Abul Hasnat[1],
Anindya Ghosh[1],
Amina Khatun[2],
Santanu Halder[3]

Pattern Classification of Fabric Defects Using a Probabilistic Neural Network and Its Hardware Implementation using the Field Programmable Gate Array System

DOI: 10.5604/12303666.1227881

[1] **Government College of Engineering and Textile Technology,**
Berhampore, West Benga, India
E-mail: abulhasnat@gmail.com,
anindya.textile@gmail.com,

[2] **Jadavpur University,**
Kolkata, West Bengal, India
E-mail: aminak77@gmail.com

[3] **Government Govt. College of Engineering & Leather Technology,**
Kolkata, West Bengal, India
E-mail: sant.halder@gmail.com

Abstract
This study proposes a fabric defect classification system using a Probabilistic Neural Network (PNN) and its hardware implementation using a Field Programmable Gate Arrays (FPGA) based system. The PNN classifier achieves an accuracy of 98 ± 2% for the test data set, whereas the FPGA based hardware system of the PNN classifier realises about 94±2% testing accuracy. The FPGA system operates as fast as 50.777 MHz, corresponding to a clock period of 19.694 ns.

Key words: *classification, fabric defect, field programmable gate arrays, radial basis function, probabilistic neural network.*

Introduction

We must acknowledge the reality that every mass production system engenders a wide variety of flaws or defects in its products, and in this the respect textile industry is no exception. In the weaving shed, the production of flawless woven fabrics principally preoccupies the probing eye of the quality control department because a defective piece of fabric would weigh heavily on the cost burden of the garment industry and a second or off-quality product can barely reclaim more than 45-60% of the value of a fresh product. Different types of fabric defects are as follows: neps, broken threads, broken picks, and oil strains. Sample images of these defects are shown in ***Figure 1***. In the cut-throat competitive market of today, a conscious and discerning buyer would be expect to have zero tolerance for defects. And present day state-of-the-art technology enables a weaving machine to insert the weft at a rate of over 2000 m/min. Having scaled such a fantastic height for the weft insertion rate, a machine producing defective fabrics that escapes the vigilant eye of the weaver can be halted to

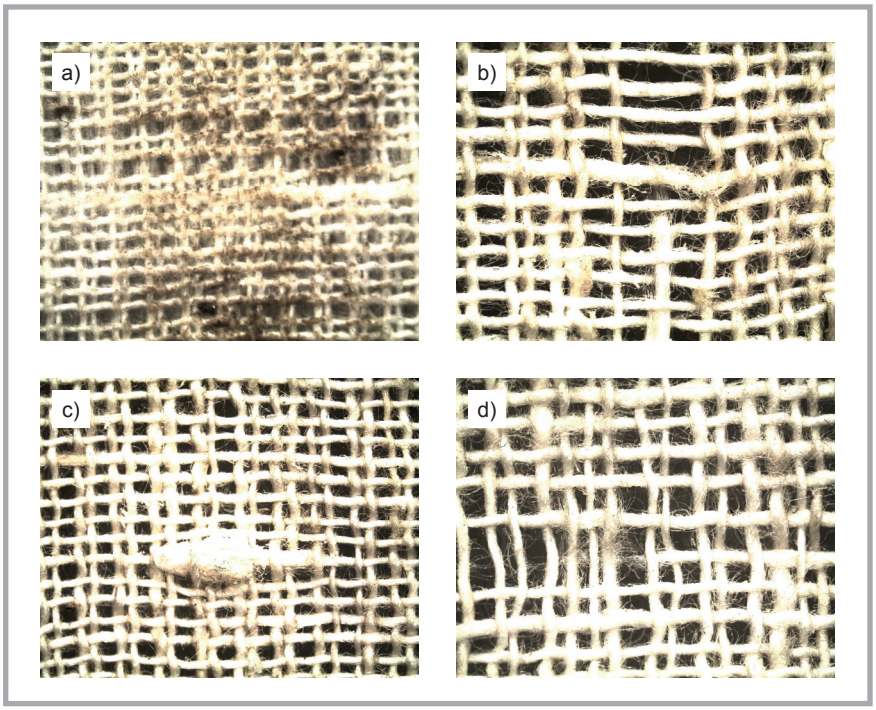

Figure 1. *Typical image of different fabric defects: (a) oil stain, (b) broken end, (c) neps, (d) broken pick.*

set right its malfunctioning not before an appreciable length of fabric has been woven forth. Hence it is highly desirable that as soon as a defect starts generating, it must be detected and its class must be ascertained so that the right remedial action can be initiated for the earliest restoration. Therefore the need of the hour is a real time on-line automatic fabric inspection system.

Recently Field Programmable Gate Arrays (FPGA) have become the dominant form of programmable logic [1-4]. FPGA can implement far larger logic functions compared to other programmable devices like programmable array logic (PAL) and complex programmable logic (CPLD). FPGA supports sufficient logic to implement complete systems and sub-systems for real time applications. FPGA exploits the increasing capacity of integrated circuits to provide designers with reconfigurable logic that can be programmed on an application-specific basis. FPGA design allows designers to create their own modules according to their needs and further upgrade the system conveniently. This drastically increases flexibility in both the design process and the final artifact by permitting one board-level design to perform many functions, or to be upgraded in the field. A system design based on FPGA is flexible, with the advantages of parallelism and low cost. But little work has been reported in the literature [5-6] related to the application of FPGA in textile technology. Hai-feng [5] used the Gabor filtering algorithm for defect detection and implemented the same using FPGA. Diana et al. [6] reported on the development of wearable smart fabrics with wireless communication capabilities using FPGA.

In this study, a Field Programmable Gate Array(FPGA) based fabric defect classification system using a Probabilistic Neural Network(PNN) is proposed to identify different fabric defects. Firstly a multi-class PNN [7-15] is used to construct a pattern recognition system for classifying fabric defects under different categories. In the case of the PNN classifier, the training process requires the computation of two equations only for

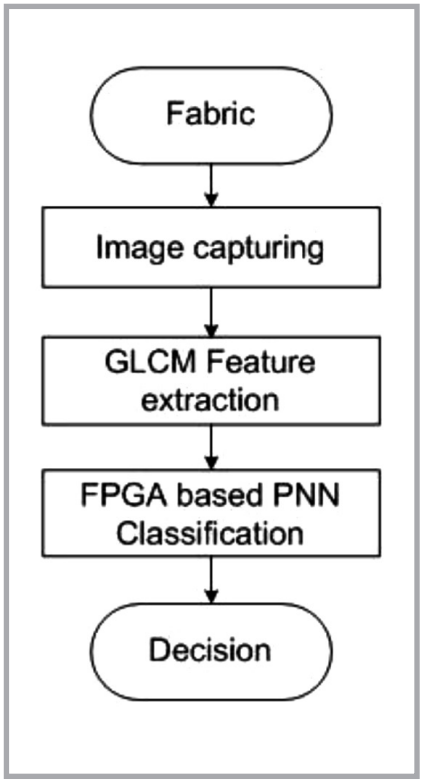

Figure 2. *Flowchart of fabric defect classification system.*

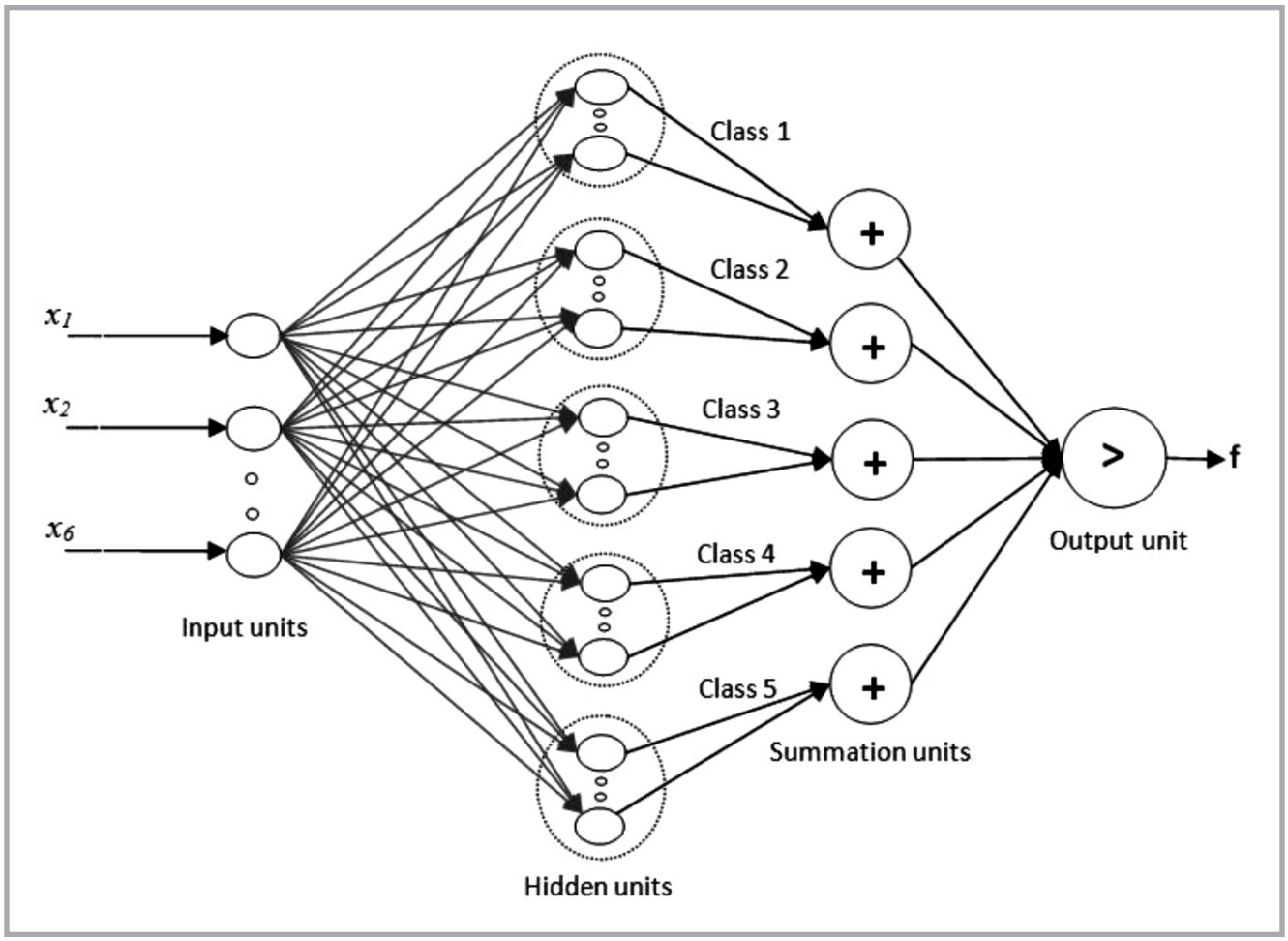

Figure 3. *Schematic representation of PNN architecture.*

each training vector [7-16], thus the time required for training using the training data set (data set for which the output class is known) is less [7-9] compared to other classifiers. The PNN training algorithm is a deterministic process, whereas the training algorithm of other classifiers involves a convergence process (where the number of iteration cannot be determined prior to execution). In a second application, a FPGA based hardware system is designed for possible implementation of the PNN classifier for identifying fabric defects.

Table 1. *Dataset for various kinds of fabric defects.*

Sample	f_1	f_2	f_3	f_4	f_5	f_6	Defects*
1	0.3900	0.6402	0.3584	0.4205	0.3726	0.3434	1
2	0.4026	0.6362	0.3601	0.4320	0.3438	0.3442	1
3	0.3879	0.6161	0.3419	0.4153	0.3228	0.3547	1
4	0.3931	0.6381	0.3569	0.4284	0.3694	0.4308	1
5	0.3826	0.6298	0.3537	0.4234	0.3489	0.3435	1
6	0.3978	0.6433	0.3704	0.4430	0.3584	0.3811	1
7	0.3920	0.6464	0.3532	0.4221	0.3352	0.3859	1
8	0.3887	0.6363	0.3601	0.4202	0.3220	0.3257	1
9	0.3880	0.6322	0.3672	0.4302	0.3481	0.3378	1
10	0.3851	0.6228	0.3567	0.4361	0.3496	0.3371	1
11	0.3689	0.6188	0.3483	0.4026	0.4393	0.4813	2
12	0.3789	0.6173	0.3447	0.4042	0.3954	0.4213	2
13	0.3663	0.6173	0.3444	0.4045	0.4439	0.4788	2
14	0.3881	0.6345	0.3569	0.4305	0.4214	0.5121	2
15	0.3964	0.6362	0.3512	0.4236	0.4049	0.4210	2
16	0.3529	0.5768	0.3219	0.3865	0.4417	0.4725	2
17	0.3465	0.5874	0.3225	0.3819	0.4740	0.5255	2
18	0.3467	0.5767	0.313	0.3782	0.3845	0.4925	2
19	0.3697	0.5805	0.3232	0.3978	0.4660	0.4953	2
20	0.3537	0.5642	0.3182	0.3918	0.4358	0.5035	2
21	0.3509	0.5957	0.3507	0.4079	0.5432	0.3107	3
22	0.3661	0.5915	0.3361	0.4137	0.4808	0.2884	3
23	0.3717	0.5968	0.3237	0.4003	0.4708	0.3376	3
24	0.3589	0.5903	0.323	0.3931	0.4377	0.3266	3
25	0.3436	0.5775	0.3298	0.3907	0.4888	0.3454	3
26	0.3159	0.5158	0.3214	0.3981	0.5433	0.3301	3
27	0.3354	0.5356	0.3373	0.4095	0.5594	0.3677	3
28	0.3231	0.5202	0.3197	0.3899	0.5466	0.3510	3
29	0.3534	0.5655	0.3275	0.4129	0.5210	0.3302	3
30	0.3761	0.5795	0.3399	0.4324	0.5290	0.3305	3
31	0.3723	0.5821	0.2097	0.3695	0.3453	0.3765	4
32	0.3836	0.6022	0.3054	0.3861	0.3383	0.3429	4
33	0.3716	0.5918	0.3101	0.3761	0.3595	0.3248	4
34	0.4115	0.6037	0.2797	0.4036	0.3987	0.3294	4
35	0.4321	0.6446	0.3090	0.4157	0.4254	0.3284	4
36	0.3765	0.608	0.3098	0.3842	0.3198	0.3587	4
37	0.3987	0.6132	0.3145	0.3954	0.3272	0.3829	4
38	0.3840	0.5953	0.3123	0.3920	0.3165	0.4022	4
39	0.3854	0.6023	0.3101	0.3890	0.3154	0.3635	4
40	0.3873	0.5970	0.3074	0.3944	0.3554	0.3735	4
41	0.4000	0.4976	0.3254	0.3969	0.5242	0.4233	5
42	0.2626	0.3115	0.2417	0.2633	0.4584	0.3841	5
43	0.2657	0.3276	0.2263	0.2723	0.3681	0.4321	5
44	0.3640	0.4823	0.3034	0.3518	0.5274	0.6200	5
45	0.4051	0.5158	0.3361	0.4082	0.6228	0.6095	5
46	0.3592	0.4453	0.3003	0.3543	0.4673	0.4100	5
47	0.4049	0.4874	0.3207	0.3977	0.5187	0.4240	5
48	0.3586	0.4805	0.3102	0.3614	0.4967	0.8066	5
49	0.3049	0.3866	0.2726	0.3215	0.4967	0.5492	5
50	0.4029	0.5257	0.3363	0.4028	0.5465	0.4661	5

*Fabric defects 1-normal, 2-neps, 3-broken threads, 4-broken picks, 5-oil strains

Fabric defect classification: PNN based software approach

The yarn defect classification system consists of the following steps – i) capturing images of the yarn defects, ii) feature extraction from the captured images, and iii) classification of the images using the Probabilistic Neural Network. The pattern recognition system for classifying fabric defects can be partitioned into a numbers of steps, as illustrated in ***Figure 2*** (see page 43). In this article, an attempt has been made to implement a FPGA based system for the classification step only. For this purpose, Tsai et al's [16] experimental data has been used, where they employed a grey level co-occurrence matrix to obtain the feature parameters f_1, f_2, f_3, f_4, f_5 & f_6 for various defect categories such as neps, broken ends, broken picks and oil strains. The categories are identified by numbers, namely, 1-normal, 2-nep, 3-broken threads, 4-broken picks, 5-oil strain. Among the feature vectors, f_1, f_2, f_3, and f_4 are the contrast measurement of texture images along 0°, 45°, 90° and 135° when spatial displacement $d = 1$, while f_5 and f_6 are the contrast values at $d = 12$, $\theta = 0°$ and $d = 16$, $\theta = 90°$, respectively, where θ is the direction angle. The dataset comprises of a total of 50 experimental data encompassing 10 experiments per category. ***Table 1*** refers to the datasets representing various fabric defects. These features are normalised in the range of [0 1] using the following ***Equation (1)***

$$f_i' = \frac{f_i - MIN\{f_i\}}{MAX\{f_i\} - MIN\{f_i\}} \qquad (1)$$

Five different categories, i.e., one for normal and the rest for four fabric defects, are classified using the PNN. The training data set along with its known class is used to train the PNN, and the test data set is used for cross validation.

Figure 3 (see page 43) shows a block diagram of the PNN classification system. When an input is presented, the hidden layer computes the distance from the input vector to the training input vectors. This produces a vector whose elements indicate how close the input is to the training inputs. The summation layer sums the contribution for each class of inputs and produces its net output as a vector of probabilities. Finally a compete transfer function for the output of the summation layer picks the maximum of these probabilities and indicates a particular class as the output.

The function of each layer of the PNN is discussed below.

The input layer contains $_m$ nodes (in this study $_m = 6$) for each of the input features of the vector, $x = \{f_1, f_2, ..., f_6\}$. These are fan-out nodes that branch at each feature input node to all nodes in the hidden layer, so that each hidden node receives the complete input feature vectors.

The hidden layer contains one neuron for each vector in the training data set. It stores the values of the predictor variables for the vector along with the target value. A hidden neuron computes the Euclidean distance of the test case from the neuron's center point (that is the stored vector, $x^{(P)}$) and then maps it to the Radial Basis Function (RBF), as given in ***Equation 2***:

$$f(x) = \frac{1}{\sqrt{(2\pi\sigma^2)^m}} e^{-\frac{1}{2}\left\{\frac{\|x-x^{(P)}\|}{\sigma}\right\}^2} \quad (2)$$

The σ value can be taken to be one-half the average distance between the feature vectors in the same group, or at each exemplar it can be one-half the distance from the exemplar to its nearest other exemplar vector.

The summation layer neurons compute the maximum likelihood of the pattern, x being classified into class c_j; $j = 1, 2, ... 5$, by summarising and averaging the output of all neurons that belong to the same class. The actual target category of each training case is stored with each hidden neuron; all the weighted values coming out from the hidden neurons (of a specific class) are fed only to the summation neuron that corresponds to the hidden neuron's category. The j^{th} summation node sums up the values received from the k^{th} group of hidden nodes using ***Equation 3***:

$$p_j(x) = \frac{1}{\sqrt{(2\pi\sigma^2)^m}} \frac{1}{N} \sum_{i=1}^{N} e^{-\frac{1}{2}\left\{\frac{\|x-x_i^{(P)}\|}{\sigma}\right\}^2} \quad (3)$$

Where N = number of hidden nodes for a particular class, $x_i^{(P)}$ denotes the vector stored in a hidden node.

The output layer classifies the pattern, x, in accordance with Bayes's decision rule based on the output of all the summation layer neurons as follows:

$$\hat{c}(j) = \arg\max_{j=1}^{5} \{p_j(x)\} \quad (4)$$

Hence the output layer compares the weighted votes for each of the five pattern nodes of the pattern layer, uses the largest vote to predict the target category, and thereby recognizes the test vector in the j^{th} class.

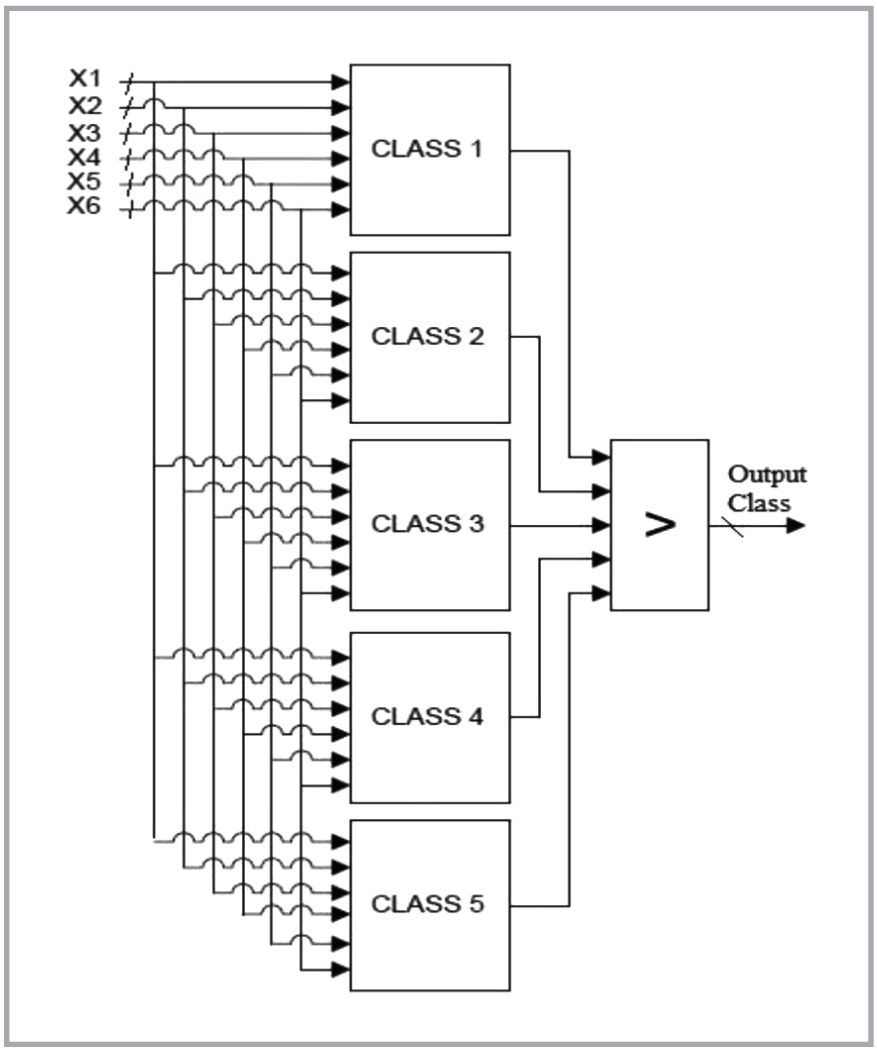

Figure 4. *System architecture of system proposed.*

The features of fabric defects extracted from the captured image and associated type of defects are initially used to train the five class PNN classifier and thereafter the trained PNN is used for test dataset classification. The performance of the PNN classifier is cross validated using k-fold cross validation.

FPGA implementation of PNN classifier

The system architecture for classification proposed is shown in ***Figure 4***, which has a pipelined architecture and consists of 5 class modules of 6 inputs of 23 bits and a comparator which has 1 output of 3 bits. From the output we can identify the type of defect corresponding to the input values.

Class modules

Allclass modules take a test vector $x = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ of six features, each of 23 bits as inputs. It computes the probability of belonging to the each class of the test vector and its computed value is forwarded to the comparator module. The probability value is computed using ***Equation 2***. Each of the class modules is comprised of 8 vector modules, one summation module and one multiplier module. The vector modules corresponding to each class perform the function of the hidden node in the hidden layer of the PNN. ***Figure 5*** (see page 46) shows the internal architecture of a class module.

Vector modules

Inside every class module there are 8 vector modules which store a training vector of 6 features, each of 23 bits. Each vector module calculates the exponent expression $\frac{1}{2}\left\{\frac{\|x-x^{(P)}\|}{\sigma}\right\}^2$ of ***Equation 2*** and forwards the value to the summation mod-

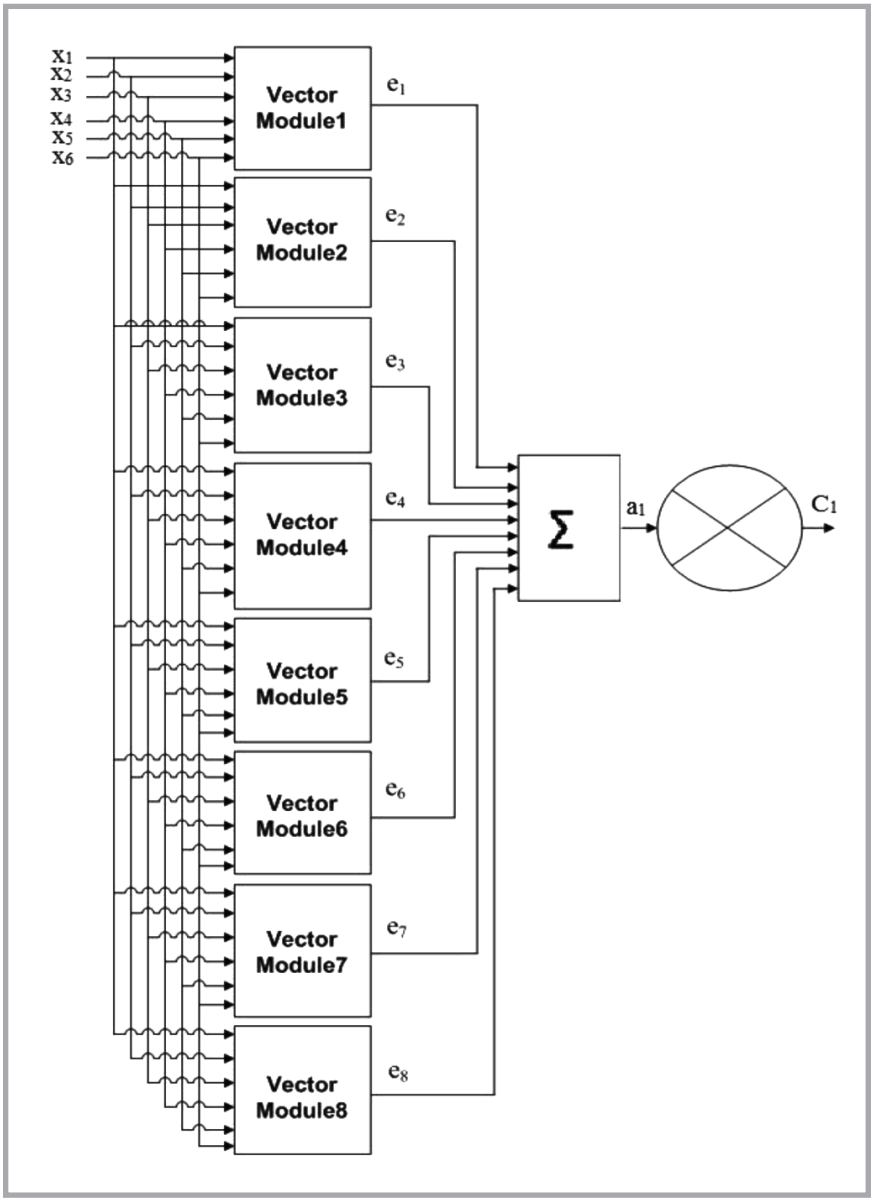

Figure 5. *Internal architecture of class module.*

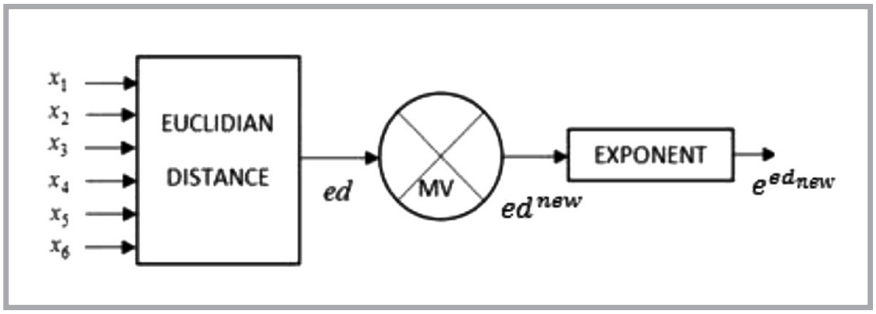

Figure 6. *Internal architecture of vector module.*

ule, which acts as the summation layer of the PNN. Each vector module again consists of a Euclidean distance calculator module, one multiplier unit and one exponent calculation unit. ***Figure 6*** shows the internal architecture of each vector module.

Euclidean distance unit in the vector module

One Euclidean distance unit exists in each of the vector modules, which stores the training vector $x^{(p)} = \{x_1^p, x_2^p, \ldots, x_6^p\}$, and upon receiving the test vector , it calculates the square of the Euclidean distance between the training vector $x = \{x_1, x_2, \ldots x_6\}$ and test vector using the equation given below:

$$ed = \sum_{i=1}^{n} \left\| x_i - x_i^{(P)} \right\|^2 \qquad (5)$$

The Euclidean distance calculation unit consists of 6 subtraction sub-units and 6 square sub-units. The architecture of the Euclidean distance calculation unit is shown in ***Figure 7***. Each of the subtractor sub-units stores one attribute (as 23 bit binary value) of the training vector and computes the absolute distance between the respective feature value of the test vector and forwards the result to the corresponding multiplier. The square sub-unit in the Euclidean distance calculation unit simply squares the input and forwards the result to the summation module, which sums up the 6 inputs and its output *ed* is forwarded to the outer multiplier unit.

Multiplier unit in the vector module

The multiplier unit in the vector module is used to multiply the constant value with the output value of the Euclidian distance unit. In this present work $\sigma = 0{,}10$ and $\frac{1}{2\sigma^2}$ is multiplied with *ed* as:

$$ed_{new} = ed \times \frac{1}{2\sigma^2} \qquad (6)$$

This calculated value is forwarded to the exponent unit for further calculation.

Exponent unit in the vector module

A Taylor series expansion of the exponent is given as:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots \text{ up to } \infty \qquad (7)$$

The present work takes into consideration up to the 10th power of x values. As $\frac{1}{2!}, \frac{1}{3!}, \ldots, \frac{1}{10!}$ are constants, these values are stored in a look-up table. The system architecture proposed is shown in ***Figure 8***, which has a pipelined architecture and consists of 18 multipliers as well as 10 adders for exponent calculation to achieve 21-bit accuracy.

All the inputs in each and every block are taken in 23 bits as well as all block forward outputs. The exponent unit takes $e^{ed_{new}}$ as the input and calculates the value of $e^{ed_{new}}$. ***Table 2*** shows the output of the calculated exponent value using the exponent block proposed.

Summation module

The summation module calculates the sum of all the vector modules to estimate the fitness of a test vector to a specific class using the following ***Equation (8)***:

$$S = \sum_{i=1}^{N} e_i^{ed_{new}} \tag{8}$$

where $e_i^{ed_{new}}$ is the calculated exponent value of the i^{th} vector module of a specific class module.

Multiplier unit in class module

The multiplier unit in the class module stores the value of constant $\frac{1}{\sqrt{(2\pi\sigma^2)^m}}\frac{1}{N}$ as a binary number and finally multiples this constant value (N = 8, m = 6, σ = 0.10) with the output of the summation module as given in ***Equation 9***.

$$F = \left(\frac{1}{\sqrt{(2\pi\sigma^2)^m}}\frac{1}{N}\right) \times e^{ed_{new}} \tag{9}$$

Comparator module

The comparator takes six 23 bit values as 6 inputs and finds the minimum value among them, because in the FPGA implementation the exponent unit calculates the exponent expression $e^{\frac{1}{2}\left\{\frac{\|x-x^{(P)}\|}{\sigma}\right\}^2}$ instead of $e^{-\frac{1}{2}\left\{\frac{\|x-x^{(P)}\|}{\sigma}\right\}^2}$ to avoid reciprocal calculation; because it is known from the duality principle that finding the maximum value from a set $A = \{\frac{1}{a_1}, \frac{1}{a_2}, .. \frac{1}{a_n}\}$ is equivalent to finding the minimum reciprocal of each element of that set, i.e., $A' = \{a_1, a_2, .. a_n\}$. This module generates a 3 bit output as a result. ***Table 3*** (see page 48) shows the output value and its interpretation regarding the type of fabric defect.

Results and discussion

In the software based approach, the dataset is divided into training and testing data sets using the *k*-fold cross validation technique to make the validation of the model more general and unbiased. The performance accuracy of the PNN

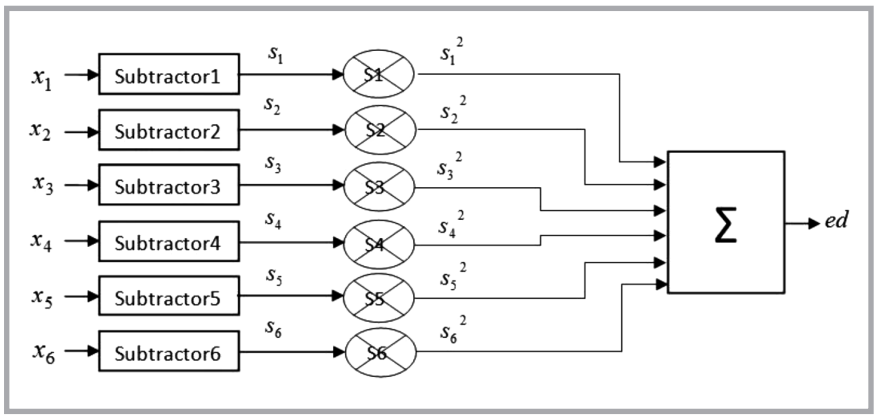

Figure 7. *Architecture of Euclidean distance calculation unit.*

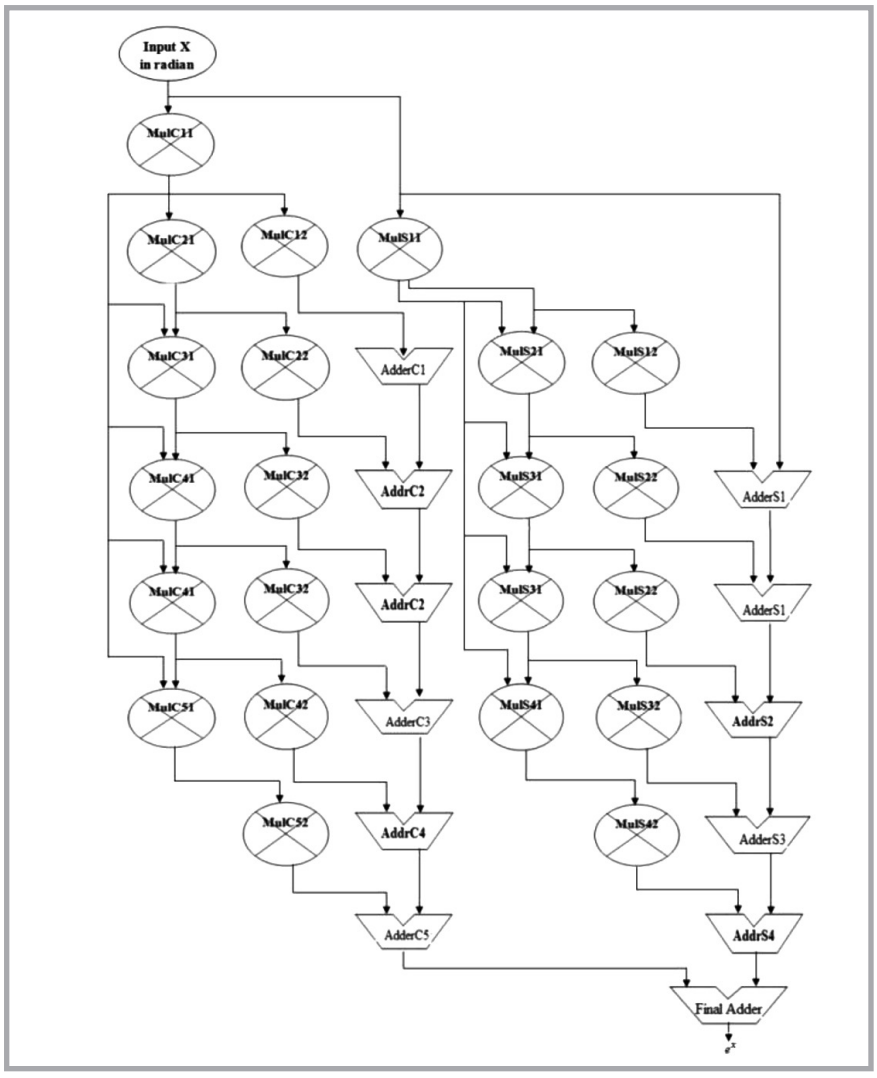

Figure 8. *System architecture of exponent unit.*

Table 2. *Sample outputs of exponent block.*

SL	*x* (Float)	*x*(23 bit)	e^x (Actual)	e^x (Computed 23 bit value)
1	0.45999991893768	01110101110000101000111	1.584073857	1.100101011000010111011 1
2	0.789999996185303	11001010001111010111000	2.203396342	10.001101000001000111001
3	0.98999989032745	11111101011100001010001	2.691234177	10.101100001111010010111
4	0.65809988975525	10101000011110010011110	1.9311197	1.11101110010111011110111
5	0.25959992408752	01000010011101010010010	1.2964114	1.01001011111000001100111

Table 3. Different classes of fabric defects and thier respective binary output.

Output	Defect
001	Normal (C_1)
010	Neps (C_2)
011	Broken threads (C_3)
100	Broken picks (C_4)
101	Oil strains (C_5)

Table 4. Device utilisation summary of architecture proposed.

Parameter	Used
Number of slices	57892
Number of flip flops	42196
Number of 4 input LUTs	93511
Number of bonded IOBs	143
Number of GCLKs	1
Maximum frequency	50.777 MHz

based software approach is assessed using the 5-fold cross validation technique. One fold is used as a test dataset at a time, while the system was trained using other four folds. In this way training and testing were done for 5 times. The generalized accuracies referring to the testing are estimated as the average accuracy±standard deviation (σ) of five cycles for the test set. The value of σ is tuned to be 0.1. The accuracies of testing are found to be 98 ± 2%.

In the FPGA based system, the same dataset is applied in a similar approach as described above. Initially each class module stores 8 training vectors for the respective class in the system. One test vector is given as the input to the FPGA based system, which predicts its class. For performance assessment of the system proposed, we stored the sample vectors alternately as training and testing in accordance with the 5-fold cross validation method. The system has demonstrated a 94±2% accuracy for the test vectors. The performance of the FPGA based system may produce a better result if more training vectors are stored by creating more hidden nodes for the respective class. A device utilisation summary is given in ***Table 4***. The FPGA based system proposed operates at a maximum frequency of 50.777 MHz, corresponding to a clock period of 19.694ns.

In comparison to the PNN based software approach, the FPGA based system shows less accuracy in predicting the class of the input vector because different parts of the system module truncate the result of various operations to the approximate value. But installation of a dedicated FPGA based system for inspecting fabric defects will be beneficial to the textile industry. The degree of parallelism in processing using the FPGA on the board system results in faster execution.

Conclusions

The present study outlines the application of computational intelligence in the field of automatic fabric defect classification. In this work a PNN based defect classification system and its respective FPGA based system have been designed. The training procedure of PNN involves no weight adjustment, but it simply stores each training vector as a hidden node in the hidden layer, and there is no concept of convergence of the training algorithm, hence hardware implementation of this PNN classifier is a feasible solution. Therefore FPGA implementation of a PNN based fabric defect identification system is a novel one and has great potentiality for automatic inspection of fabric defects in the textile industry. The PNN based software system and its FPGA based hardware implementation gives 98±2% and 94±2% accuracy, respectively, for the test data set. The FPGA based system operates with a maximum frequency of 50.777MHz. Future work may be extended to implement the FPGA based system for image capturing and GLCM matrix construction, which will be integrated with the present architecture to design a complete fabric inspection system.

References

1. John M and Sebastian S. Application Specific Integrated Circuits. Addison-Wesley, 1997.
2. Jenkins JH. Designing with FPGAs and CPLDs. Prentice-Hall, 1994.
3. Weste NH and Eshraghian K. Principles of CMOS VLSI Design: A Systems Perspective. Pearson Education Asia, 2000.
4. Bhasker J. A VHDL Primer. 3rd ed, Prentice Hall PTR, 1998.
5. Hai-feng C. Design and Implementation of Real-time Fabric Defect Detection System. *Advances in information Sciences and Service Sciences* 2012; 4(21): 23-30.
6. Diana M, Radu M, Nicholas HZ, Phillip S, Pradeep KK, Sungmee P, Sundaresan J, Stefan J, Christl L, Werner W, Tunde K, Didier C, Janusz G, Gerhard T, Mark J, Tom M and Zahi N. Electronic Textiles: A Platform for Pervasive Computing. *The IEEE,* 2003; 91(12): 1995-2018.
7. Specht DF. Probabilistic neural networks. *Neural Networks*1990; 3(1): 109-118.
8. El-EmaryIMM and RamakrishnanS.On the Application of Various Probabilistic Neural Networks in Solving Different Pattern Classification Problems. *World Applied Sciences Journal* 2008; 4(6): 772-780.
9. Eason G, Randall Wilson and D. Center Point Selection for Probabilistic Neural Networks. In: *International Conference on Artificial Neural Networks and Genetic Algorithms,* 1997. pp. 514-517.
10. Wilson DR and Tony RM. Heterogeneous Radial Basis Functions. *International Conference on Neural Networks,*1996; paper no.2, pp.1263-1267.
11. Wilson DR. and Tony RM. Improved Heterogeneous Distance Function. *Journal of Artificial Intelligence Research* 1997; 6(1):1-34.
12. Stanfill C. and Waltz D. Towards Memory Based Reasoning. *Communication of the ACM* 1986; 29(12): 1213-1228.
13. Ramakrishnan S. and Selvan S. Classification of Brain Tissues Using Multiwavelet Transformation and Probabilistic Neural Network. *International Journal of Simulation: Systems, Science and Technology* 2006; 7(9): 9-25.
14. Ramakrishnan S and Selvan S. Image Texture Classification Using Wavelet Based Curve Fitting and Probabilistic Neural Network. *International Journal of Imaging Systems and Technology* 2007; 17(4): 266-275.
15. Tian B, Mahmood R A, Thomas H, Vonder H and Donald R. Temporal Updating Scheme for Probabilistic Neural Network with Application to Satellite Cloud Classification. *IEEE Transactions on Neural Networks* 2000; 11(4): 903-920.
16. Tsai S, Lin CH and Lin JJ. Applying an Artificial Neural Network to Pattern Recognition in Fabric Defects. *Text Res J* 1995; 65(3): 123-130.

Received 15.03.2016 Reviewed 30.06.2016