensity of light scatter at larger angles is attributable primarily to refraction and reflection of light from larger structures inside the cell. Structures such as nuclei and cytoplasmic granules determine the intensity of the light scattering at right angles [5].

The CELL-DYN 3200 System uses the Flow Cytometry method for counting RBC, WBC, and the platelet that is based on the light scattering principle. The system also measures MCV using the same principle. Later the rest of the values of the complete blood count are obtained simply by mathematical calculation [2]. On the other hand, the CELL-DYN 4000 system is implemented using the electronic impedance principle.

At present, both the CELL-DYN 3200 and the CELL-DYN 4000 systems are being used in hospital laboratories in different countries. Both systems are considered to be reliable but expensive.

1.1.3 Diagnosis of Diseases Based on Erythrocyte Parameters

According to the Medical Encyclopedia of the United States National Library of Medicine [7], the erythrocyte parameter, RBC count, may indicate several abnormalities in human health and also aids in the diagnosis of several diseases. For example, a higher-than-normal numbers of RBCs may indicate [7]: (i) Congenital heart disease (CHD): CHDs are abnormalities of the heart's structure and function caused by abnormal or disordered heart development before birth [8]; (ii) Cor pulmonale: Cor pulmonale is a failure of the right side of the heart caused by prolonged high blood pressure in the pulmonary artery and right ventricle of the heart [9]; (iii) Pulmonary fibrosis: Pulmonary fibrosis is a disease of the lower respiratory tract that damages the air sacs and leads to reduced transfer of oxygen to the blood [10]; (iv) Polycythemia vera: Polycythemia vera can be termed as myeloproliferative disorders, where blood cell production, notably red cells, is increased as a result of an acquired stem cell mutation [11]. The disease usually develops slowly, and most patients do not experience any problems related to the disease

after being diagnosed. However, the abnormal bone marrow cells may begin to grow uncontrollably in some patients leading to acute myelogenous leukemia [12]; (v) Dehydration: The two main causes of dehydration are vomitting and diarrhea.

On the other hand, lower-than-normal numbers of RBCs may indicate [7]: (i) Anemia: Anemia is a disease caused by the reduction of RBC count and/or hemoglobin value in human blood. Severe anemia can cause low oxygen levels in vital organs such as the heart and can lead to heart attack [13]; (ii) Hemorrhage (bleeding); (iii) Bone marrow failure (for example, from radiation, toxin, fibrosis, tumor); (iv) Hemolysis (RBC destruction) from transfusion reaction: Transfusion reaction is a complication of blood transfusion where there is an immune response against the transfused blood cells or other components of the transfusion [14]; (v) Leukemia: Leukemia is a cancer of blood cells. When leukemia develops, the blood produces large number of white blood cells. Leukemia patients often donot have enough healthy red blood cells [15]. As a result, the body does not receive enough oxygen; (vi) Multiple myeloma: Multiple myeloma is characterized by the excessive growth and malfunction of plasma cells in the bone marrow. The growth of these extra plasma cells interferes with the production of red blood cells, white blood cells, and platelets [16]; (vii) Malnutrition.

Another erythrocyte parameter, MCH value, is useful in the diagnosis of certain types of anemia. As for example, if MCH is less than the lower limit of normal, it indicates hypochromic anemia; if MCH is within the normal range, it indicates normochromic anemia; finally if MCH is greater than the upper limit of normal, it is the index of hyperchromic anemia. [17].

1.2 Basics of Artificial Neural Network

An ANN is an approximate representation of the human brain. The human brain has an estimated 10^{11} tiny units called neurons, which are interconnected with an estimated 10^{15} links [18]. Similarly an ANN is composed of a large number of highly interconnected artificial neurons (processing elements) working in parallel to solve a specific problem.

Different authors have defined artificial neural networks in various ways. According to Haykin [19]: "A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Inter-neuron connection strengths known as synaptic weights are used to store the knowledge."

Nigrin [20] on the other hand defined it as: "A circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore each element operates asynchronously; thus there is no overall system clock."

In conclusion, an artificial neural network is a powerful data-modeling tool capable of capturing and representing relationships. The advantages of neural networks lie in their ability to represent both linear and non-linear relationships and to learn these relationships directly from the data being modeled. Due to this reason, neural networks offer the opportunity of solving problems in an arena where traditional computing environments fail. At present, neural networks have numerous real-life applications such as natural language processing, character recognition, image processing and compression, speech recognition and compression, financial decision making, stock market trading analysis, medical data analysis and diseases diagnosis to name a few.

An artificial neuron, or simply neuron, is the basic building block for all types of neural networks. Each neuron has a collection of input links from other neurons (X_1, X_2, ...,X_n) and a set of output links to other neurons (Y_1, Y_2, ...,Y_m). All links are associated with a corresponding weighting factor where the network's knowledge is stored. Each neuron performs a simple computation: it receives signals from its inputs and computes a new activation level that it sends along each of its output links [21]. A simple schematic neuron model is illustrated in Fig. 1.1.

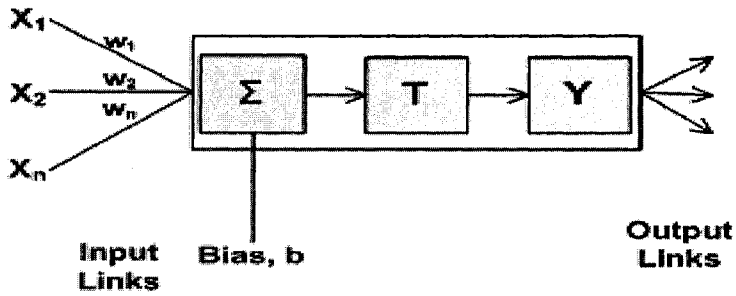


Figure 1.1: A schematic of a simple neuron

To generate a generic artificial neural network model, the inputs X_1, X_2, ..., X_n are entering into the neuron; each input X_i is multiplied by its corresponding weight w_i, then the product $X_i w_i$ is fed into the neuron. The neuron adds up all the products for i=1, ..., n. The weighted sum of the products is denoted by the summation function, ($\sum$). In certain cases, the summation function includes bias functions attached with each neuron to influence certain desired output. Mathematically, the summation function can be expressed as:

$$(\Sigma) = X_1 \times w_1 + X_2 \times w_2 + \cdots + X_n \times w_n + b \qquad (1)$$

where b represents the bias function attached to the neuron.

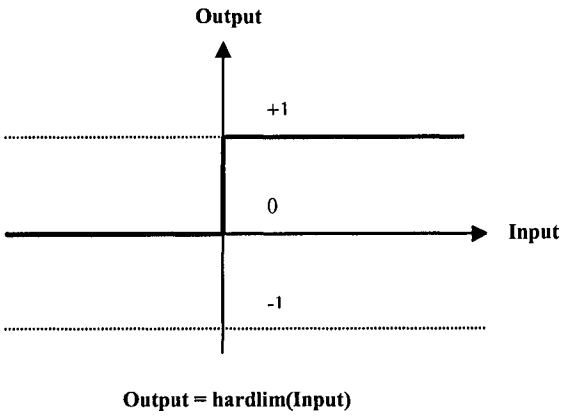
Finally, the neuron computes its output Y as a function of ($\sum$), i.e., Y = T ($\sum$). The function, T is called a transfer function. There are many transfer functions in the literature. Among them, (1) the hard-limit transfer function, which limits the output of the neuron to either 0 or 1; (2) the log-sigmoid transfer function, which allows the input value between plus and minus infinity, and squashes the output into the range 0 to 1; (3) the tan-sigmoid transfer function takes the input value in the same range as log-sigmoid function, and squashes the output into the range -1 to 1; (4) the pure linear function has both input and output values from plus to minus infinity. The transfer functions of hard-limit, log-sigmoid, tan-sigmoid, and pure linear neurons are given in equations (2), (3), (4), & (5) respectively. The plots of these transfer functions are depicted in Fig. 1.2.

$$T(x) = \begin{cases} 1 \ ; \text{ if } \sum \geq 0 \\ 0 \ ; \text{ Otherwise} \end{cases} \quad (2)$$

$$T(x) = \frac{1}{1+e^{-x}} \quad (3)$$

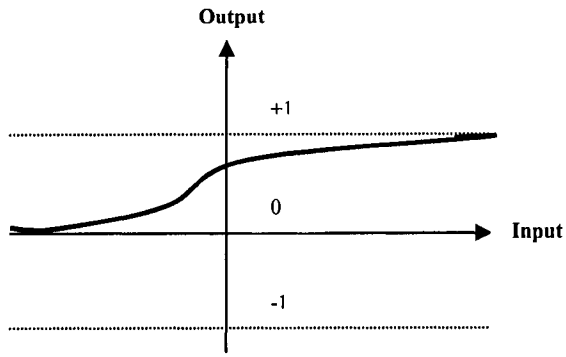
$$T(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (4)$$

$$T(x) = x \quad (5)$$



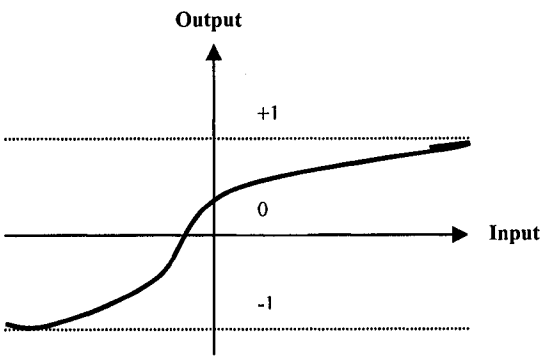
Output = hardlim(Input)

(a)



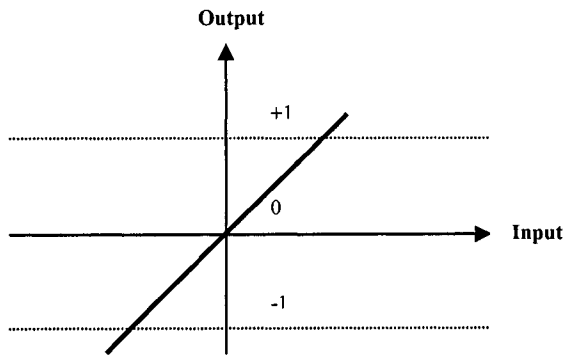
Output = logsig(Input)

(b)



Output = tansig(Input)

(c)



Output = purelin(Input)

(d)

Figure 1.2: (a) Hard-Limit transfer function (b) Log-Sigmoid transfer function (c) Tan-Sigmoid transfer function (d) Linear transfer function

The architecture of an artificial neural network consists of the description of how many layers a network has, the number of neurons in each layer, each layer's transfer function, and the pattern of connections between the neurons. Different network architecture results in different computational properties. The best architecture to use depends on the type of the problem to be represented by the network.

The pattern of connections between the neurons can be categorized as: feed-forward and recurrent networks [21]. In a feed-forward network, links are unidirectional, and there are no cycles. A feed-forward network diagram is shown in Fig. 1.3. The main characteristics of this network are as follows:

(i) Neurons are arranged in layers. The first layer of neurons, which receives the inputs, is called input layers. The layer of neurons producing the output values is called output layer. The layers of neurons between input and output layers are referred to as hidden layers.

(ii) Each neuron in one layer is connected to every neuron on the next layer. So the input is constantly "fed forward" from one layer to the next until the values of the output units are determined.

(iii) There is no link among neurons in the same layer, no link backward to a previous layer and no link that skips a layer.

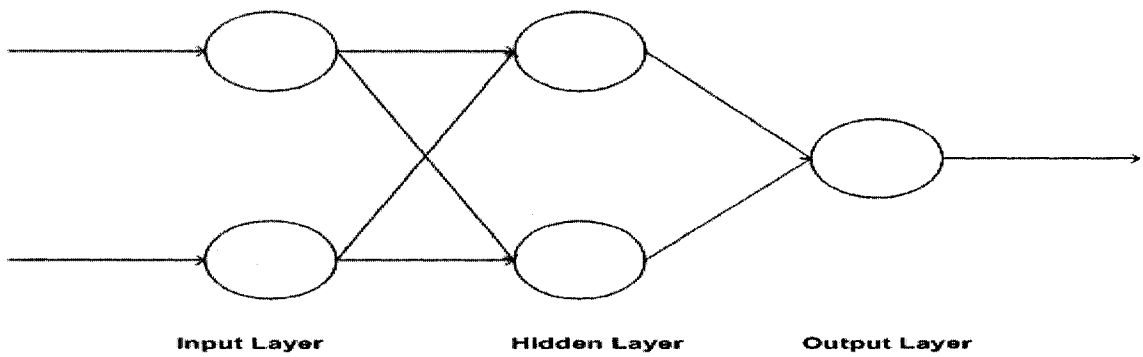


Figure 1.3: Feed-forward network

Feed-forward networks often have one or more hidden layers of neurons with sigmoid transfer function followed by an output layer of neurons with linear transfer function.

Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors [22].

On the other hand, a recurrent network is defined as one in which the outputs can propagate in both directions forward and backward. In this model, either the network's hidden unit activations or output values are fed back into the network as inputs. As the output of the network at time (t) is used along with a new input to compute the output of the network at time (t+1), the response of the network is dynamic. The recurrent network is shown in Fig. 1.4.

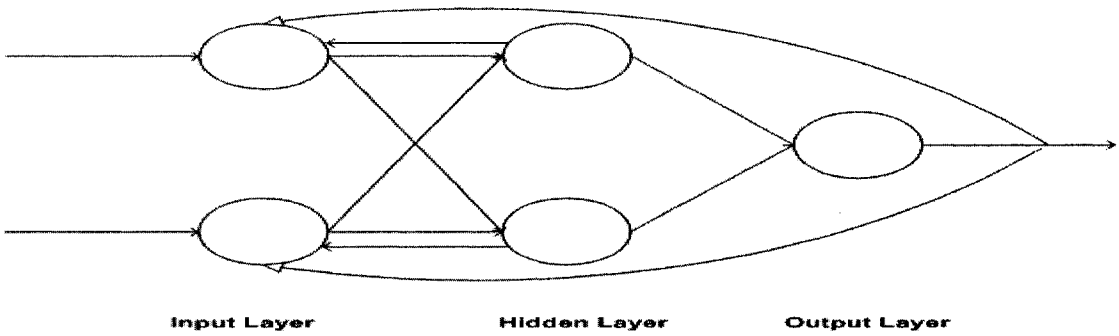


Figure 1.4: Recurrent network

1.2.1 Backpropagation Learning Algorithm

The backpropagation learning algorithm, also referred to as the gradient descent rule, is the generalized version of Widrow-Hoff algorithm to multi-layer networks and non-linear differentiable transfer function. This learning algorithm trains a multi-layered feed-forward network for a given set of input patterns with known classifications. The backpropagation algorithm consists of two phases: (i) the forward phase where the input activations are propagated from the input to the output nodes; and (ii) the backward phase, where the error between the observed output and the target output is propagated backwards from output to the inner nodes. The algorithm works as follows:

(i) Each entry of the training sample set is presented to the network.

(ii) Typically the weights in a neural network are initially set to small random values.

(iii) Network's output response to the sample input pattern is examined and then compared to the target (known and desired) output and the error value in each output neuron is calculated.

(iv) For each output neuron, a scaling factor is computed to determine how much lower or higher the output must be adjusted to match the desired output.

(v) The weight and bias of each output neuron are adjusted according to the scaling factor to minimize the error.

(vi) Now, the error is back propagated to the neurons at the previous layers and accordingly weights and biases are adjusted, giving greater responsibility to neurons connected by stronger weights.

(vii) After weights and biases of each layer are adjusted, the steps from (i) to (v) are repeated until the desired performance function is reached.

The backpropagation algorithm calculates the gradient of the performance function and updates the network weights and biases in the direction along the negative of the gradient in which the performance function decreases most rapidly [22]. Usually, the mean square error between the target and the output value is used as the performance function of this algorithm. Here the gradient is on the error surface: the surface that describes the error on each example as a function of the all the weights in the network [21]. An ideal error surface for gradient descent search in weight space is shown in Fig. 1.5 [21].

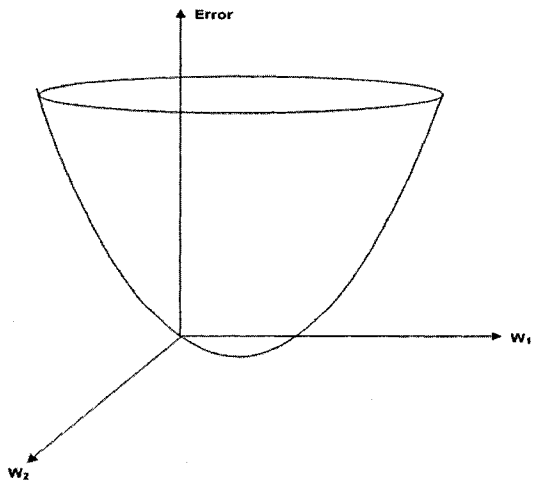


Figure 1.5: An ideal error surface

In Fig. 1.5, weight vectors W_1 and W_2 form axes on the error surface and a set of weights define a point on this surface. At that point, the slope of the surface along the axis formed by each weight is calculated. This is known as the partial derivative of the surface with respect to each weight - how much the error would change if a small change in weight is made [21]. The weights are altered in an amount proportional to the slope in each direction. This moves the network as a whole in the direction of steepest descent on the error surface.

There are limitations to the backpropagation learning algorithms. Backpropagation requires lots of supervised training, with lots of input-output examples to learn a concept. Additionally, the internal mapping procedures are not well understood, and there is no guarantee that the system will converge to an acceptable solution. At times, the learning gets stuck in local minima, which limits the best solution. Many learning applications add an additional term in the computations to overcome the problem.

1.2.2 Variations of Backpropagation Learning Algorithms

The backpropagation algorithm is widely used method for training a multilayer feed-forward network to perform supervised learning tasks. The goal of implementation of this algorithm is not only to learn the patterns in the training set but also to learn the characteristics of mapping that enables generalization (i.e. produce correct responses for patterns that the network has not been trained on). Since the first publication of the backpropagation algorithm by Rumelhart et al. [23], a large number of research activities have been carried on the development of algorithms that improve on backpropagation. The researchers mainly focused on the improvement of the following criteria:

- Training speed- algorithms which aim to converge towards the goal in as little time as possible.
- Performance function- algorithms which aim to devise an efficient performance function (e.g. mean square error, sum square error etc.) during training.
- Error value- algorithms which aim to train the network to lower error values.
- Generalization- algorithms which aim to improve the generalization ability of the trained network.

Based on these above criteria, several variations of backpropagation learning algorithms have been proposed in the literature. In this thesis, we briefly discuss some of these algorithms namely: standard backpropagation learning algorithm (Gradient Descent), Gradient Descent with Momentum, Gradient Descent with Variable Learning Rate, and Conjugate Gradient algorithms in the following sections.

1.2.3 Standard Backpropagation Learning Algorithm

The standard backpropagation is a simple Gradient Descent algorithm. There are several ways in which this gradient descent algorithm can be implemented.

Batch mode Gradient Descent: In this training algorithm, all of the inputs are applied to the network before the weights are updated [22]. The gradients calculated at each training example are added together to determine the change in the weights and biases. The algorithm proceeds as follows:

i) Initialize the network weights.

ii) Repeat the following steps:

- Process all the training data to compute the gradient of the average error function.
- Update the weights by subtracting the gradient times the learning rate.

Incremental mode Gradient Descent: In this training algorithm, the gradient is computed and the weights are updated after each input is applied to the network [22]. The algorithm proceeds as follows:

i) Initialize the network weights.

ii) Repeat the following steps:

- Process one training case to compute the gradient of the average error function.
- Update the weights by subtracting the gradient times the learning rate.

1.2.4 Gradient Descent with Momentum

In gradient descent algorithm, training is started at some point on the error surface defined over the weights. The training process attempts to move the error function towards the global minimum of the error surface. In a simplified error surface, shown in

Fig. 3.5, any step in the downward direction will shift the error function closer to the global minimum. But in case of real problems, error surfaces are typically complex as shown in Fig. 1.6 where weight vector W_1 and W_2 form axes on the error surface.

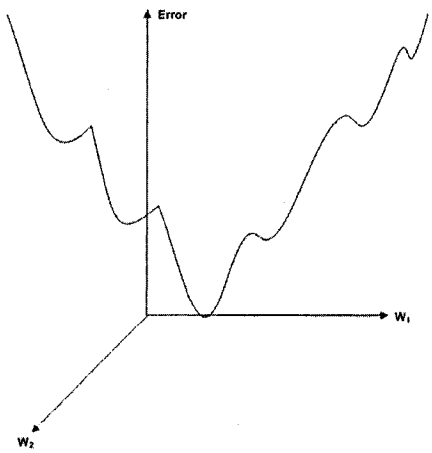


Figure 1.6: A real problem error surface.

It has also been proved formally that even in a very simple situation there exist numerous local minima [24]. Auer et al. [25], while examining a single neuron network, proved that the error surface could have numerous local minima for any combination of the error and transfer functions. The error function may be trapped in one such minima. Rumelhart et al. [23] introduced a momentum term into the backpropagation weight update rule, as without momentum a network may get stuck in a swallow local minimum. Momentum not only allows a network to respond to both the local gradient and the recent trends in the error surface but also provides faster convergence.

Momentum can be added to backpropagation learning by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the backpropagation rule [22]. When the momentum constant is 0, a weight change is based solely on the gradient. When the momentum constant is 1, the new weight change is set to equal the last weight change and the gradient is simply ignored. So the range of momentum constant is between 0 and 1. When the gradient keeps pointing in the same direction, momentum will increase the size of the steps taken towards the minimum. When the gradient keeps changing direction, momentum will smooth out the variations. This phenomenon is very useful for ill-conditioned network where the error surface has

substantially different curvature along different directions. For most of the points on the error surface, the gradient does not point towards the minimum, and successive steps of Gradient Descent can oscillate from one side to other, progressing only very slowly to the minimum [26]. The addition of momentum helps to speed up convergence towards the global minimum of the error surface by dumping these oscillations.

1.2.5 Gradient Descent with Variable Learning Rate

Various variable learning rate adaptations have been adopted with the standard backpropagation algorithm to improve the convergence speed and avoid convergence to local minima.

The standard Gradient Descent algorithm keeps the learning rate constant throughout the training process. The value of learning rate influences the performance of the algorithm very much; with too high learning rate an algorithm can become unstable, with too small learning rate the algorithm can take long time to achieve the performance goal [22]. It is difficult to determine the optimal setting for the learning rate before training. An optimal learning rate should be changed during the training process while the algorithm advances towards the performance goal. Allowing the learning rate to change during the training process improves the performance of the standard descent algorithm. An adaptive learning rate maintains the learning step size as large as possible in accordance with the stability in learning rate. There are several strategies relative to adaptive learning rate that aim to accelerate the learning process.

In strategy 1 [22] [27] [28], training is first started with small learning rate. Then using the current learning rate, new weights and biases are computed at each epoch and consequently outputs and errors are calculated. If the new error is greater than the old error by a predefined amount, the new weights and biases are discarded and the learning rate is decreased. Otherwise, the new weights are stored. If the new error is less than the old error, the learning rate is increased.

In strategy 2 [29], training is first started with small learning rate. If successive epochs

keep gradient direction fairly constant, the rate is increased. If the direction of the gradient varies greatly at each epoch, the learning rate is rapidly decreased.

In strategy 3 [30] [31] [32] [33], an individual learning rate is given for each weight. The learning rate is increased if the successive changes in the weights are in the same direction and decreased otherwise.

All the above–mentioned strategies attempt to enforce the decrease of the learning error at each iteration and thus secure the convergence of the training algorithm.

1.2.6 Conjugate Gradient Algorithms

The conjugate gradient method was developed independently by Hestenes [34] and Stiefel [35] for solving linear equations [36]. Fletcher and Reeves [37] generalized this method for nonlinear problems based on previous work by Davidson [38], and Fletcher and Powell [39] [36]. This algorithm uses conjugate directions instead of the local gradient for going downhill towards the local minima. Whereas steepest descent approaches the solution asymptotically by performing many small steps in going down a long, narrow valley of the error surface, the conjugate gradient method will find the solution within few (n) iterations. Typically, conjugate gradient algorithms allow the step size to be adjusted in every iteration. The step size is determined by searching along the conjugate gradient direction, which minimizes the performance function. The conjugate gradient algorithms usually run according to following steps [22]:

Step 1: An initial search direction is chosen using the initial gradient on the first iteration.
Step 2: A line search is then carried out to determine the optimal distance to move along the current search direction.
Step 3: Then the next search direction is determined so that it is conjugate to previous search direction. The general procedure for determining the next direction is to combine the new steepest descent direction (negative of the current gradient) with the previous search direction.

There are several versions of this algorithm such as Fletcher-Reeves Update, Polak-Ribiére Update, Powell-Beale Restarts, Scaled Conjugate Gradient etc. These versions are distinguished by the variations of the update formula required for generating the conjugate directions. Most of the conjugate gradient algorithms do not require computing the Hessian (second order derivative). So this first-derivative method is best for large problems with large number of weights where the gradients can be computed much faster than the Hessian and where too much memory is required to store the Hessian. So these algorithms are usually much faster than Variable Learning Rate backpropagation algorithm and are sometimes faster than Resilient backpropagation algorithm, although the results may vary from one problem to another [22]. The memory required by these algorithms is proportional to the number of weights. Each of the conjugate gradient algorithms except the Scaled Conjugate Gradient requires a line search at each iteration. This line search is computationally expensive, since it requires that the network response to all training inputs be computed several times for each search. By using a step size scaling mechanism Scaled Conjugate Gradient algorithm avoids the time consuming line-search per learning iteration, which makes the algorithm faster than other second order algorithms [40].

Due to its numerous advantages, researchers have used conjugate gradient algorithms for many applications such as image processing [41] [42], agricultural prediction [43], power system control [44] etc.

1.3 Thesis Contributions

This thesis contains the following contributions.

- An image dataset has been built by capturing digital microscopic color images (at 10× magnification) of the blood samples smeared on the glass slides. All examined values of a complete blood count test using a standard laboratory method (CELL-DYN 3200 System) of the corresponding blood samples have been included in the dataset.

- A functional-link artificial neural network with microscopic color image analysis method to determine erythrocyte parameters (hemoglobin value and RBC count) has been introduced.
- Based on the hemoglobin value and the RBC count, the MCH value in human blood has also been calculated to aid physicians categorize certain types of anemia.
- Eleven variations of backpropagation learning algorithms, exploiting different network architectures, have been applied to the automated systems to measure the simulation error for each algorithm. Based on the minimum simulation error of the test samples, an optimal and reliable medical diagnostic model has been proposed.

MATLAB Interpreter was used for the implementation of the artificial neural network algorithms.

1.4 Thesis Organization

In Chapter 1, an introductory background related to human blood components and complete blood count has been presented. Then the current approaches for testing various values of complete blood count have been explained. An overall discussion related to artificial neural networks has also been presented in this chapter.

In Chapter 2, previous work related to hemoglobin value estimation, anemia classification, blood cell count and analysis, and other types of cell count and analysis has been investigated.

In Chapter 3, the proposed scheme and its implementation have been introduced. Then the experimental approaches, along with the simulation results and analysis, have been described. Later, an overall discussion on the simulation study has been presented. Finally, the implemented artificial neural network model has been presented along with the simulation results.

In Chapter 4, there is a conclusion emphasizing the significance of the research work accomplished in this thesis. At last, an outlining of the possible future work that can be based on this thesis is provided.

Chapter 2

Related Work

2.1 Introduction

Artificial neural networks can be applied to medicine in several fields: simulating and modeling the functions of the brain, bioelectric signal filtering i.e. signal processing, intelligent artificial machine control and checking based on responses of biological or technical systems given to any signals, interpretation of physical and instrumental findings to achieve more accurate diagnosis i.e. classification, providing prognostic information based on retrospective parameter analysis i.e. prediction [45].

Neural networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognize the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The examples need to be selected very carefully if the system is to perform reliably and efficiently. Neural networks can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

Applications of neural networks in the medical arena fall into two broad categories: pattern classification and image processing. Several applications of pattern classification using ANN are: diagnosing acute myocardial infarction by analyzing heart serum enzyme levels [46], choosing therapy for peptic ulcers [47], diagnosing breast cancer [47], interpreting tissue sections and blood chemistry [48], diagnosing dementia [48],

interpreting electrocardiograms [48], determining the prognosis of patients undergoing hepatectomy [49] etc. Again several applications of image analysis using ANN are: interpreting radiograms [48], recognizing macromolecules [50], assisting in breast cancer diagnosis [51], classifying nuclei by texture [52], diagnosing shape abnormalities of the cornea [53] etc.

2.2 Related Research on Hemoglobin Estimation and Cell Count

The complete blood count has been a target for several automations attempts by researchers from different fields. For the purpose of this research, we classify published research attempts into two categories: (1) artificial neural network (ANN) based methods, (2) other methods.

1. ANN based methods

1.1 ANN based methods for hemoglobin value estimation and anemia classification:

Ranganathan and Gunasekaran [54] implemented the ANN approach in the estimation of hemoglobin in human blood. They used the backpropagation learning algorithm to estimate the hemoglobin value using the color information of the blood sample. They trained their network on a limited number of samples and tested on a few samples (100). They claim results accuracy within 15% to 20% of real values. They reported their training time as 20 hours. But their procedure of smearing the blood samples over the slides was not standard as it was manual. So there was a chance of affecting the results due to the non-uniform smearing of blood over the slides. More over, they used a hand-held digital camera to capture the images of the blood samples slides. So the accuracy of their results was mostly dependent on the angle of taking photograph, distance between the camera and the slide and the ambient light.

X Liu, G Loizou, and S Jami [55] worked on Hemoglobin identification by artificial intelligence. To identify hemoglobin, they took into consideration domain knowledge (general biochemistry knowledge about proteins and hemoglobin, various hemoglobinopathy tests, etc.). These data include those from isoelectric focusing and electrophoresis, HPLC(high performance liquid chromatography),

mass spectrography, the DNA structural analysis, tests about oxygen affinity, and stability of hemoglobin and those about the patient. They implemented a system which emulates the identification procedures followed by the experts.

Another group of researchers also worked on the classification of anemia using the neural network approach [56]. They collected information regarding anemia cases from hematology forms, which were used in the government hospital. Seventeen attributes were identified and used in the training model. They used a multi-layer perceptron with 17 units of input layer, 15 units of hidden layer and 8 units of output. They demonstrated the ability of the multi-layer perceptron for predicting classes of anemia, which could be used by the hematologists and other medical staff.

1.2 ANN based methods for white blood count and analysis:

Nipon and Gader [57] proposed a new method for applying neural networks to the bone marrow white blood cell counting problem. A batch-mode training scheme based on back-propagation and gradient descent was derived. The authors categorized white blood cells by age into discrete classes and classified each white blood cell into one of the classes and then counted the number that was assigned to each class. They designed a feed-forward classification network with the backpropagation algorithm and class-coded output. They used one hidden layer containing 12 hidden neurons to train a neural network by minimizing the error between the total number of cells in a class estimated by the network and estimated by an expert. Their testing results showed that, although yielding lower classification rates, the network trained to minimize count error performs better in counting than a classification network with the same structure.

Kovalev, Grigoriev, and Ahn [58] investigated the white blood cell (WBC) image recognition problem at all involved stages and introduced a robust and effective method for automatic WBC differentiation based on both statistical pattern recognition and neural network approaches. In their proposed method, they first segmented the stained blood images to extract blood cells from a complicated

background into such morphological components as nucleus, cytoplasm, holes, and some others. They proposed a three-step algorithm for segmentation: extraction of nucleus segments and grouping them into cells, a circle-shaped approximation of a cytoplasm region around nucleus center, and the cytoplasm improvement considering a priori knowledge of background color and possible cell occlusions. After the segmentation process, several WBC features such as the nucleus center, the cell area and the nucleus area, the area of nucleus holes, the length of external and internal borders etc. were extracted. At the final step, WBC cells were recognized using both the statistical and the neural network approach. The neural network approach was implemented using the backpropagation algorithm with an adaptive learning rate and momentum. The network was trained using a representative training image set and vectors of extracted features. The authors claimed that the recognition accuracy on the test set of 662 images of five WBC types obtained at different conditions was not less than 98%.

1.3 ANN based methods for other cell count and analysis:

Sjostrom, Frydel, and Wahlberg [59] constructed an ANN based automated cell counting system in encapsulated devices used for central nervous system drug delivery. Their system was based on modified NIH (US National Institute of Health) Image software running on a Power Macintosh desktop computer. In their system, ANN functioned as an intelligent filter that was used to scan regions of interest in the input image to produce an output image in which debris and other objects classified as "non-cells" had been rejected. The output image was then processed using standard image analysis tools available in NIH Image database and an object count for the image was thus obtained.

Zheng, Milthropen and Jones [60] applied a direct neural network approach for automated recognition of inflammatory cells from animal biomaterial implants in rabbit paravertebral muscle. There were 3072 input neurons in the input layer (3072 = 32 × 32 input image pixel matrix × 3 colors for each pixel), a varying number of neurons in the hidden layers, and 4 outputs in the output layer. The number of the

hidden neurons were chosen on the basis of trials with the training set to be as low as possible consistent with training. The original images (1280 × 1024) were cropped to include only the object of interest (one cell in each cropped frame) and then scaled down to form a 32 × 32 pixel frame containing each cell image. These scaled images were used as direct RGB inputs to the neural networks. The outputs from the neurons in the output layer of the neural network were presented in percentages, from 0% to 100%. With respect to the target outputs, "0" equals 0% and "1" equals 100%. The target outputs were defined as 0010, 0100, 1000 respectively for different types of cells. They became successful at classifying the cells from the pixel intensity information of the images using three layers and four layers of backpropagation neural networks with 97% and 98% correct recognition rates respectively.

2. *Other methods*

2.1 Other methods for hemoglobin level estimation:

Rendell et al. [61] determined the concentration of hemoglobin level in tissue using near infrared transmittance. They attempted to apply this technique in the finger using a unique handheld multiple wavelength spectrophotometer. They performed correlation analysis between the hemoglobin values and absorbance values and observed strong correlation between hemoglobin levels and the 14 wavelengths.

2.2 Other methods for red blood cell analysis:

Bacus and Weens [62] proposed a method of automated red cell analysis for the rapid classification of large numbers of red cells from individual blood specimens, and preliminarily tested on normal bloods and clinically proven cases of anemia and red cell disorders. According to their method, digital image processing techniques provided several features relating to shape and internal central pallor configurations of red cells. These features were used with a fully automated decision logic to rapidly provide a quantitative "red cell differential" analysis. The nine types of red cell disorders studied with their method were: (a) iron deficiency anemia, (b) the

anemia of chronic disease, (c) beta-thalassemia trait, (d) sickle cell anemia, (e) hemoglobin C disease, (f) intravascular hemolysis, (g) hereditary elliptocytosis, (h) hereditary spherocytosis, and (i) megaloblastic anemia due to folic acid deficiency.

Leon et al. [63] applied digital-image analysis technique for classification of sickle red blood cells in order to diagnose sickle cell anemia. Blood samples were collected from 24 patients with sickle cell anemia and 10 hematologically normal persons. One hundred fifty cells were analyzed from each sickle specimen, and 100 were analyzed from each nonsickle specimen. Expert observers classified each cell as normal (N), sickle (S), or other abnormal (A). Cells were analyzed with a custom, high-resolution image-analysis instrument. A total of 42 features including metric, optical density-derived, and textural features were extracted. Recursive partitioning analysis technique, based on the concept of finding the cut-points for the features that best separate different populations, was used to develop a cell classifier. Their results indicated that image analysis might be used for the automatic classification of red blood cells from patients with sickle cell anemia into the pattern classes of normal, sickle, and other abnormal.

Westerman and Bacus [64] studied red blood cell morphology in the peripheral blood of adults with sickle cell anemia to determine if changes occur during painful crises. They applied image processing of the cells with an automated system of red blood cell analysis. The authors concluded that the image processing approach with automated red blood cell analysis allowed for accurate assessment of all the morphologic groups of red blood cells in patients with sickle cell anemia and compared well with standard methods for measuring the concentration of irreversibly sickled cells.

Bacus et al. [65] applied digital image processing and pattern recognition techniques to determine the feasibility of a natural n-space subgrouping of normal and abnormal peripheral blood erythrocytes (red cells) into well separated categories. They introduced the concept of a quantitative "red cell differential" to

establish subpopulations of red cells for the diagnosis of anemia.

2.3 Other methods for white blood cell count and analysis:

Sinha and Ramakrishnan [66] proposed a technique for automating the differential count of white blood cells (WBC). Their proposed system considered color images of stained peripheral blood smears as input and identified the class of each of the white blood cells, which eventually determined the count of cells in each class.

Bikhet et al. [67] reported segmentation and classification of the five types of WBC in peripheral blood using gray images of blood smears. The features extracted were the areas of nucleus and cytoplasm, average gray level, circularity measure and the ratio of nucleus to cell area. They claimed classification accuracy of 90%, but they did not disclose the classifier.

Ongun et al. [68] worked on color smear images, containing both peripheral and immature cells. WBCs were segmented by morphological preprocessing combined with fuzzy patch labeling. Shape features were areas of cell and nucleus, ratio of nucleus to overall cell area, cell perimeter, compactness and boundary energy of the nucleus. Texture features included contrast, homogeneity and entropy derived from the gray-level co-occurrence matrix. Color histogram, mean and standard deviation of the color components in CIE-Lab domain, form the color features. This 57-dimensional feature set was used for classification using various classifiers, with a peak performance of 91% using SVM.

Park and Keller [69] reported segmentation and classification of white blood cells in bone marrow carried out on low resolution gray images.

Nilsson and Heyden [70] presented a model-based segmentation algorithm of leukocyte clusters that used interface propagation models to locate nuclear segments and their adherent cytoplasms. They succeeded in separating dense, complex clustered mixed-subclass leukocytes from peripheral blood and bone

marrow.

Kovalev et al. [71] proposed a method for automatic localization and feature extraction of white blood cells (WBCs) with color images to develop an efficient automated WBC counting system based on image analysis and recognition.

Kovalev et al. [72] proposed a method to develop an efficient automated leukocyte counter by using pattern recognition-based slide readers. Their proposed technique yielded correct segmentation of complex image scenes for blood smears prepared by ordinary manual staining methods in 99% of tested images.

Poon, Ward, and Palcic [73] proposed a technique for automatic detection and segmentation of nucleated cells in blood smears. Their method involved: 1) acquisition of spectral images; 2) preprocessing the acquired images; 3) detection of single and touching cells in the scene; 4) segmentation of the cells into nuclear and cytoplasmic regions; and 5) postprocessing of the segmented regions. Using the initial cell masks, touching nucleated cells were detected and separated. Simple features were then extracted. The intensity variations of the cells were then used to segment the nucleus from the cytoplasm. They concluded that the success rate in segmenting the nucleated cells was between 81 and 93%.

The published research has failed to supply accurate results for calculating blood parameters. In this research, we have been successful in obtaining near perfect result using ANN.

Chapter 3

Proposed Scheme & Simulation Results

3.1 Proposed Scheme

The main purpose of this thesis is to computerize the testing of blood parameters using microscopic image analysis and ANN algorithms. The proposed scheme counts RBC, and estimates hemoglobin value in human blood using the color information of the blood samples' images and a trained functional-link neural network (FNN). Three separate architectures have been implemented to perform the following tasks:

1. Counting RBC.
2. Estimating hemoglobin value.
3. Counting RBC and estimating hemoglobin value using a combined system.

We have used eleven variations of the backpropagation learning procedures to measure the simulation error of the test set for each architecture. Based on the minimum simulation error, an efficient backpropagation learning procedure has been chosen to implement the architecture of the FNN.

3.2 Implementation of the Proposed Scheme

In this section, we introduce the proposed scheme and explain the method of hemoglobin estimation and RBC counting. This scheme comprises of six steps as follows: (1) blood samples collection, (2) microscopic images capturing, (3) image processing, (4) image

analysis, (5) neural network training, and (6) testing and verification. A simple flow diagram of the proposed scheme is depicted in Fig. 3.1.

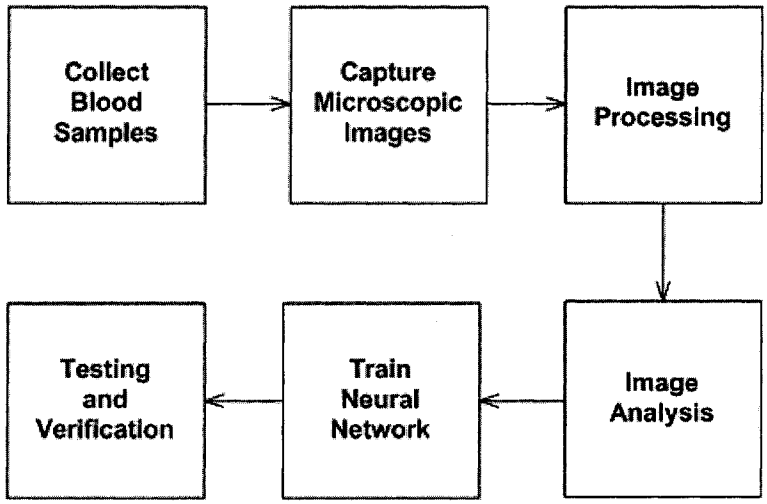


Figure 3.1: Flow diagram of the proposed scheme

3.2.1. The scheme's main steps:

The following is a brief description of the scheme's main steps:

Step 1: Blood samples collection

One thousand blood samples were obtained from Prince George Regional Hospital (PGRH) Laboratory over a period of three weeks. The automated system at the PGRH Laboratory was used to smear those blood samples on glass slides. The PGRH Laboratory also provided their corresponding complete blood count values of those blood samples examined by the CELL-DYN 3200 system for our reference and use in the research.

Step 2: Microscopic images capturing

We captured microscopic (at 10× magnification) color images of those blood samples using a digital camera mounted on a microscope at PGRH Laboratory. Eight hundred (800) blood sample images were used for training and the remaining 200 blood sample images were used for testing and comparison purposes.

Step 3: Image processing

We selected a segment of fixed window size of (256×256) pixels from each image for further processing and analysis. We opted to use this window size to avoid non-homogeneous regions due to the irregular smear of blood on the glass slides. As for choosing the size of the window, we experimented with different window sizes and found that (256×256) pixels gave similar results and it was faster to process. Fig. 3.2 shows images of the first nine blood samples in our test set.

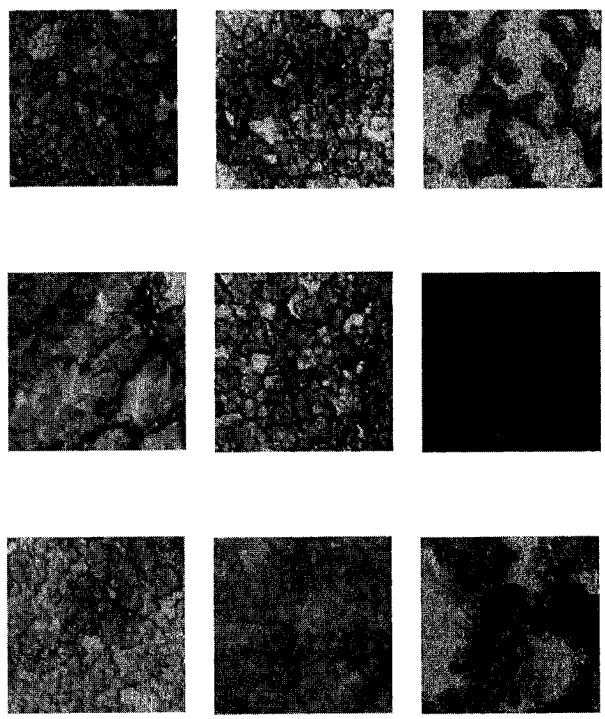


Figure 3.2: Test samples

Step 4: Image analysis

In this step, we calculated the color information for each image. Each image had three planes which represented the three primary colors (red, green & blue). We computed the average value for each plane using equations (6), (7), & (8) respectively, which gave the average value of red, green and blue color for each sample image.

$$R = \sum_{i,j=1}^{256} R_{i,j} \div (256 \times 256) \quad (6)$$

where R is the average value of red color of the sample image and $R_{i,j}$ is the value of the $(i,j)^{th}$ pixel.

$$G = \sum_{i,j=1}^{256} G_{i,j} \div (256 \times 256) \quad (7)$$

where G is the average value of green color of the sample image and $G_{i,j}$ is the value of the $(i,j)^{th}$ pixel.

$$B = \sum_{i,j=1}^{256} B_{i,j} \div (256 \times 256) \quad (8)$$

where B is the average value of blue color of the sample image and $B_{i,j}$ is the value of the $(i,j)^{th}$ pixel.

The average values of red, green, and blue colors for all sample images were stored in a matrix of size 3×800.

Step 5: Neural network training

In this step, we trained three FNN as follows:

i) Assemble the training data (Input matrix & target matrix).

ii) Create the network object.

iii) Train the network.

We started with training the network for the hemoglobin value, and then for the RBC count. We also trained the network for simultaneous hemoglobin value and RBC count.

Step 6: Testing and verification

In this step, we compare our results with those of PGRH Laboratory

3.2.2. Neural network training for the hemoglobin value:

Before training the network for the hemoglobin value, we studied the basic statistics of the 800 hemoglobin values in our sample set. The maximum and minimum values of the training samples were found to be 185 g/L and 57.30 g/L. We also plotted the histogram for the hemoglobin value of the total training samples as shown in Fig. 3.3.

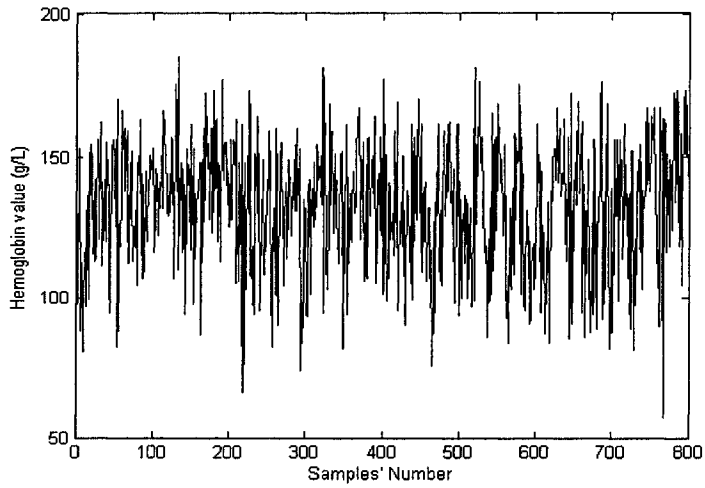


Figure 3.3: Histogram of blood samples' hemoglobin value

From the graph in Fig. 3.3, we found that only eight samples (i.e. 1% of the total samples) with a hemoglobin value greater than 175 g/L. Out of these eight samples, the hemoglobin values of seven samples were between 176 g/L and 178 g/L. Based on this, we introduced an upper cut-off threshold at 175 g/L. We assigned the value 175 g/L for any sample value higher than this cut-off threshold. Similarly, for hemoglobin values that were less than 75 g/L, we equated to 75 g/L and called the 75 g/L the lower cut-off threshold. This created a range of values between (75-175). The thresholds were adopted to facilitate the normalization process during training. Then the hemoglobin values were normalized within the range between 0 and 1.

Two approaches were adopted for assembling the training data. First, the color information (R, G, & B values) of the 800 blood sample images was fed as the input matrix and the corresponding hemoglobin values of those blood samples

measured using CELL-DYN 3200 System were provided as the target matrix into the neural network. The size of the input matrix was (3×800) and that of the target matrix was (1×800). The network architecture is shown in Fig. 3.4.

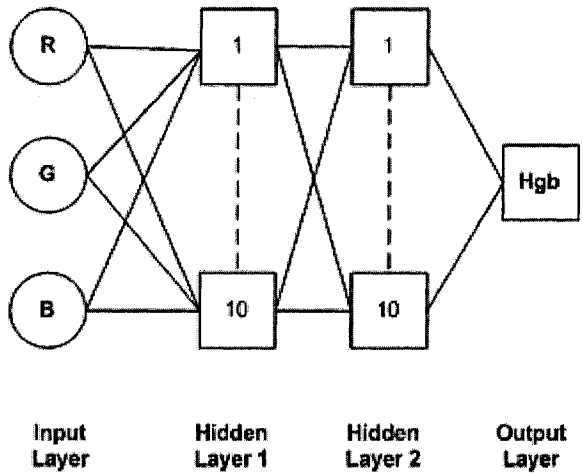


Figure 3.4: ANN architecture for training for hemoglobin value

During training, the network got stuck in a local minima of the error surface and failed to converge towards the performance function. The network, using original input and target vectors, did not learn well at all because of its property of ill-conditioning. Ill-conditioning in neural networks can be caused by the training data, the network's architecture, and/or its initial weights [74]. Large inputs cause ill-conditioning by leading to very small weights, large targets do so by leading to very large weights.

To overcome this problem, the second approach was applied. In this approach, the input and the target matrices were normalized between (0-1). The standard Gradient Descent backpropagation algorithm, using 2 hidden layers having 5 neurons in each layer, was applied for training the network. During training, the network converged towards very close to the performance function. Thus the network was successfully trained. The average percentage error of the simulated test samples was also calculated using equation (9) and found to be 3.59%.

$$APSE = \sum_{i=1}^{TTS} \left(\frac{|(LR_i - AR_i)| \times 100}{LR_i} \right) \div TTS \qquad (9)$$

where APSE is the average percentage of simulation error , LR is the PGRH Laboratory result, AR is the ANN result, and TTS is total number of test samples.

We have experimented with different architectures using 2, 3, 5 and 10 hidden layers and 5, 10, and 20 neurons for each layer. All results we got were not very satisfactory.

Finally, we extended our experiments using a FNN. Functional-links were developed by Yoh-Han Pao. This type of neural network expands the standard feed-forward backpropagation architecture to include nodes at the input layer providing the network with a more complete understanding of the input [75]. In this model, input variables are individually acted upon by appropriate functions. The overall effect is to provide the network with an enhanced representation of the input without adding any new information. The FNN increases the probability of learning the concept of the network in a comparatively easier way. In our case, the network was built with five neurons in the input layer, varying number of neurons (5 and 10) with 2 hidden layers, and one neuron in the output layer as shown in Fig. 3.5. For this network, the size of the input matrix was (5×800) and that of the target matrix was (1×800). Five neurons in the input layer are represented by five different functions of R, G, and B values of each sample image f1(R, G, B), f2(R, G, B), f3(R, G, B), f4(R, G, B), and f5(R, G, B). These five functions are (R, G, B, (R-G), and (R-B)) respectively. The FNN produced better results than the original 3-input neural network.

We have experimented with eleven variations of the backpropagation learning algorithm such as the Gradient Descent, Gradient Descent with Momentum, Gradient Descent with Variable Learning Rate, Resilient Backpropagation, Conjugate Gradient with Fletcher-Reeves Update, Conjugate Gradient with Polak-Ribiére Update, Conjugate Gradient with Powell-Beale Restarts, Scaled Conjugate Gradient, Quasi-Newton, One Step Secant, and Levenberg-Marquardt

algorithms. We found that the Conjugate Gradient with Fletcher-Reeves Update backpropagation algorithm is the most suitable for this set of data.

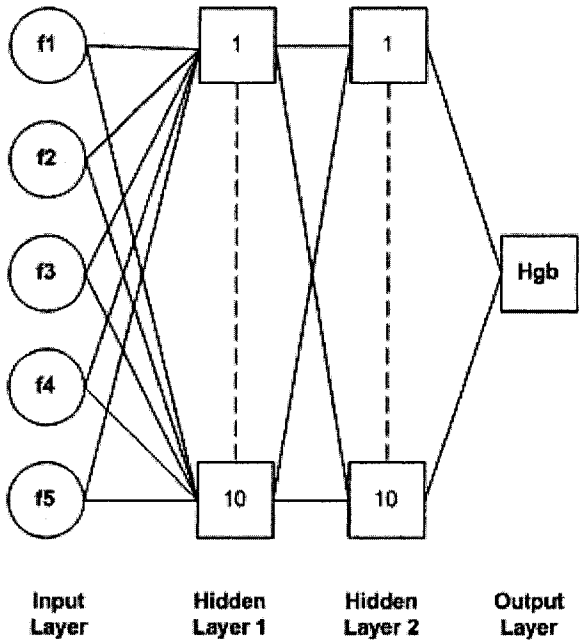


Figure 3.5: FNN architecture for training for hemoglobin value

3.2.3. Neural network training for the RBC count:

Before training the network for the RBC count, we studied the basic statistics of the 800 RBC count values in our sample set. The maximum and minimum values of the training samples were found to be 6.27×10^{12}/ L and 1.96×10^{12} / L. We also plotted the histogram for RBC count of the total training samples as shown in Fig. 3.6.

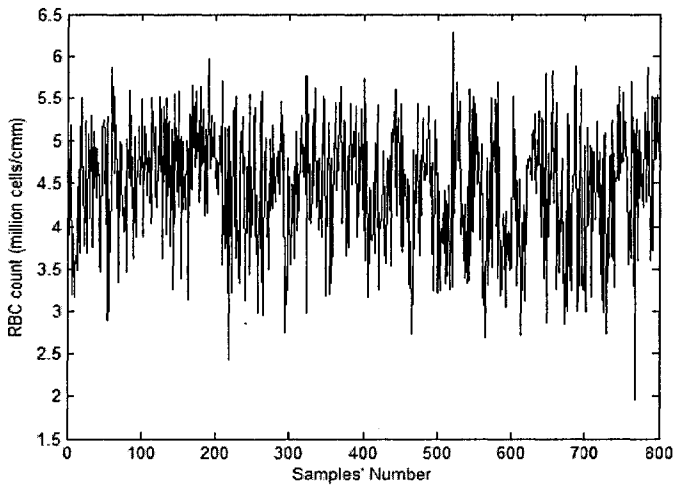


Figure 3.6: Histogram of blood samples' RBC count

These statistics show that only twelve values (i.e., 1.5 % of the total sample populations) are found outside the range $(2.75 - 5.75) \times 10^{12}$ / L. Out of these twelve samples, only one sample was below 2.00×10^{12} / L and two samples were above 6.00×10^{12} / L. The remaining nine samples were very close to the above mentioned range. This finding introduced the idea of upper and lower thresholds. Based on these thresholds and for efficient training, any value in the sample higher than the upper threshold was equated to 5.75×10^{12} and any value less than the lower threshold was equated to 2.75×10^{12} . The strategy of using the lower and upper cut-off thresholds did not affect the neural network training process or the simulation results, as only three samples in the training set were affected significantly by this strategy.

In our experimental approaches, we trained the neural network with different architectural set-ups as shown in Fig. 3.7.

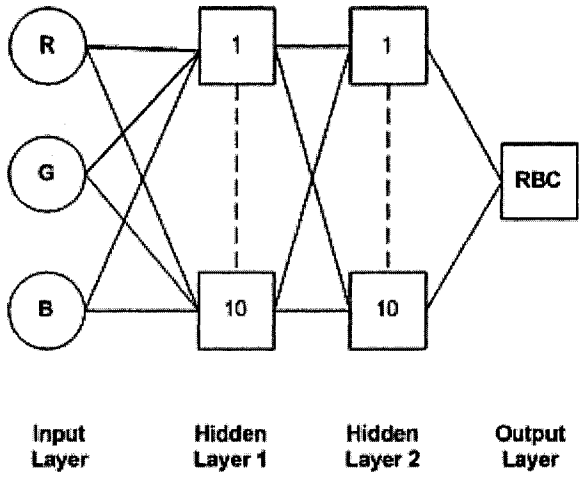


Figure 3.7: ANN architecture for training for RBC count

Two experimental approaches (original 3-input network and 5-input FNN) using different network architectures and parameter settings were carried out replicating the training of the hemoglobin value. Eleven different variations of the backpropagation algorithm were applied for training. The training time and performance function (mean square error) were noted. Then the trained network was simulated using the test samples' images. The average percentage error for the RBC values of all the test samples was also calculated to select the optimum neural network architecture.

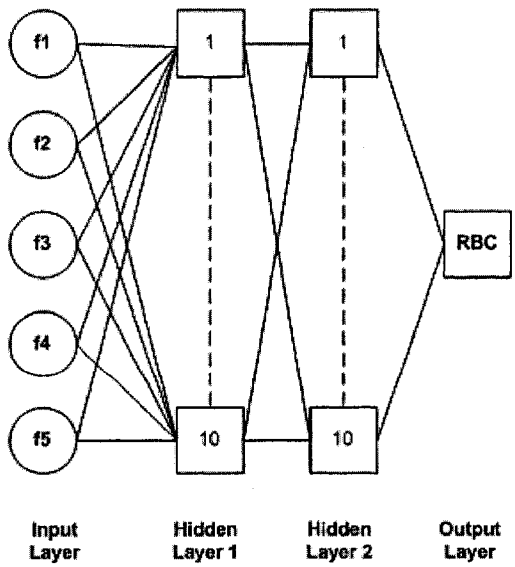


Figure 3.8: FNN architecture for training for RBC count

3.2.4. Neural network training for the hemoglobin value and the RBC count simultaneously:

The backpropagation algorithm can train a network to learn to recognize multiple concepts simultaneously. In this approach, the network was trained simultaneously for the hemoglobin value estimation and the RBC count as shown in Fig. 3.9. The network architecture was composed of three neurons in the input layer, varying number of neurons with varying hidden layers, and two neurons in the output layer representing the values for hemoglobin and RBC. Later, the network was trained using a 5-input FNN (as shown in Fig. 3.10). Several experiments were carried out to investigate the learning time, performance function, simulation errors for both hemoglobin and RBC using different network architectures and different backpropagation algorithms.

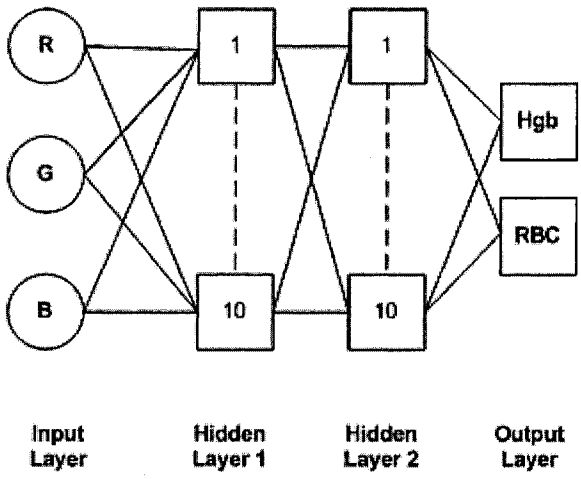


Figure 3.9: ANN architecture for simultaneous training for hemoglobin value and RBC count

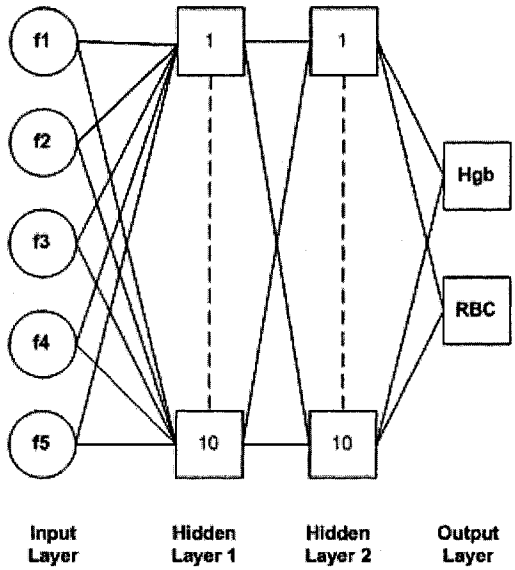


Figure 3.10: FNN architecture for simultaneous training for hemoglobin value and RBC count

3.2.5. Calculation of Mean Corpuscular Hemoglobin:

MCH was calculated from the measurement of the hemoglobin value and the RBC count using the formula: MCH= (HGB/RBC) pg.

3.3 Simulation Results

The experimental results of our proposed scheme using 2, 3, 5, and 10 hidden layers and 5, 10, and 20 neurons in each hidden layer are shown in Table 3.1 and 3.2 where APSE represents average percentage of simulation error.

No. of Hidden Layers	No. of Neurons (Each layer)	APSE for Hemoglobin	APSE for RBC
2	5	3.59	4.51
2	10	3.61	4.54
2	20	4.05	5.68
3	5	3.81	5.20
3	10	4.01	6.86
3	20	4.21	8.01
5	5	4.04	5.57
5	10	3.96	5.28
5	20	4.16	7.29
10	5	3.97	13.83
10	10	4.29	13.82
10	20	4.48	13.83

Table 3.1: Simulation results using 3-input ANN and Gradient Descent algorithm

No. of Hidden Layers	No. of Neurons (Each layer)	APSE for Hemoglobin	APSE for RBC
2	5	3.22	4.32
2	10	3.21	4.30
2	20	3.86	4.99
3	5	3.58	4.67
3	10	3.66	4.65
3	20	3.94	5.06
5	5	3.88	4.96
5	10	3.52	4.85
5	20	3.69	5.33
10	5	3.63	6.94
10	10	3.77	7.16
10	20	3.98	7.73

Table 3.2: Simulation results using 5-input FNN and Gradient Descent algorithm

In addition, we plotted eight graphs comparing the average percentage of simulation error (APSE) between the ANN and the FNN architectures. The APSE for the hemoglobin value using 5, 10, and 20 neurons in each of the 2, 3, 5, or 10 hidden layers is plotted in Fig. 3.11, 3.12, 3.13, and 3.14. Similarly, the APSE for the RBC count using 5, 10, and 20 neurons in each of the 2, 3, 5, or 10 hidden layers is plotted in Fig. 3.15, 3.16, 3.17, and 3.18.

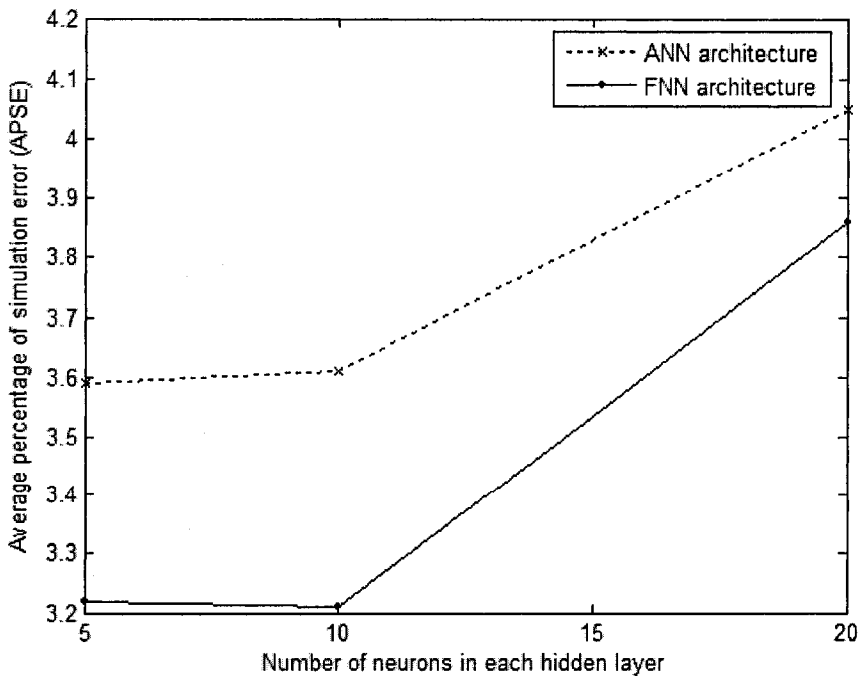


Figure 3.11: APSE for hemoglobin value using 2 hidden layers

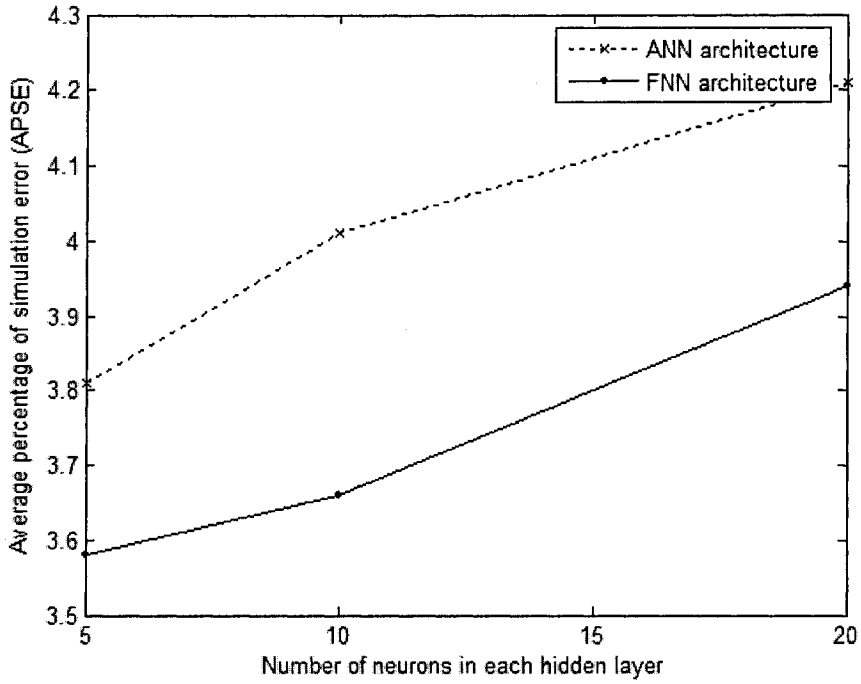


Figure 3.12: APSE for hemoglobin value using 3 hidden layers

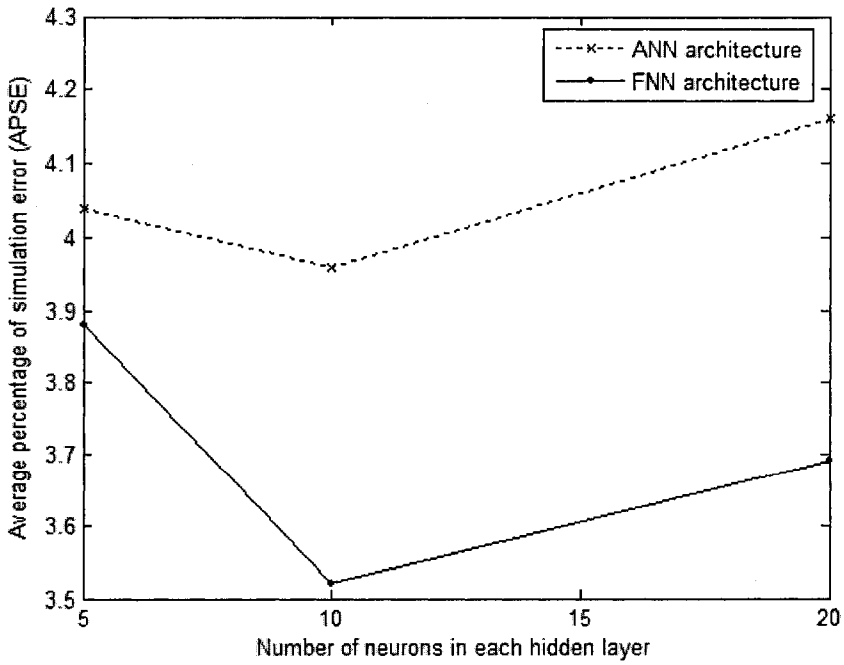


Figure 3.13: APSE for hemoglobin value using 5 hidden layers

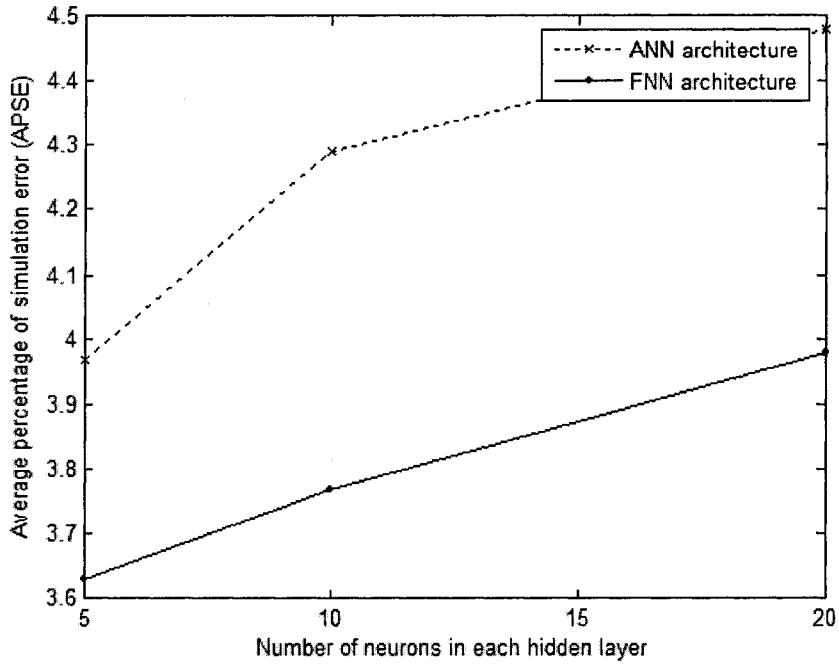


Figure 3.14: APSE for hemoglobin value using 10 hidden layers

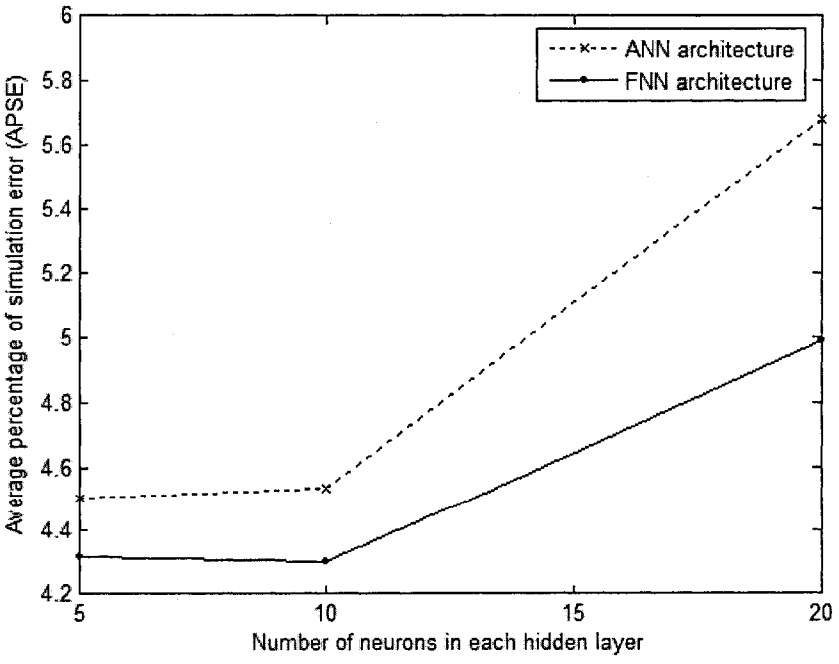


Figure 3.15: APSE for RBC count using 2 hidden layers

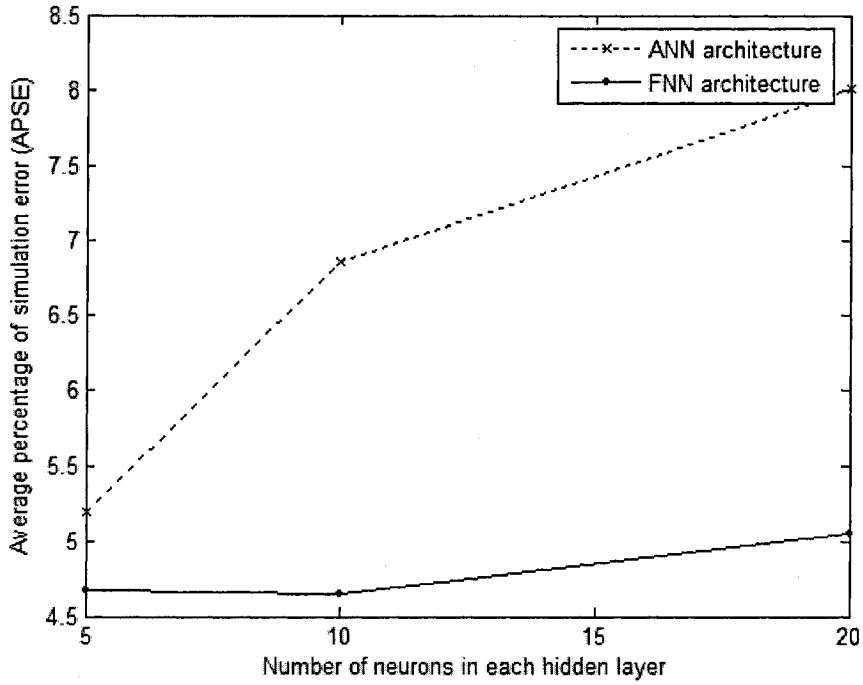


Figure 3.16: APSE for RBC count using 3 hidden layers

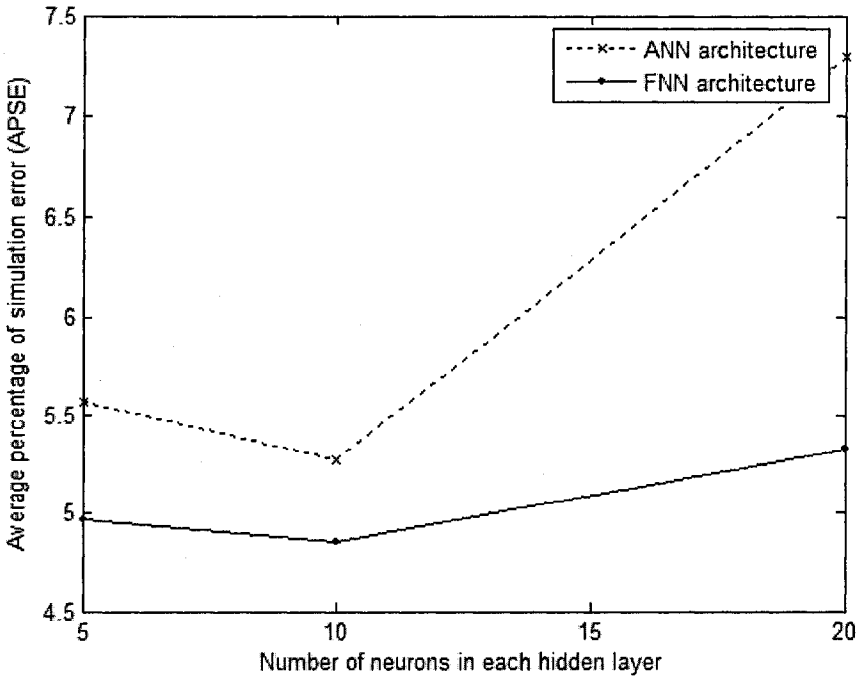


Figure 3.17: APSE for RBC count using 5 hidden layers

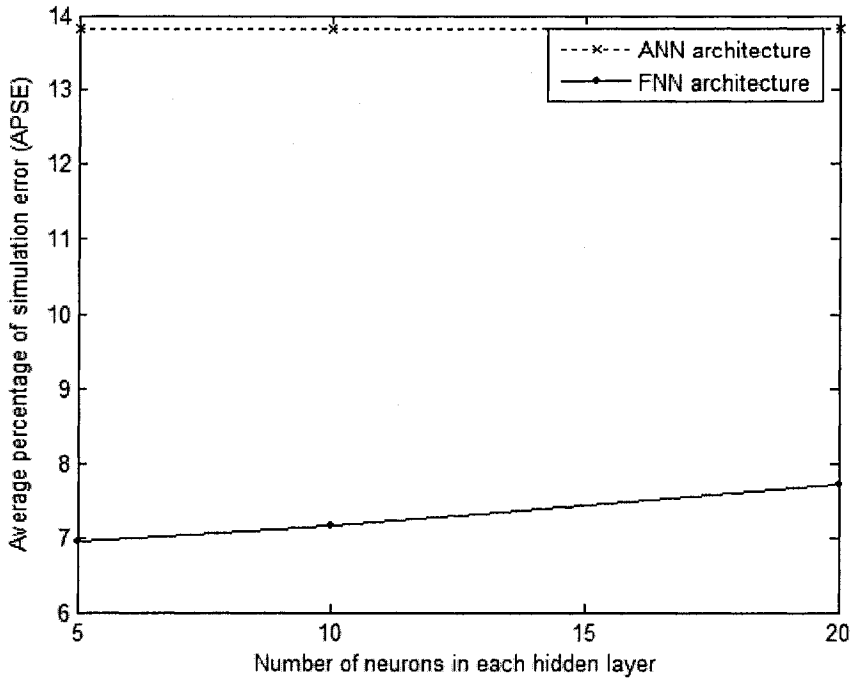


Figure 3.18: APSE for RBC count using 10 hidden layers

Fig. 3.11 - 3.18 show that the 5-input FNN produced better results than the 3-input ANN for both the hemoglobin value and the RBC count. It is also shown that the network architectures, using 2 hidden layers with 5 or 10 neurons in each hidden layer produced better results than all other architectures. We noticed an important phenomenon that the performance of a complex network (i.e. a network with more than 2 hidden layers and more than 10 neurons in each hidden layer) deteriorated due to the over-fitting of the training data.

Fig. 3.18 shows that the 3-input ANN, consisting of 10 hidden layers, generated a high percentage of simulation error for the RBC count. During training, the network becomes frozen just after 25,000 epochs. The reason behind that it has too many hidden layers which cause ill conditioning of the neural network. If the network is composed of many hidden layers, it may happen that one or more neurons in the hidden layers saturate as a result of having large incoming weights. As a result, the network produces a constant activation and the saturated hidden neurons act like a bias unit, making the output bias terms redundant. The phenomenon is that the networks without output biases are usually

ill-conditioned. An ill-conditioned network does not have the ability to generalize the relationship between the input and the target value.

In the final stage, we experimented with the FNN architecture and ten variations of the backpropagation algorithm using 2 hidden layers having 5 and 10 neurons in each hidden layer. Simulation results are summarized in Table 3.3 where GDM, GDVLR, RB, CGFVU, CGPRU, CGPBR, SCG, LM, QN, and QSS represent Gradient Descent with Momentum, Gradient Descent with Variable learning Rate, Resilient Backpropagation, Conjugate Gradient (Fletcher-Reeves Update), Conjugate Gradient (Polak-Ribiere Update), Conjugate Gradient (Powell-Beale Restarts), Scaled Conjugate Gradient, Levenberg-Marquardt, Quasi-Newton, and One Step Secant respectively.

Name of the Algorithm	No. of Neurons (Each layer)	APSE for Hemoglobin	APSE for RBC
GDM	5	3.28	4.33
GDM	10	3.17	4.35
GDVLR	5	3.26	4.38
GDVLR	10	3.34	4.43
RB	5	3.36	4.46
RB	10	3.41	4.41
CGFVU	5	3.12	4.33
CGFVU	10	3.06	4.28
CGPRU	5	3.23	4.31
CGPRU	10	3.12	4.29
CGPBR	5	3.25	4.30
CGPBR	10	3.14	4.34
SCG	5	3.38	4.51
SCG	10	3.44	4.48
LM	5	3.17	4.33
LM	10	3.20	4.31
QN	5	3.45	4.47
QN	10	173.5	200.56
OSS	5	3.37	4.49
OSS	10	3.48	4.44

Table 3.3: Simulation results using 5-input FNN with 2 hidden layers

From Table 3.3, we conclude that the FNN, using the Conjugate Gradient (Fletcher-Reeves Update) backpropagation algorithm obtained the best results. Moreover, this algorithm learns the concept (relationship between the input and the target) within few minutes, whereas other backpropagation algorithms take several hours to do so.

3.4 Final Implementation

Finally three automated architectures were implemented: (1) one for the hemoglobin value estimation, (2) one for the RBC count, (3) one combined system for estimating the hemoglobin value and counting the RBC. The FNN architecture was implemented using 2 hidden layers and 10 neurons of tan-sigmoid transfer function in each layer. The Conjugate Gradient backpropagation algorithm (Fletcher-Reeves Update) was applied as the learning rule.

The artificial neural network stores its knowledge in the weights and biases associated with each neuron. When the network object is created, its weights and biases are initialized. As the training epoch continues, the weights and biases are modified according to the corresponding artificial neural network algorithm to reduce the performance function (mean square error). After the completion of the training process, the final weights and biases are stored in the trained network. When the neural network is simulated using the test samples, these weights and biases determine the output value of the test samples. In our implemented artificial neural network model, the final network stores fifty (5 inputs × 10 neurons) weighting factors between the input layer and the first hidden layer neurons, one hundred (10 neurons × 10 neurons) weighting factors between the first hidden layer and the second hidden layer neurons, and ten (10 neurons × 1 output) weighting factors between the second hidden layer neurons and the output layer neuron for the 'single parameter' automated system. The network also stores ten bias values for the first hidden layer neurons, another ten bias values for the second hidden layer neurons, and one bias value for the output layer neuron for the 'single parameter' automated system. For the 'double parameter' automated system, there are twenty (10 neurons × 2 outputs) weighting factors between the second hidden layer and the output layer neurons and two bias values for the output layer neurons. These final weights and biases, given in the next section, are very important as these will be used for the hardware implementation of our proposed scheme. Using the final weights and biases and the following equations, the trained network is simulated against the test samples.

The neurons in the first hidden layer compute their outputs using the equation (10).

$$FH_k = T\left(\sum_{i=1}^{i=5} X_i \times a_{i,k} + u_k\right) \quad (10)$$

where FH_k is the output of the k^{th} neuron of the first hidden layer, X_i is the i^{th} input to the neural network, $a_{i,k}$ is the weighting factor between i^{th} input and the k^{th} neuron of the first hidden layer, u_k is the bias associated with the k^{th} neuron of the first hidden layer.
As the transfer function (T) of the first hidden layer neurons is tan-sigmoid, the output of the neurons will be calculated according to equation (4).

The neurons in the second hidden layer compute their outputs using the equation (11).

$$SH_l = T\left(\sum_{m=1}^{m=10} FH_m \times b_{m,l} + v_l\right) \quad (11)$$

where SH_l is the output of the l^{th} neuron of the second hidden layer, FH_m is the output of the m^{th} neuron of the first hidden layer, $b_{m,l}$ is the weighting factor between m^{th} neuron of the first hidden layer and the l^{th} neuron of the second hidden layer, v_l is the bias associated with the l^{th} neuron of the second hidden layer.
As the transfer function (T) of the second hidden layer neurons is tan-sigmoid, the output of the neurons will be calculated according to equation (4).

The output of the 'single parameter' architecture is computed using the equation (12).

$$Y = T\left(\sum_{l=1}^{l=10} SH_l \times c_l + w\right) \quad (12)$$

where Y is the output of the output layer neuron, SH_l is the output of the l^{th} neuron of the second hidden layer, c_l is the weighting factor between the l^{th} neuron of the second hidden layer and the neuron of the output layer, w is the bias associated with the output layer neuron.

The outputs of the 'double parameter' architecture are computed using the equation (13).

$$Y_n = T\left(\sum_{l=1}^{l=10} SH_l \times c_{l,n} + w_n\right) \quad (13)$$

where Y_n is the n^{th} output of the output layer neurons, SH_l is the output of the l^{th} neuron of the second hidden layer, $c_{l,n}$ is the weighting factor between the l^{th} neuron of the

second hidden layer and n^{th} neuron of the output layer, w_n is the bias associated with the n^{th} neuron of the output layer.

As the transfer function (T) of the output layer neuron is pure linear, the output of the neuron will be calculated according to equation (5).

3.5 Final Simulation Results

The final simulation results for the 'single parameter' architecture are listed below.

Hemoglobin value:
Overall average percentage of simulation error= 3.06% (200 samples)
Average percentage of simulation error for good samples (error less than 7 %) = 2.59% (188 samples)
Average percentage of simulation error for bad samples (error greater than 7 %) = 10.36% (12 samples)
Highest percentage of simulation error = 15.77% (1 sample)

Bad samples category:
Above 15 % error= 1 sample.
(10-15) % error= 5 samples.
(7- below 10) % error= 6 samples.

RBC count:
Overall average percentage of simulation error= 4.28% (200 samples)
Average percentage of simulation error for good samples (error less than 7 %) = 3.17% (168 samples)
Average percentage of simulation error for bad samples (error greater than 7 %) = 10.13% (32 samples)
Highest percentage of simulation error = 18.54 % (1 sample)

Bad samples category:
Above 15 % error= 5 samples.
(10-15) % error= 5 samples.
(7- below 10) % error= 22 samples.

The final simulation results for the 'double parameter' architecture are listed below.

Hemoglobin value:
Overall average percentage of simulation error= 3.21% (200 samples)
Average percentage of simulation error for good samples (error less than 7 %) = 2.75 % (188 samples)
Average percentage of simulation error for bad samples (error greater than 7 %) = 10.43% (12 samples)
Highest percentage of simulation error =15.96 % (1 sample)

Bad samples category:
Above 15 % error= 1 samples.
(10-15) % error= 4 samples.
(7- below 10) % error= 7 samples.

RBC count:
Overall average percentage of simulation error= 4.28% (200 samples)
Average percentage of simulation error for good samples (error less than 7 %) = 3.29% (173 samples)
Average percentage of simulation error for bad samples (error greater than 7 %) = 10.64% (27 samples)
Highest percentage of simulation error = 21.03 % (1 sample)

Bad samples category:
Above 15 % error= 5 samples.
(10-15) % error= 6 samples.
(7- below 10) % error= 16 samples.

The final simulation results of the first nine samples in our test set as compared with the PGRH Laboratory results are tabulated in Table 3.4, 3.5, and 3.6. In these tables the simulation error and percentage of simulation error are calculated using equations (14) and (15) respectively.

$$SE = |LR - AR| \tag{14}$$

where SE is the simulation error, LR is the PGRH Laboratory result, and AR is the ANN result.

$$PSE = \frac{|LR - AR|}{LR} \times 100 \tag{15}$$

where PSE is the percentage of simulation error , LR is the PGRH Laboratory result, and AR is the ANN result.

Blood Sample Number	Hemoglobin Value			
	Lab Result (g/L)	ANN Result (g/L)	Simulation Error	Percentage of Simulation Error
1	130	137.83	7.83	6.02
2	129	125.16	3.83	2.97
3	95.5	96.64	1.14	1.20
4	152	156.77	4.77	3.13
5	135	134.25	0.75	0.55
6	175	168.90	6.09	3.48
7	115	112.40	2.59	2.26
8	106	109.76	3.75	3.54
9	111	113.03	2.02	1.82

Table 3.4: Results for hemoglobin value

Blood Sample Number	RBC Count			
	Lab Result (10^{12}/L)	ANN Result (10^{12}/L)	Simulation Error	Percentage of Simulation Error
1	4.58	4.59	0.01	0.36
2	4.22	4.26	0.04	1.05
3	3.34	3.35	0.01	0.35
4	5.13	5.15	0.02	0.47
5	4.28	4.50	0.22	5.35
6	5.75	5.48	0.26	4.66
7	3.8	3.82	0.02	0.58
8	3.75	3.61	0.13	3.72
9	3.88	3.77	0.10	2.80

Table 3.5: Results for RBC count

Blood Sample Number	MCH Value			
	Lab Result (pg)	ANN Result (pg)	Simulation Error	Percentage of Simulation Error
1	28.38	30.02	1.64	5.77
2	30.57	29.38	1.19	3.89
3	28.59	28.84	0.25	0.86
4	29.62	30.44	0.82	2.76
5	31.54	29.83	1.71	5.42
6	30.43	30.82	0.39	1.28
7	30.26	29.42	0.84	2.77
8	28.27	30.40	2.13	7.53
9	28.60	29.98	1.38	4.82

Table 3.6: Results for MCH

3.6 Final Weights & Biases of the Trained FNN Model

There are three separate trained FNN models: (1) the 'single parameter' architecture for hemoglobin value estimation, (2) the 'single parameter' architecture for RBC count, (3) the 'double parameter' architecture for hemoglobin value estimation and RBC count.

1. Trained FNN Model for Hemoglobin Value Estimation

Tables 3.7, 3.8, 3.9, & 3.10 show the biases attached with the neurons, weights between the inputs and first hidden layer neurons, weights between the first hidden layer neurons and the second hidden layer neurons, weights between the second hidden neurons and the output layer neuron respectively.

Neurons	First Hidden Layer Biases	Second Hidden Layer Biases	Output Layer Bias
1st Neuron	4.638264	1.763191	0.382504
2nd Neuron	0.306209	-1.33295	
3rd Neuron	-3.08738	-0.80965	
4th Neuron	2.261967	0.921083	
5th Neuron	-2.22496	-0.12133	
6th Neuron	-2.17775	0.326849	
7th Neuron	4.294092	-0.7561	
8th Neuron	-0.33292	-1.17417	
9th Neuron	3.234807	-1.38026	
10th Neuron	-4.91271	-1.78456	

Table 3.7: Biases attached with the neurons

$a_{1,i}$ (i=1,..., 10)	$a_{2,i}$ (i=1,..., 10)	$a_{3,i}$ (i=1,..., 10)	$a_{4,i}$ (i=1,..., 10)	$a_{5,i}$ (i=1,..., 10)
-2.26365	-3.70325	4.352837	-1.6791	-2.69967
4.809925	-2.75595	-3.92982	-2.89487	-0.53066
0.101027	2.585309	3.884149	3.41469	3.775118
-3.82541	-3.65783	2.941462	3.317458	1.112025
2.525966	-1.87331	2.122619	-3.14333	4.708794
0.010899	-2.62011	4.998588	-1.02758	3.286367
-3.8081	-4.88011	1.136672	-3.80885	-0.11816
3.80353	-5.11878	3.088949	-0.55258	1.637089
-5.06999	-0.79479	-1.43214	3.202639	-3.68916
5.987371	3.208142	3.795097	-0.92612	0.492261

Table 3.8: Weights between inputs and first hidden layer neurons

$b_{1,j}$ (j=1,..., 10)	$b_{2,j}$ (j=1,..., 10)	$b_{3,j}$ (j=1,..., 10)	$b_{4,j}$ (j=1,..., 10)	$b_{5,j}$ (j=1,..., 10)	$b_{6,j}$ (j=1,..., 10)	$b_{7,j}$ (j=1,..., 10)	$b_{8,j}$ (j=1,..., 10)	$b_{9,j}$ (j=1,..., 10)	$b_{10,j}$ (j=1,..., 10)
-0.08976	-0.62817	-0.43564	-0.28575	0.660274	-0.47335	-0.56955	0.824402	-0.51398	0.928971
0.896767	-0.52377	-0.24634	-0.15039	0.639792	0.596121	-0.95749	0.326852	-0.33804	0.092085
0.784647	1.220675	0.444048	0.464145	0.642426	-0.06645	0.467498	0.758524	-1.09687	-0.48036
0.06615	-0.6978	0.484286	-0.40889	0.147227	-1.27438	0.337571	0.510156	-0.11708	-0.34113
0.503276	-0.23682	0.216666	-0.00245	0.06484	0.146138	-0.6559	0.465942	0.792963	-1.14924
0.715325	-0.53727	-0.18046	0.032843	0.394419	0.112035	0.636388	-0.18156	-1.3667	-0.09966
-1.04268	0.68524	0.597767	0.397945	-0.59441	-0.47876	-0.20624	0.923624	-0.46019	-0.2148
-0.74175	-0.52298	-0.57908	0.063921	-0.61887	-0.37266	-0.66208	-0.62166	-0.26916	-0.28737
-0.02261	0.042729	-1.03996	0.01036	-0.19238	1.08356	-0.33509	0.101566	0.78418	-0.25278
-1.15535	-0.13128	-0.31765	-0.04496	0.154461	-0.57802	-0.61592	0.099301	-0.88784	-0.30199

Table 3.9: Weights between first hidden layer neurons and second hidden layer neurons

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
0.628881	-0.02122	0.34757	-0.5429	0.025158	-0.14408	-0.06139	-0.37528	0.307522	-0.19153

Table 3.10: Weights between second hidden layer neurons and output layer neuron

2. Trained FNN Model for RBC Count

Tables 3.11, 3.12, 3.13, & 3.14 show the biases attached with the neurons, weights between the inputs and the first hidden layer neurons, weights between the first hidden layer neurons and the second hidden layer neurons, weights between the second hidden neuron and the output layer neuron respectively.

Neurons	First Hidden Layer Biases	Second Hidden Layer Biases	Output Layer Bias
1st Neuron	-2.21959	-0.60934	3.803615
2nd Neuron	-2.73855	-1.45663	
3rd Neuron	4.217894	2.490757	
4th Neuron	-3.66979	1.700343	
5th Neuron	-2.85741	-2.81692	
6th Neuron	5.662801	0.968138	
7th Neuron	8.135576	0.983799	
8th Neuron	-4.02558	-1.18023	
9th Neuron	6.696273	-0.05179	
10th Neuron	1.916216	1.811811	

Table 3.11: Biases attached with the neurons

$a_{1,i}$ (i=1,..., 10)	$a_{2,i}$ (i=1,..., 10)	$a_{3,i}$ (i=1,..., 10)	$a_{4,i}$ (i=1,..., 10)	$a_{5,i}$ (i=1,..., 10)
4.238046	0.844726	-3.97492	3.606879	-2.05631
1.826118	-2.04603	3.491623	1.75223	-9.03875
-2.39279	-7.94093	4.181038	-1.01606	-7.2944
3.975278	5.450225	-4.31725	1.700496	6.264592
-0.68068	-0.72585	0.306684	-2.40063	-6.55431
-2.84602	-1.11586	1.081974	4.133305	-2.57669
-2.10583	-0.76175	-1.33157	-4.09384	-3.72198
-2.31832	4.29635	-2.50665	0.424751	3.602058
1.96582	-2.54588	-2.76533	0.758993	-4.16117
-3.76076	-0.50849	1.713654	-5.24521	4.754858

Table 3.12: Weights between inputs and first hidden layer neurons

$b_{1,j}$ (j=1,..., 10)	$b_{2,j}$ (j=1,..., 10)	$b_{3,j}$ (j=1,..., 10)	$b_{4,j}$ (j=1,..., 10)	$b_{5,j}$ (j=1,..., 10)	$b_{6,j}$ (j=1,..., 10)	$b_{7,j}$ (j=1,..., 10)	$b_{8,j}$ (j=1,..., 10)	$b_{9,j}$ (j=1,..., 10)	$b_{10,j}$ (j=1,..., 10)
-2.10352	-1.05169	-4.13515	-4.79288	4.040381	-0.41064	1.692595	-0.97736	0.365815	-0.98958
0.791332	0.969464	-0.16784	-0.36251	0.477193	0.366661	-0.32628	0.466022	-0.86761	0.529735
-2.08306	-2.02508	-1.31401	0.054656	0.466451	0.758659	0.601134	-1.06197	1.747207	-2.14393
-1.71285	-0.64493	-0.61469	-0.01267	-1.00274	0.312789	0.322804	-0.32021	1.82385	-1.7553
2.743051	2.622672	4.220447	2.796864	-4.54212	2.651775	-1.62247	2.739278	-2.61495	3.49377
-1.16743	-1.20743	-0.79728	-0.07732	1.22093	0.147811	-0.52674	-1.87379	1.204589	-0.53166
-1.71475	-2.29731	-1.64248	-2.21919	0.170063	0.096207	0.58122	-1.03185	0.763327	-1.27622
-0.062	0.348651	0.632726	0.088787	0.234498	-0.24771	-0.58795	1.300122	-0.21687	1.045657
-1.34083	-1.49463	-1.7897	1.023245	0.601068	-1.26564	0.028784	-1.97059	1.06662	-1.64621
1.619	3.059543	1.736877	-3.40036	0.686469	-0.87869	0.607983	-0.86496	-0.38214	-5.85682

Table 3.13: Weights between first hidden layer neurons and second hidden layer neurons

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
-11.8366	-3.00722	2.299175	3.784433	-0.99436	1.440265	0.394979	-3.67301	1.385919	7.743089

Table 3.14: Weights between second hidden layer neurons and output layer neuron

3. Trained FNN Model for Hemoglobin Value Estimation and RBC Count using a Combined Architecture

Tables 3.15, 3.16, 3.17, & 3.18 show the biases attached with the neurons, weights between the inputs and the first hidden layer neurons, weights between the first hidden layer neurons and the second hidden layer neurons, weights between the second hidden and the output layer neurons respectively.

Neurons	First Hidden Layer Biases	Second Hidden Layer Biases	Output Layer Biases
1st Neuron	-7.00645	-2.96398	0.871681
2nd Neuron	-2.08689	-3.5597	5.517893
3rd Neuron	0.205907	2.400184	
4th Neuron	4.238804	-0.55618	
5th Neuron	-2.08475	-4.41242	
6th Neuron	4.306709	2.406877	
7th Neuron	-3.55978	-1.48701	
8th Neuron	1.263578	2.231104	
9th Neuron	0.764804	-0.76601	
10th Neuron	4.069355	0.262834	

Table 3.15: Biases attached with the neurons

$a_{1,i}$ (i=1,..., 10)	$a_{2,i}$ (i=1,..., 10)	$a_{3,i}$ (i=1,..., 10)	$a_{4,i}$ (i=1,..., 10)	$a_{5,i}$ (i=1,..., 10)
-0.20099	4.637237	-1.77694	0.788687	3.596626
-1.92658	-7.44203	-3.34647	-3.33728	-3.28944
-0.58236	3.345127	3.429597	4.384336	5.218329
-6.86785	-1.50232	2.846274	1.147276	-9.04909
-6.2262	2.882708	2.87716	-1.28733	-1.45233
-2.84694	-7.05369	3.804615	-2.39193	-6.78588
7.351831	-3.38968	-3.38374	-0.46389	8.932537
-5.04775	-1.57312	-1.60295	3.718551	-3.87075
-8.24261	-2.15845	-1.40853	-0.94052	2.865561
-7.8202	-2.24522	-0.84376	-2.83133	-9.9923

Table 3.16: Weights between inputs and first hidden layer neurons

$b_{1,j}$ (j=1,..., 10)	$b_{2,j}$ (j=1,..., 10)	$b_{3,j}$ (j=1,..., 10)	$b_{4,j}$ (j=1,..., 10)	$b_{5,j}$ (j=1,..., 10)	$b_{6,j}$ (j=1,..., 10)	$b_{7,j}$ (j=1,..., 10)	$b_{8,j}$ (j=1,..., 10)	$b_{9,j}$ (j=1,..., 10)	$b_{10,j}$ (j=1,..., 10)
1.407515	1.110944	-0.73086	-1.1003	0.152292	0.236573	-0.11318	0.806933	1.832485	0.117548
3.082052	0.459998	0.195221	-0.49017	1.647219	-2.49558	-1.69966	2.220236	1.859827	0.202399
-1.63277	-0.03071	-0.7528	1.035608	-2.58428	1.032777	0.383916	-0.65138	-1.5062	-0.77701
0.015791	-1.47355	1.326478	4.131184	-0.36738	-4.08103	-0.50589	-0.64146	-0.91514	4.504985
4.516253	-0.92426	1.946027	-0.71507	2.875905	-3.31543	-2.117	1.796231	1.477686	1.671972
-1.44427	-4.78042	1.93869	3.925607	-0.87716	2.955972	-9.58469	-2.72503	-3.31063	0.488492
0.487284	0.109843	0.010332	0.256829	0.317712	-0.81804	-0.1613	1.364355	0.257603	1.433076
-0.85498	-1.44538	-0.44378	0.12999	0.114988	1.382751	0.629846	-0.5571	-0.78407	-0.18756
3.02984	0.837173	-0.11046	-1.64583	1.400251	-1.05848	-0.63909	2.215051	1.315555	0.032499
-2.37521	-0.90295	1.151627	1.26768	0.000256	0.54614	2.124502	-2.50973	-2.3804	-1.20161

Table 3.17: Weights between first hidden layer neurons and second hidden layer neurons

$c_{1,k}$ (k=1, 2)	$c_{2,k}$ (k=1, 2)	$c_{3,k}$ (k=1, 2)	$c_{4,k}$ (k=1, 2)	$c_{5,k}$ (k=1, 2)	$c_{6,k}$ (k=1, 2)	$c_{7,k}$ (k=1, 2)	$c_{8,k}$ (k=1, 2)	$c_{9,k}$ (k=1, 2)	$c_{10,k}$ (k=1, 2)
0.580311	-0.10319	0.731543	-0.20162	0.049514	-0.18768	0.836882	-0.02063	-0.18372	0.484135
-5.40991	-6.0272	5.356109	-11.3146	0.247634	-8.87749	-0.41971	0.271372	0.493517	0.577772

Table 3.18: Weights between second hidden layer neurons and output layer neurons

Chapter 4

Conclusion & Future Directions

4.1 Conclusion

At present, artificial neural networks are employed in a significant number of applications and research. Many of the artificial neural networks based applications are being implemented in hardware especially in the medical arena. In this thesis, we propose a functional-link neural network based automated system for the assessment of erythrocyte parameters. Our proposed scheme can estimate hemoglobin value and count RBC in human blood which can aid physicians to diagnose several diseases. This scheme can be implemented in hardware with minimal cost.

To ensure near optimal results, we have experimented with eleven variations of the backpropagation learning algorithm using different network architectures. Conjugate Gradient (Fletcher-Reeves Update) backpropagation algorithm has been chosen to implement the proposed scheme as it gave the best results.

The proposed 5-input FNN produced excellent results for the hemoglobin value that are much higher than any published results. This scheme offers quick and very efficient results for diagnosing erythrocyte disorders such as anemia and polycythemia based on hemoglobin, RBC count, and MCH values. We achieved very promising results for hemoglobin with 94% of the samples less than 7% tolerance compared to the laboratory results. For the remaining 6%, which are outside the specified range, the tolerance was

between 7% and 15%. For RBC, we also achieved promising results for 84% of the samples with results tolerance of less than 7% compared to the laboratory results. There are no studies in the literature to compare our results with in regarding to RBC which makes our results as a new reference for researchers in this field.

4.2 Future Directions

There are many directions of extending the current research work presented in this thesis by capturing microscopic digital images of the blood samples at 100x magnification. Some possible directions of future work are outlined below:

- Several diseases such as sickle-cell anemia, which are related to the abnormality of the red blood cells, can be diagnosed. Sickle cell anemia is a serious disease in which red blood cells are abnormally shaped. Normal red blood cells are smooth and round like a doughnut without a hole, which move easily through blood vessels to carry oxygen to all parts of the body. In sickle cell anemia, red blood cells become crescent-shaped. These "sickle cells" are hard and sticky and they don't move easily through blood vessels. They tend to get stuck and block the flow of blood to the limbs and organs. This can cause pain, organ damage, and a low blood count [76].

  In order to diagnose sickle-cell anemia, the important features of the sickle red blood cells need to be extracted and applied to the artificial neural network as the input matrix. After sufficient training, the network will be able to classify the sickle cells from normal red blood cells and diagnose sickle cell anemia.
- The extension of the thesis work can be applied to diagnose other diseases such as spherocytosis. Spherocytosis is a genetic disease that causes a defect in the red blood cell's cytoskeleton, causing the RBC to be small and sphere-shaped [77]. Like the sickle-cell anemia, the significant features for spherocytosis affected red blood cells will be extracted and exploited by the artificial neural network algorithms.

- Thus the automated system proposed in this thesis can be extended as an application of computer vision. From the microscopic images captured at 100x magnification, the automated system will be intelligent enough not only to count the red blood cells but also to interpret the condition of the red blood cells.
- The proposed automated system can also be extended for counting and differentiating white blood cells (WBC) by capturing the images and extracting the corresponding features of different types of WBC, and then exploiting the extracted features for training the neural network.
- The extension of the proposed automated system may also be capable of differentiating between the normal and abnormal types of white blood cells. Thus the automated system can successfully diagnose several diseases of white blood cells such as leukemia.
- The proposed automated system can also be extended for counting platelets in human blood and diagnosing several diseases of platelet.

Bibliography

[1] Chernecky, Cynthia C. and Barbara Berger, Laboratory Tests and Diagnostic Procedures, 3rd Edition, Philadelphia, PA: W. B. Saunders, 2001.

[2] L. Van Hove, T. Schisano, L. Brace, "Anemia diagnosis, classification, and monitoring using Cell-Dyn technology reviewed for the new millennium", Laboratory Hematology, vol. 6, pp. 93-108, 2000.

[3] Henry, John B., Clinical Diagnosis and Management by Laboratory Methods, Philadelphia: W. B. Saunders, 2001.

[4] John P. Greer, John Foerster, John N. Lukens, George M. Rodgers, Frixos Paraskevas, Bertil Glader, Wintrobe's Clinical hematology, 11th Edition, Lippincott Williams & Wilkins, Philadelphia, New York, 2003.

[5] E. Anne Stiene Martin, Cheryl A. Lotspeich Steininger, and John A. Koepke, Clinical Hematology- Principles, Procedures, Correlation, 2nd Edition, Lippincott, Philadephia, New York, 1998.

[6] CELL-DYN 3200 System Training Guide.

[7] http://www.nlm.nih.gov/medlineplus/ency/article/003644.htm

[8] http://www.nlm.nih.gov/medlineplus/ency/article/001114.htm

[9] http://www.nlm.nih.gov/medlineplus/ency/article/000129.htm

[10] http://www.nlm.nih.gov/medlineplus/ency/article/000069.htm

[11] http://www.leukemia-lymphoma.org.

[12] http://www.nlm.nih.gov/medlineplus/ency/article /000589.htm.

[13] http://www.nlm.nih.gov/medlineplus/ency/article /000560.htm.

[14] http://www.nlm.nih.gov/medlineplus/ency/article/001303.htm

[15] http://www.nlm.nih.gov/medlineplus/tutorials/leukemia/htm/_no_75_no_0.htm

[16] http://www.nlm.nih.gov/medlineplus/ency/article/000583.htm

[17] http://health.allrefer.com/health/rbc-indices-info.html

[18] Toshinori Munakata, Fundamentals of the New Artificial Intelligence: Beyond Traditional Paradigms, 1st Edition, Springer-Verlag, New York, 1998.

[19] Haykin, S., Neural Networks: A Comprehensive Foundation, NY: Macmillan, 1994.

[20] Nigrin, A., Neural Networks for Pattern Recognition, Cambridge, MA: The MIT Press, 1993.

[21] Staurt J. Russel and Peter Norving, Artificial Intelligence: A Modern Approach, Pearson Education (Singapore) Pte. Ltd., Indian Branch, India, 2002.

[22] H. Demuth and M. Beale, Neural Network Toolbox (user's guide) http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/nnet.shtml

[23] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representation by error backpropagation", In Parallel distributed processing, vol. 1, pp. 318-362, MIT Press, Cambridge, MA, 1986.

[24] E.D. Sontag and H.J. Sussman, "Backpropagation can give rise to spurious local minima even for networks without hidden layers", Complex Systems, vol. 3, pp. 91-106, 1989.

[25] P.Auer, M. Herbster, and M.K. Warmuth, "Exponentially many local minima for single neuron", Advances in Neural Information Processing Systems, vol. 8, pp. 316-322, Cambridge, MA, 1996.

[26] http://www.willamette.edu/~gorr/classes/cs449/precond.html

[27] R. Battiti, "Accelerated backpropagation learning: two optimization methods", Complex Systems, vol. 3, pp. 331–342, 1989.

[28] T.P. Vogl, J.K. Mangis, J.K. Rigler, W.T. Zink and D.L. Alkon, "Accelerating the Convergence of the Back–propagation Method", Biological Cybernetics, vol. 59, pp. 257–263,1988.

[29] L.W. Chan and F. Fallside (1987), "An adaptive training algorithm for back–propagation networks", Computers Speech and Language, vol. 2, pp. 205–218, 1987.

[30] R.A. Jacobs, "Increased rates of convergence through learning rate adaptation", Neural Networks, vol. 1, pp. 295–307, 1988.

[31] M. Pfister and R. Rojas, "Speeding–up backpropagation – A comparison of orthogonal techniques", In Proceedings of the Joint Conference on Neural Networks, pp. 517–523, Nagoya, Japan, 1993.

[32] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the Rprop algorithm", In Proceedings of the IEEE International Conference on Neural Networks, pp. 586–591. San Francisco, 1993.

[33] F. Silva and L. Almeida (1990), "Acceleration techniques for the back–propagation algorithm", Lecture Notes in Computer Science, vol. 412, pp. 110–119. Berlin: Springer–Verlag, 1990.

[34] Magnus R. Hestens, "Iterative methods for solving linear equations", Journal of Optimization Theory and Applications 11, vol. 4, pp. 323-334, 1973. (Originally published in 1951 as NAML Report No. 52-9, National Bureau of Standards, Washington, D.C.

[35] Eduard Stiefel, "Uber einige Methoden der Relaxationsrechnung", Zeitschrift fur Angewandte Mathematik and Physik 3, vol. 1, pp. 1-33, 1952.

[36] http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

[37] R. Fletcher and C.M. Reeves, "Function minimization by conjugate gradients", Computer Journal, vol.7, pp. 149-154, 1964.

[38] W.C. Davidson, "Variable metric method for minimization", Tech. Report ANL-5990, Argonne National laboratory, Argonne, Illionis, 1959.

[39] R. Fletcher and M.J.D. Powell, "A rapidly convergent descent method for minimization", Computer Journal, Vol. 6, pp. 163-168, 1963.

[40] Martin Fodslette Moller, "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks*, vol. 6, pp. 525--533, 1993.

[41] Yap K. H.,Guan L., "A recursive soft-decision PSF and neural network approach to adaptive blind image regularization", Proceedings of International Conference on Image Processing, vol. 3, pp. 813-816, 2000.

[42] Yap K. H.,Guan L, " A recursive approach to joint image restoration and compensated blur identification", Proceedings of the Signal Processing Society Workshop, vol. 2, pp. 567-575, 2000.

[43] Serele C. Z., Gwyn Q. H. J., Boisvert J.B., Pattey E., McLaughlin N., Daoust G., "Corn yield prediction with artificial neural network using airborne remote sensing and topographic data", IEEE International Proceedings of IGARSS, vol. 1, pp. 384-386, 2000.

[44] Lizi Zhang, Jinping Kang, Xianshu Lin, Yinghui Xu, "Application of neural networks trained with an improved conjugate gradient algorithm to the turbine fast valving control", Proceedings of the International Conference on PowerCon, vol. 3, pp. 1679-1682, 2000.

[45] Kornel Papik, Bela Molnar, Rainer Schaefer, Zalan Dombovari, Zsolt Tulassy, and Janos Feher, "Application of neural networks in medicine – a review", Medical Science Monitor, vol. 4(3), pp. 538-546, 1998.

[46] Baxt WG, "Use of an artificial neural network for the diagnosis of myocardial infarction", Ann Intern Med., vol. 115(11), pp. 843-848, 1991.

[47] B.G. Batchelor, Practical approach to pattern classification, London: Plenum, 1974.

[48] M.L. Astion and P. Wilding, "The application of backpropagation neural networks to problems in pathology and laboratory medicine", Archives of Pathology and Laboratory Medicine, vol. 116, pp. 995-1001, October, 1992

[49] Hamamoto I, Okada S, Hashimoto T, Wakabayashi H, Maeba T, Maeta H, "Prediction of the early prognosis of the hepatectomized patient with hepatocellular carcinoma with a neural network", Journal of Computers in Biology and Medicine, vol. 25(1), pp. 49-59, 1995.

[50] R. Marabini and J.M. Carazo, "Pattern recognition and classification of images of biological macromolecules using artificial neural networks", Biophysical Journal, vol. 66, no. 6, pp. 1804-1814, June 1994.

[51] P. Belhomme, A. Elmoatz, P. Herlin and D. Bloyet, "Generalized region growing operator with optimal scanning: application to segmentation of breast cancer images", Journal of Microscopy, vol. 186, pp. 41-50, April 1997.

[52] G. Diaz, C. Cappai, M.D. Setzu and A. Diana, "Nuclear pattern recognition by two-parameter texture analysis", Computer Methods & Programs in Biomedicine, vol. 49, pp. 1-9, January 1996.

[53] Maeda N, Klyce SD, Smolek NK, "Neural network classification of corneal topography. Preliminary demonstration", Journal of Investigative Ophthalmology & Visual Science, vol. 36(7), pp. 1227-1235, 1995.

[54] H.Ranganath and N.Gunasekaran, "Artificial neural network approach in estimation of hemoglobin in human blood", International Computer Engineering Conference on New Technologies for the Information Society, ICENCO, pp. 341-344, 2004.

[55] http://www.dcs.bbk.ac.uk/~hui/IDA/projects.html

[56] http://www.generation5.org/content/2004 /NninAnaemia.asp

[57] Nipon Theera-Umpon and Paul D. Gader, "Training Neural Networks to Count White Blood Cells via a Minimum Counting Error Objective Function", 15th International Conference on Pattern Recognition (ICPR'00) , vol. 2, pp. 2299-2302, 2000.

[58] V.A. Kovalev, A.Y. Grigoriev, and H. Ahn, "Robust recognition of white blood cell images", IEEE Proceedings of the International Conference on Pattern Recognition, pp. 371-375, 1996.

[59] Sjostrom PJ, Frydel BP, Wahlerg LU, "Artificial neural network-aided image analysis system for cell counting", *Cytometry,* vol. 36, pp. 18-26, 1999.

[60] Q. Zheng, B. K. Milthrope, and A. S. Jones, "Direct neural network application for automated cell recognition", Cytometry Part A : The journal of the International Society for Analytical Cytology, vol. 57(1), pp. 1-9, Jan, 2004.

[61] M. Rendell, E. Anderson, W. Schlueter, J. Mailliard, D. Honigs, and R. Rosenthal, "Determination of hemoglobin levels in the finger using near infrared spectroscopy", Journal of Clinical & Laboratory Haematology, vol. 25, pp. 93, April 2003.

[62] Bacus JW, Weens JH, "An automated method of differential red blood cell classification with application to the diagnosis of anemia", The journal of histochemistry and cytochemistry: official journal of the Histochemistry Society, vol. 25(7), pp. 614-632, July, 1977.

[63] Leon L. Wheeless, Roy D. Robinson, Oleg P. Lapets, Christopher Cox, Ana Rubio, Michael Weintraub, Lennette J. Benjamin, "Classification of red blood cells as normal, sickle, or other abnormal, using a single image analysis feature", Journal of Cytometry, vol. 17, pp. 159-166, June, 1994

[64] Westerman MP, Bacus JW, "Red blood cell morphology in sickle cell anemia as determined by image processing analysis: the relationship to painful crises", American journal of clinical pathology, vol. 79(6), pp. 667-672, June, 1983.

[65] Bacus JW, Belanger MG, Aggarwal RK, Trobaugh FE Jr, "Image processing for automated erythrocyte classification", The journal of histochemistry and cytochemistry: official journal of the Histochemistry Society, vol. 24(1), pp. 195-201, Jan, 1976.

[66] N. Sinha and A.G. Ramakrishnan, "Automation of Differential Blood Count", Proceedings Conference on Convergent Technologies for Asia-Pacific Region TENCON, vol. 2, pp. 547 -551, Bangalore,India, 2003.

[67] S. F. Bikhet and A. M. Darwish and H. A. Tolba and S. I. Shaheen, "Segentation and Classification of White Blood Cells," IEEE Intl. Conf. On Acoustics, Speechand Signal Processing, ICASSP,pp. 2259-2261, 1999.

[68] Guclu Ongun and Ugur Halici and Kemal Leblebicioglu and Volkan Atalay, "Feature Extraction and Classification of Blood Cells for an Automated Differential Blood Count System," IEEE IJCNN, pp. 2461-2466, 2001

[69] J. Park and J. Keller, "Fuzzy Patch Label Relaxation in Bone Marrow Cell Segmentation," IEEE International Conference on Computational Cybernetics and Simulation, pp. 1133-1138, 1997.

[70] Bjorn Nilsson, Anders Heyden, "Segmentation of Dense Leukocyte Clusters", IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA'01), pp. 221, USA, Dec 2001.

[71] V. Kovalev, A.Y. Grigoriev, H. Ahn, N.K Myshkin, "Automatic localization and feature extraction of white blood cells", Proc. of SPIE Image Processing in Medical Imaging, vol. 2434, pp. 754-765, 1995.

[72] V. Kovalev, A.Y. Grigoriev, H. Ahn, N.K Myshkin, "Segmentation technique of complex image scene for an automatic blood-cell-counting system", Proc. of SPIE Image Processing in Medical Imaging, vol. 2710, pp. 805-810, 1996.

[73] S. Poon, R. Ward, and B. Palcic, "Automated image detection and segmentation in blood smears", Cytometry, vol. 13, pp. 766-774, 1992.

[74] http://www.willamette.edu/~gorr/classes/cs449/precond.html

[75] Yoh-Han Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Pub (Sd), March 1989.

[76] http://www.nhlbi.nih.gov/health/dci/Diseases/Sca/SCA_WhatIs.html

[77] http://en.wikipedia.org/wiki/Red_blood_cell