English Doctoral Thesis

Usability of the Artificial Intelligence and Modern Techniques for Securing Computer Systems

Ing. David Malaník

Supervisor: doc. Mgr. Roman Jašek, Ph.D.

Tomas Bata University in Zlín
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence

Zlín, 2011

Acknowledgements:

I would like to express my warm thanks to:

- *My supervisor **assoc. prof. Roman Jašek**, for his support and invaluable discussions during my study and research on this thesis.*
- ***Martina Vaculíková** for her love which helps me a lot.*
- *and my **parents** who supported me during my studies*

RESUMÉ

Tato práce je zaměřena na možnosti využití neuronových sítí v oblasti zabezpečení počítačových systémů. Hlavní část této práce je zaměřena na využitelnost neuronových sítí pro identifikaci uživatelů. Neuronové sítě představují inteligentní systém, který je adaptovatelný na specifické vlastnosti uživatelů a lze jej použít i pro inteligentní rekonstrukci a identifikaci uživatelů.

Počítačová bezpečnost hraje v současném světě velmi významnou roli, PC systémy si postupně našly cestu do životního stylu, a mnoho lidí (potenciálních uživatelů) si už svět bez počítačů nedovede představit. Z tohoto vyplývá jedno bezpečnostní úskalí, kterým je personifikace dat uložených v těchto systémech. Uživatelé mají v PC systémech stále více svých osobních dat. Řešení této situace je ale velmi jednoduché, stačí mít svůj vlastní uživatelský účet a osobní data si chránit pomocí hesla. Problém tohoto řešení je v jisté uživatelské nepřívětivosti a v horší dostupnosti těchto dat.

Tato práce se zaměřuje na další možný postup ověřování uživatelů. Metody popsané v této práci zvyšují soukromí uživatelů PC. Neuronová síť je využitelné pro inteligentní verifikaci uživatelů, hlavní přínosy využití neuronových sítí jsou popsány v následujících kapitolách. Můžeme mezi ně zařadit možnost rekonstrukce poškozených verifikačních vzorků, nebo schopnost identifikovat jednotlivé biometrické vzorky. Sebeopravné funkce jsou demonstrovány na příkladu Hopfieldovy sítě. Další možnosti využití jsou popsány jako kombinace standartního uživatelského jména s heslem a biometrické identikace. Jednou z nejlepších vlastností neuronových sítí je schopnost se dynamicky přiučovat na měnící se charakteristiky chování ověřovaných uživatelů. Zde hlavně na jejich přirozené a vrozené charakteristiky. Tyto charakteristiky se s věkem mohou vyvíjet a je nutné aby na tento vývoj reagovala inteligentní struktura, které je schopná se na ně přiučit. Tento problém nastává hlavně u nestejně starých vzorků chování na které má systém reagovat.

V další části práce je popsáno jiné možné využití neuronové sítě, jedná se o možnost využití neuronové sítě jako cryptografického aparátu pro zabezpečení komunikace. Další část ukazuje nový pohled na verifikaci uživatelů. Běžné metody pro ověřování uživatelů jsou založeny na jednorázové identifikaci uživatele. Navržený systém ale obsahuje inteligentní rutinu, která periodicky ověřuje identitu uživatele. Toto oveření ale není realizováno pomocí dotazu na jméno a heslo ale využívá statistického vzorce chování uživatele. Systém je tedy schopen detekovat změnu uživatele bez korektního odhlášení stávajícího uživatele a přihlášení nového.

V poslední části této práce je popsán návrh systému, který obsahuje neuronovou síť pro biometrickou identifikaci uživatelů (otisky prstů v kombinaci se snímkem tváře).

Každá z popsaných aplikací je zaměřena na bezpečnost počítačových systémů a popisy jsou také přizpůsobeny tomuto vědnímu oboru.

Klíčová slova: bezpečnost počítačových systémů, umělé neuronové sítě, identity manager, autentizace uživatelů, biometrická identifikace, kryptografie

SUMMARY

This thesis is aimed to practical usability of the neural network in computer security application. The main point of this thesis is focused on the small part of neural network inside the user verification process. The neural network represented some intelligent system which might be adapted to specific user characteristic and might be usable for the smart reconstruction and identification users.

The computer security plays the one of the most important role in present World, computer system is embedded inside the living style and many people (or potentially users) do not imagine the World without some computers. But the computers and the data saved inside the computer systems represent potential dangerous. The dangerous is flowing from the personalization of data inside the computer world. The problem has a simple solution, just make your own account and protect these data by the username and password. This is not the best solution for the usability from the use friendly angle of view.

This work describes the next potential level of the user authorization and verification. The method described inside this thesis might grow of the user personal security and privacy. The neural network represents the intelligent structure which might be used for the smart and secure user verification. The abilities showing inside this thesis represent the procedures how to reconstruct the damaged samples for the biometric identification. This application is described on the Hopfield neural network with self-repair function. The next par of usability might be represented the user identification based on the combination of biometric and standard user verification processes. The best ability of each neural network is the ability to learn the network for the new problems and the dynamically self-teaching connected to the nature evolution of verifies users, because each user characteristic (this mean occasionally the behaviour of tested user) might be changed by the time. One of the best examples is the face identification problem based on the age difference between the stored sample and the actually scanned sample.

Other part of this work deals with the next potentially usability of smart neural network. One of these parts includes the cipher algorithms based on the neural network. Other one represented the new angle of view onto user verification. The commonly base method use only one-time user verification. But the systems included some intelligent behaviour, might periodically checking the user identity. Obviously not by providing the username and some password, but by processing the statistics analyses of current user. These systems are able to detect the user-switching without the correct logout and login procedure.

The last part of this thesis is based on the description of designed solution which contains the neural network for the user identification based on the biometric methods (fingerprint and the face identification in this case).

Each of these applications is focused to the computer security and description is adapted to this part of computer science.

Keywords: computer security, artificial neural networks, identity manager, user authentication, biometric identification, cryptography.

CONTENTS

LIST OF FIGURES **11**

LIST OF TABLES **14**

LIST OF SYMBOLS AND ABBEREVIATIONS **15**

1 INTRODUTION AND STATE OF ART **16**

2 DISSERTATION GOALS **19**

3 USER VERIFICATION PROBLEM **21**

3.1 BIOMETRIC IDENTIFICATION 21

3.1.1 Self-repair functions of neural network *21*

3.1.2 Neural network identification *22*

3.1.3 Advanced security for the username and password *23*

3.2 MULTIDIMENSIONAL AUTHENTICATION METHOD 24

3.2.1 Authentication continuum *24*

3.2.2 Blind shooting algorithm (BSA) *25*

4 NEURAL NETWORK CRYPTOGRAPHY **29**

4.1 CRYPTOGRAPHIC COMPLEXITY 29

4.2 CRYPTOGRAPHY STRUCTURE 30

4.3 CRYPTOGRAPHY ALGORITHM 30

5 SECURITY TOKEN **33**

6 SELF REPAIR ABILITIES **37**

6.1 HOPFIELD NEURAL NETWORK - FINGERPRINTS 37

6.2 HOPFIELD NEURAL NETWORK – FACE 40

6.2.1 Face reconstruction *40*

7 CLASSIFICATION OF SAMPLES **45**

7.1 NON-SECURE CLASSIFICATION 45

7.2 SECURING THE CODE BY BCH CODE 45

8 NEURAL NETWORK FINGERPRINT IDENTIFICATION **47**

8.1 TRAINING PROCESS DEFINITION 48

8.2 TRAINING PROGRESS 48

8.2.1 Progress – one hidden layer with 30 neurons *49*

8.2.2 Progress – one hidden layer with 50 neurons *51*

8.2.3 Progress – one hidden layer with 100 neurons *52*

8.2.4 Progress – one hidden layer with 1000 neurons *54*

8.2.5 Progress – two hidden layer with 500,100 neurons ... 55
8.2.6 Progress – two hidden layer with 750,300 neurons ... 57
8.2.7 Progress – two hidden layer with 1000,500 neurons ... 58
8.2.8 Progress – three hidden layer with 1000,1500,1000 neurons ... 60
8.2.9 Progress – three hidden layer with 1000,3000,1000 neurons ... 62
8.3 RESULTS – USABILITY OF NN ... 64
9 NEURAL NETWORK FACE IDENTIFICATION ... 65
9.1 TRAINING PROCESS DEFINITION ... 66
9.2 TRAINING PROGRESS ... 66
9.2.1 Progress – one hidden layer with 30 neurons ... 67
9.2.2 Progress – one hidden layer with 50 neurons ... 68
9.2.3 Progress – one hidden layer with 100 neurons ... 70
9.2.4 Progress – one hidden layer with 1000 neurons ... 71
9.2.5 Progress – two hidden layer with 500,100 neurons ... 73
9.2.6 Progress – two hidden layer with 750,300 neurons ... 74
9.2.7 Progress – two hidden layer with 1000,500 neurons ... 76
9.2.8 Progress – three hidden layer with 1000,1500,1000 neurons ... 77
9.2.9 Progress – three hidden layer with 1000,3000,1000 neurons ... 79
9.3 RESULTS – USABILITY OF NN ... 81
10 SECURITY SYSTEM ... 82
10.1 VERIFICATION GUI ... 83
10.1.1 User identification process ... 84
10.1.2 Class and function documentation ... 85
10.2 ADMIN GUI ... 93
10.2.1 Creating security token ... 93
10.2.2 Class and function documentation ... 94
11 CONCLUSION AND DISCUSSIONS ... 98
12 LIST OF AUTHOR'S PUBLICATION ACTIVITIES ... 101
13 REFERENCES ... 103
APPENDIX ... 105

LIST OF FIGURES

Fig. 1.1 - User name and password dialog 16
Fig. 1.2 – Eye IRIS and fingerprint 17
Fig. 3.1 - Hopfield neural network schema [26] 21
Fig. 3.2 - BSA schema 26
Fig. 3.3 - BSA progress 27
Fig. 4.1 - Basic/extended (real) structure of neural network 29
Fig. 4.2 - Public and private part of crypto-network 30
Fig. 4.3 - Encryption process 31
Fig. 4.4 - Decryption process 32
Fig. 5.1 - Security token structure 33
Fig. 5.2 - Duplex trust relationship 34
Fig. 5.3 - One way trust model 35
Fig. 6.1 - Testing samples 37
Fig. 6.2 - Output string format 37
Fig. 6.3 -75% noise, testing samples 38
Fig. 6.4 - Testing result 39
Fig. 6.5 - Source faces 40
Fig. 6.6 - 40% on noise 41
Fig. 6.7 - 40% of noise – reconstruction 42
Fig. 6.8 - 60% of noise 43
Fig. 6.9 - 60% of noise - reconstruction 44
Fig. 8.1 - Training set generation process 47
Fig. 8.2 - Input pair generation process 48
Fig. 8.3 - Training error L30 50
Fig. 8.4 - Training error difference L30 50
Fig. 8.5 - Training error L50 51
Fig. 8.6 - Training error difference L50 52
Fig. 8.7 - Training error L100 53
Fig. 8.8 - Training error differences L100 53
Fig. 8.9 - Training error L1000 54

Fig. 8.10 - Training error differences L1000 .... *55*
Fig. 8.11 - Training error L500, L100 .... *56*
Fig. 8.12 - Training error differences L500, L100 .... *56*
Fig. 8.13 - Training error L750, L300 .... *57*
Fig. 8.14 - Training error differences L750, L300 .... *58*
Fig. 8.15 - Training error L1000, L500 .... *59*
Fig. 8.16 - Training error differences L1000, L500 .... *59*
Fig. 8.17 - Training error L1000, L1500, L1000 .... *61*
Fig. 8.18 - Training error differences L1000, L1500, L1000 .... *61*
Fig. 8.19 - Training error L1000, L3000, L1000 .... *63*
Fig. 8.20 - Training error differences L1000, L3000, L1000 .... *63*
Fig. 9.1 - Training set generation - face *65*
Fig. 9.2 - Input pair generation process *65*
Fig. 9.3 - Training error L30 .... *67*
Fig. 9.4 - Training error differences L30 .... *68*
Fig. 9.5 - Training error L50 .... *69*
Fig. 9.6 - Training error differences L50 .... *69*
Fig. 9.7 - Training error L100 .... *70*
Fig. 9.8 - Training error differences L100 .... *71*
Fig. 9.9 - Training error L1000 .... *72*
Fig. 9.10 - Training error differences L1000 .... *72*
Fig. 9.11 - Training error L500, L100 .... *73*
Fig. 9.12 - Training error differences L500, L100 .... *74*
Fig. 9.13 - Training error L750, L300 .... *75*
Fig. 9.14 - Training error differences L750, L300 .... *75*
Fig. 9.15 - Training error L1000, L500 .... *76*
Fig. 9.16 - Training error differences L1000, L500 .... *77*
Fig. 9.17 - Training error L1000, L1500, L1000 .... *78*
Fig. 9.18 - Training error differences L1000, L1500, L1000 .... *78*
Fig. 9.19 - Training error L1000, L3000, L1000 .... *80*
Fig. 9.20 - Training error differences L1000, L3000, L1000 .... *80*
Fig. 10.1 - Security system UserGUI *82*

Fig. 10.2 - Security system Admin part *83*

Fig. 10.3 - User verification process *84*

Fig. 10.4 - Interface UserGUI *85*

Fig. 10.5 - Class SecurityToken *86*

Fig. 10.6 - Class UserCharacteristics *87*

Fig. 10.7 - Class Characteristic *88*

Fig. 10.8 - Class DatabaseClass *89*

Fig. 10.9 - Class ImageOperations *90*

Fig. 10.10 - Class NeuralNet *91*

Fig. 10.11 - Class NeuralNetOperations *92*

Fig. 10.12 - Creating of security token *93*

Fig. 10.13 - Class AdminGUI *95*

Fig. 10.14 - Class BCHCodes *96*

Fig. 10.15 - Class FileOperations *97*

LIST OF TABLES

Table 3.1 - Binary and double neuron comparision 22
Table 3.2 - userhash generator 23
Table 7.1 - Non-secure classification 45
Table 7.2 - Secure classification by the BCH code 46
Table 8.1 - Training L30 49
Table 8.2 - Training L50 51
Table 8.3 - Training L100 52
Table 8.4 - Training L1000 54
Table 8.5 - Training L500, L100 55
Table 8.6 - Training L750, L300 57
Table 8.7 - Training L1000, L500 58
Table 8.8 - Training L1000, L1500, L1000 60
Table 8.9 - Training L1000, L3000, L1000 62
Table 9.1 - Training L30 67
Table 9.2 - Training L50 68
Table 9.3 - Training L100 70
Table 9.4 - Training L1000 71
Table 9.5 - Training L500, L100 73
Table 9.6 - Training L750, L300 74
Table 9.7 - Training L1000, L500 76
Table 9.8 - Training L1000, L1500, L1000 77
Table 9.9 - Training L1000, L3000, L1000 79

LIST OF SYMBOLS AND ABBEREVIATIONS

symbol	*description*
B/W	2 colour picture, Black and White
Double	type of programing variable, 64-bit number
NN	neural network
BSA	blind shooting algorithm
GUI	Graphical user interface
userhash	combination of hash value from username and password
IDS	Intruder detection system

1 INTRODUTION AND STATE OF ART

The computer security is one of the most expandable parts of the computer science in the modern world. Some companies starting solve the problem of the computer security. Obviously the first problem is secure authentication of employees. The classical method for solving this problem is the authentication manager based on simply method such as a user name and assignment password. This will be a good solution for same basic level of authentication. It will be secure or very unsecure; it depends on usage of these methods [22].

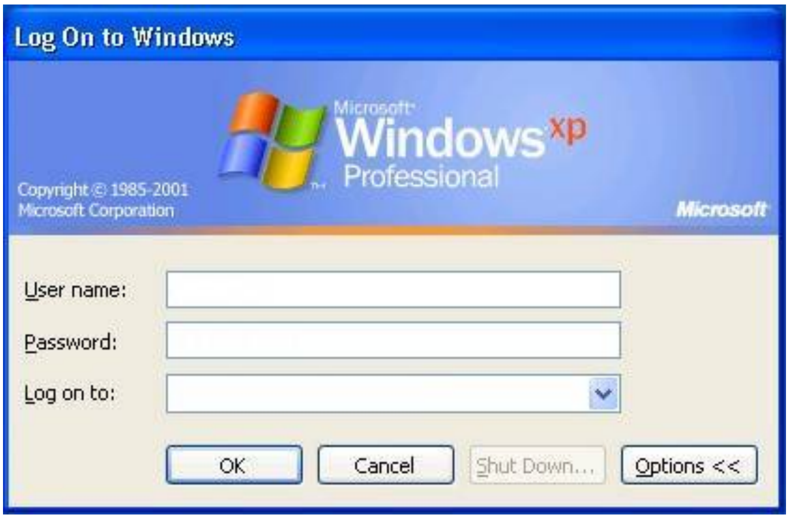


Fig. 1.1 - User name and password dialog

Fig. 1.1 shows the typical login screen based on username and password authentication. There is only one level of security, each people, which known the username and the password are able to log into this computer. Each people don’t mean the allowed people, because the login credentials are specific for each user. Access permission granularity is based on specific user names. But the username and the password are distributive. It is only the text information, which will be revealed, stole or gifted [9,4].

More secure methods for authentication people, which are allowed for working with specific computer is biometric identification. Each people have specific biometric markants, which identified everyone. These markants are findable in fingerprints, voice, eyes, ears and DNA. Reliability of these biometric branches is not the same. There are more factors witch affecting percent of potentially errors in the identification. The most commonly used markants are shown on Fig. 1.2 [2].

Fig. 1.2 – Eye IRIS and fingerprint

The combination of these two main featured methods is one the most discussed solution for the computer security round the World. This combination is based on very simple username and password authentication, but there is something more; the human originality. The difference is original biometric factor for secure authentication of each user. The user name and the password will be stolen or gifted; inside biometric methods are these potentially dangerous weakness minimized. Stealing or gifting of biometric data is quite complicated. Now there is a complexity password with originally biometric information about the user.

The commonly uses biometrical system for user verification procedures work with the markant, the specific type of fingerprint structures [2,7,11]. But using of these specific part income the specific problem. If there is only few number of markants or the scanned sample has poor quality, the system could not verify this sample (or user). The quality and the amount of data, in this case amount of visible markants affect the identification process. The next widely described problem is the question of some few injuries, for example the scratch on the finger. This might hit the marginal markants and system identification is impossible.

More or les, the biometrical identification is suggested to places with the high level of security or with the very limited access [7,9,19,22,24].

This thesis deals with the next generation of the user authentication methods. These methods are represented as a multidimensional functions collected from many factors of the user identification. Each dimensions of the security token represents the one originally and stable user account attribute for the user authentication. The result will be graphically shown as a multidimensional function with the local and the global extreme values.

2 DISSERTATION GOALS

The aim of the work is to apply and verify that it is possible to create a new angle of view to the assembly of the neural networks to the computer security. The aim of the verify part contains the verification procedures for the specific usability of the neural networks in user identification and verification problems.

The steps leading to this dissertation could be summarized as follows:

- To prove that there is the place for the assembly of the neural networks to the user verification processes
- To prove that the neural network might successfully verified the user or might repaired the sample used for the user verification
- To prove that the speed of the solution based on the neural network is inside the practical limited (for example: the time for the learning, identification, etc.)
- To define the part of standard user verification procedure that might be replaced or proved by the neural network.
- To define the limited of usage for the neural network inside the computer security application.
- To develop a system prototype for testing neural network application inside the computer security processes.

THEORETICAL FRAMEWORK

3 USER VERIFICATION PROBLEM

3.1 Biometric identification

The biometric identification is the perfect solution for the verification of computer users. There are many methods to minimize the incorrect verification. But there is the one so important problem. The problem is quality of the scanned fingerprint or the eye iris. The quality of these “pictures” will be perfect/normal/poor. The technical environment might produce a less detailed picture. These messy pictures might produce the fail during the user authentication. This failure might be critical in emergency systems such as the biometric lock on the door.

The artificial neural network will solve question of quality, especially by self-repair functions of neural network.

3.1.1 Self-repair functions of neural network

The thesis deals with the self-repaired function of the Hopfield neural network for reconstruction of fingerprints. The Hopfield artificial neural network was chosen for this excellent self-repaired function [27]. The general schema of used neural network shows Fig. 3.1.

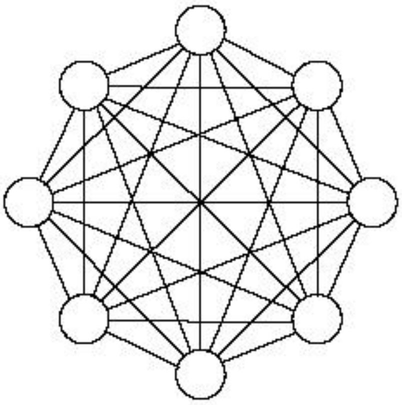


Fig. 3.1 - Hopfield neural network schema [26]

Experiments described in 6.1 tested three basic samples of fingerprints markants. The first step was teaching the network for specific samples. Samples were prepared as a binary string of monochromatic fingerprint. Each neuron in network represented one pixel of the source image. The sample with dimension 300 x 400 pixels required the neural network with 120 000 binary neurons. This amount of binary neurons t represents the big problem for the computer system, high loading and slow computing.

The next possible step for the minimizing these affect is compression or grouping of the neurons/pixel of pictures. The solution might be in grouping binary pixel with representation B/W colours to Double type values. Examples are shown below.

Table 3.1 - Binary and double neuron comparision

Picture size	n. binary neurons in Hopfield NN	n. of double neurons in NN
300 x 400	120 000	1 875
280 x 320	89 600	1 400

This step reduces the size of NN and allows fast teaching process.

3.1.2 Neural network identification

The next problem that is flowing from the user authentication or verification processes is the problem of identification. This problem will be solved by the neural network classification. The first step is teaching the network with the pattern samples such as fingerprints from verified users, users face, and specific users characteristics. This method for authentication is occasionally un-implementable from the start. The starting point must be some commonly used method for user verification. Then might be learned the NN and process might continue. The basic problem from the auto classification of testing samples is flowing from the problem of code differences between the samples. There is impossible (extremely dangerous) to design the NN classification as an increment ID number for the user, because there is only the 1 bit code differences between the neighbours samples. This problem is detailed described in 7.2.

3.1.3 Advanced security for the username and password

The username and password hash represents the potential security issues in any databases or other methods for authentications systems. In the front of this potential dangerous which is flowing from the potentially misuse of the username, demonstration program used in this thesis implement different type of authentication with username and password.

The commonly based systems use username and passwords as a two values stored in the internal or external database. The username is stored as a blank test and the password is stored as a hash function of real password. The problem is coming from the data-mining methods. If the attacker receives the username, he is able to use it. If the attacker stole the password hash, he is able to try crack it.

Next possible method is store username as a hash and password as a hash, but the problem is still same. If there is a full hash of various string values, it is possible to try crack it by the brutal force.

This system use combination of both hashes as the userhash authentication string for secure verification of user identity. The principle is quite simple. There are two hashes at the start. The first of them is has of username and the second one is the hash from the password. Next, the system mixes it to one hash. The odd characters of final hash are the odd characters from the username hash and the even characters are the even characters from the password hash.

This principle is better illustrated in the table bellow.

Table 3.2 - userhash generator

Username hash	**ee26b0dd4af7e749aa1a8ee3c10ae9923f61898077...**
Password hash	**021b5b2ac76d969722e8b55d88d7a0c83dc61a73f...**
Final userhash	**e22bbbda47fd…**

Some information about username and hash is missing, but amount of information is still sufficient. If the attacker stole the userhash he does not have full information about username or password.

3.2 MULTIDIMENSIONAL AUTHENTICATION METHOD

There are many methods for the user authentication. Every method is built on the small group of cryptographic models. The most commonly using method for user password verification is hash verification. The password is as a hash code of the password string. The authentication routine just compares the stored hash code and the hashed password which user type to the authentication window. But after this verification is the user verified or not. This action is only once. This is only at the beginning of work. There were some other methods to prevent a user changing without de-authorization and other user authentication. The common rule is: if you leave your computer, you must log out or lock your account. But this is not real in many organisations. For example: if user goes to the toilet, the user doesn't log out his account with some processing work. In many cases, the user just goes. In minority cases, the user locks his account.

The problem of commonly using authentication methods is, that the user was verified only once; at the beginning of work. Next, the user is verified until he stop his work and "touch logout button". During his work, the authentication environment doesn't check if the working user is the authentication user. The system just trusts him [7].

3.2.1 Authentication continuum

The developing system is based on the authentication continuum. The continuum is assignment to the user action and using multidimensional space for a multifactor authentication of user login. The main point is approximation of the user character. Each user is original. Each user has specific characterisation, which might verify him. It will be the specific style of writing (the statistics style – not the literal). The aim is combine this user characteristic factor to one multidimensional function combined with his password. Each characteristic represents one dimension of the user verification surface. The surface is deformed by many factors. There is some basic verification surface based on the specific function such as the Ackley, the De Jong, the Rana, etc.; this surface is combined with the user characteristic and makes the originally surface with the global extreme value. The extreme value is the floating point in the multidimensional space, which represent the originally user private key. This is the one private key forever. The extreme value is not stable, it is not the one point with coordinates {x,y,z,...}. It is floating point; it must be adaptable to the human life. This adaptability is solving with the neural network, especially by the one feature of the neural network. The feature is dynamic self-learning of the neural network.

The authentication subsystem periodically collects user characteristic and if is in the safe limit, the system allow the neural network adaptation. Next, the system controls the specific extreme point of the user behaviour. If the extreme position and the value is the same as in the multidimensional user surface, the system allows the user to continue in working (user doesn't know that the system made some actions). If there are some unexpected changes, the system logout the user for preventing potentially unsecure behaviour and save the log contains this action.

3.2.2 Blind shooting algorithm (BSA)

The BSA was developed as the algorithm for finding extreme values and location of the multidimensional function. The algorithm was named as a "Blind Shooting Algorithm". The origin behaviour of the BSA is in nature behaviour sequent from the shooting without vision.

The basic schema of the BSA is shows on the figure below, there were many parallel computing values during searching for the local extreme in the specific area [6]. The BSA is written as a multithread algorithm for extreme finding application.

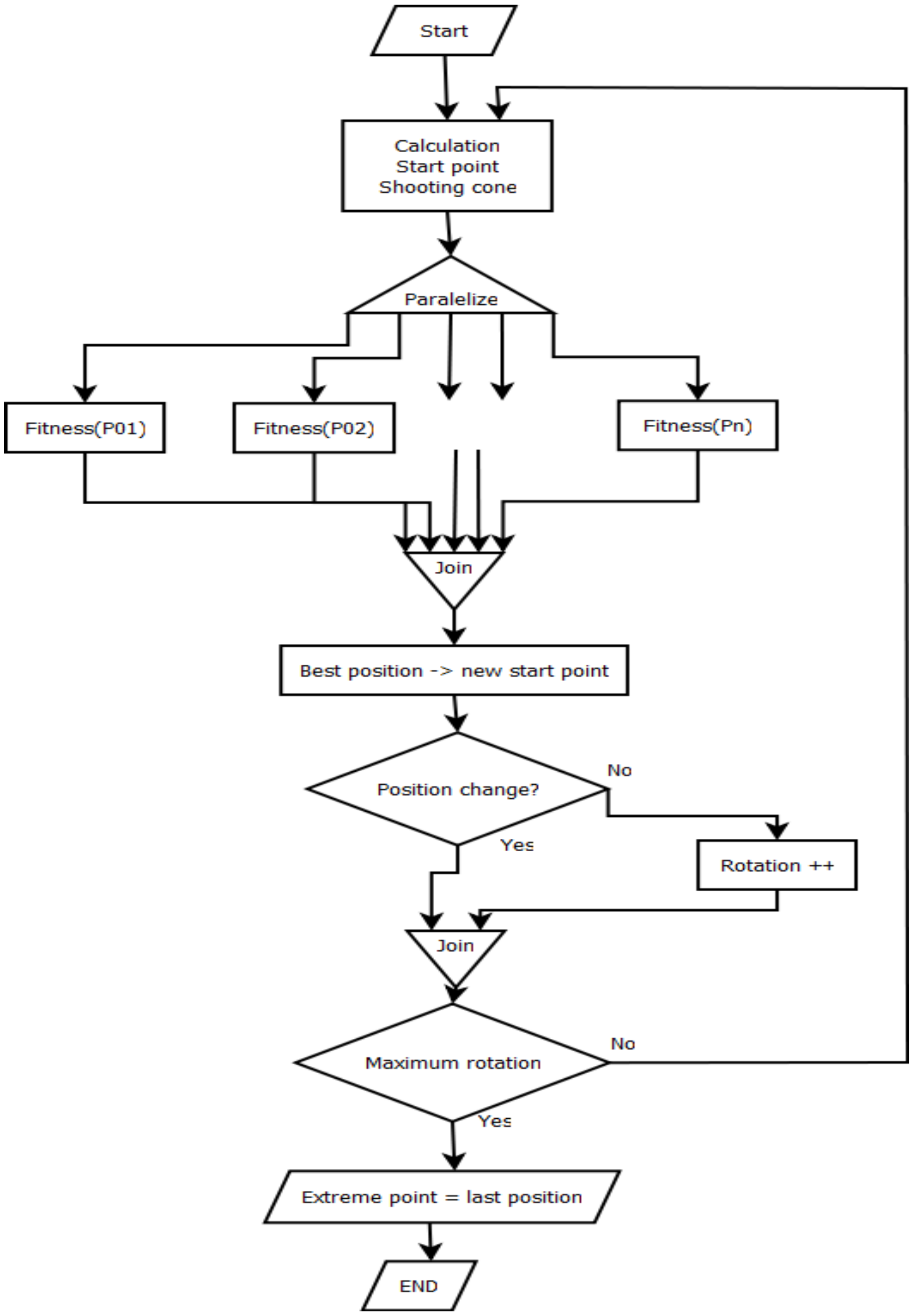


Fig. 3.2 - BSA schema

The graphical progress of the algorithm is on the Fig. 3.3. The algorithm starts at the top left part of the picture and end at the right bottom part.

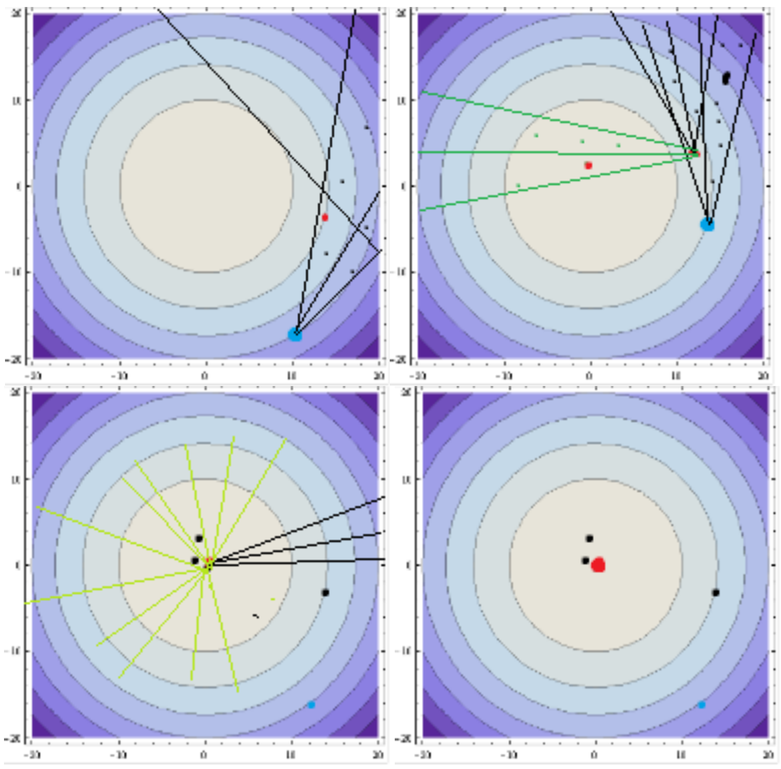


Fig. 3.3 - BSA progress

The first part of the picture represented the starting point of the algorithm. The blue point is the starting point. This point was generated as a random set of coordinates. The black cone is the searching area for this step. The black points are generated point coordinates for calculating values. The black points represent the next potentially starting point. After the calculation, there is the algorithm for finding the best value in testing set of black points. This algorithm computes with some stochastic behaviour and might choose a different value that the local maximum value. The stochastic part is adjustable by BSA parameters. The local extreme was marked as the red point. This is the starting point for the next iteration.

The second part of the Fig. 3.3 (the right top) describes the second iteration of the BSA. There is a new starting point marked as blue point and a new shooting black cone. The cone is in direction with the gradient between the last starting point and the present starting point. In this case, the iteration was not found a better value than is the starting point. The BSA produces rotation of the searching cone. The rotation is user specified. The rotation is present as the green cone in the picture simulation. The new best value is marked as the red point.

The left bottom part shows the last position changing in the algorithm. The BSA possibly found the best solution. There was a user specified parameter, which represent the maximal number of rotation from one starting point. Rotations are marked as green cones on the picture.

The last part of the picture shows the end of searching. The red point represents the global extreme of the testing function.

The detailed test outputs from BSA are described in APPENDIX A.

4 NEURAL NETWORK CRYPTOGRAPHY

The neural network will be used for the cryptography. The main subject, which is important for this purpose is complexity and structure with connection between nodes (neurons). The node couldn't be connected only with the one other node. One node occasionally has more parents and more children in neural network hierarchy. There will be connections between levels of neural network. The basic schema of neural network shows Fig. 4.1. The left side of figure shows very simple type of artificial neural network, there are only 3 levels/layers of network. The input layer is for incoming values. The output layer is for displaying results and the hidden layer for transforming data. The right side represents the massive structure of neural network. There are many connections, layers and nodes [28].

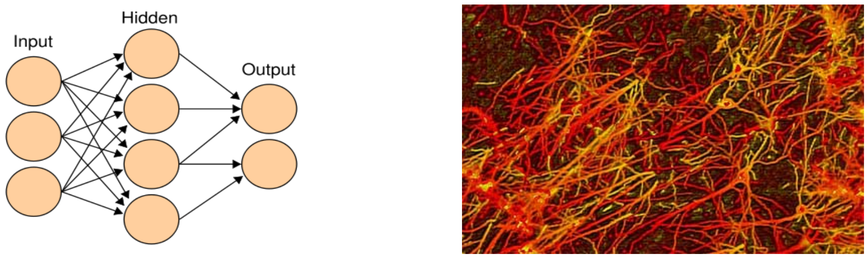


Fig. 4.1 - Basic/extended (real) structure of neural network

4.1 Cryptographic complexity

As is shown on the figure above, the structure of "the basic network" will be reproduced significantly easy. There are only 3 layers and 9 neurons. The number of combination is relatively small. But the right side of Fig. 4.1 contains many hundred/thousand/million/etc. neurons and many connections. It is impossible to reproduce the network without knowing the information about structure, connections and weights. The number off possibilities is enormous. This property (extremely complexity of crypto analysis) is one of the best features for cryptography [5].

4.2 Cryptography structure

The complexity of structure is the main point. Firstly, there was a structure with many neurons and layers. The minimal limit for layers is 8 layers. The minimal limit for neurons in the next layer depends on previous layer as shows below. The letter *i* represents layer index, the letter *n* is number of neurons in specific layer.

$$n_{i+1} \gg n_i \qquad (3.1)$$

The Fig. 4.2 represents the crypto-neural network. There are 8 binary neurons in the input layer. The network will encrypt 8-bit words (ASCII). The left part, which includes the input layer and some randomly generated part of the network, is as a public part of the algorithm. Everybody might use it and publish. The other part is the private part with output/result layer and reminders neurons of the network.

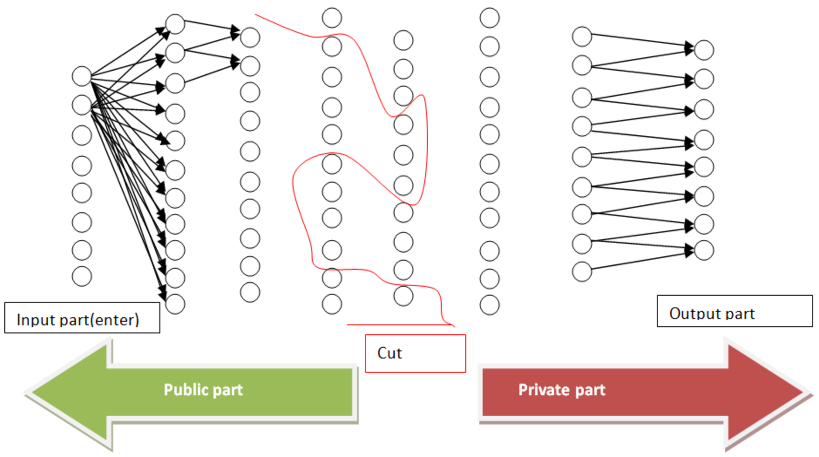


Fig. 4.2 - Public and private part of crypto-network

4.3 Cryptography algorithm

Firstly, specify the dimension of inputs. It depends on the cryptography input text. It is not recommended to use the network with one binary input. For text is recommended to use a 16 binary input layer, the 16 binary words represent Unicode coding extension. The number of input neurons depends on data string, which is used for data in secured message. The quantum of input neurons is unlimited.

The next step is generation pseudorandom noise of the message. The generated noise string must have the same dimension as the source (open) message. There were 3 vectors of binary numbers:

1. Binary representation of open message (*OM ={0,0,0,1,1,0,1,1,0,1,}*)
2. Pseudorandom binary noise of message (*PRN={0,1,1,1,1,0,1,0,1,1}*)
3. Binary message as an input to the network – source open message with pseudorandom noise $IM = OM + PRN$ (*IM = {0,0,0,1,1,0,1,0,0,1}*). The addition of noise is realizes by first layer of the neural network. The noise vector is a part of he neural network structure.

The training set for teaching of neural network contains *OM* and *OM*. The *OM* is set with inputs to network; the *OM* is set of requested outputs. Next continue with training of network by the Back propagation algorithm, the Differential evolution or the Genetic algorithms.

Now there was the learned network. Then it is necessary to generate a random cutting path between layers of the neural network. The cut distribute the network to two main parts. The first is the public part, which contain input layer and some hidden layer of the network. The second part contains the output layer and some rest neurons.

The encryption process shows Fig. *4.3*. The network cut represented output layer from the public part of crypto-network.

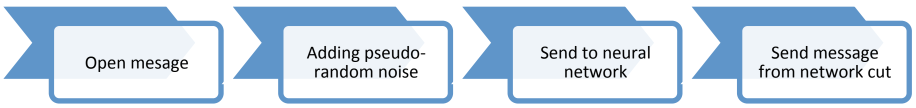


Fig. 4.3 - Encryption process

The decryption process is applied rest part of the network (signed as private part of the neural network). The cut place works as an input layer of the "new" neural network. Output layer of rest neural network represent the decrypted message.

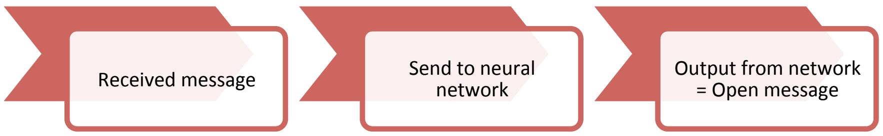


Fig. 4.4 - Decryption process

5 SECURITY TOKEN

The security token represents the virtualized solution for the user authorization. The token is a combination of each previously described technique. The token uses the biometric identification subroutine, which enables the biometric identification of the user. The self-repair function of the biometric identification is included inside these biometric subroutines. The next part of the security token is the multidimensional user identification subsystem, which represents the real-time verification system. This system could identify the user switching without the logoff and the login actions. The data was secure by the neural network cryptography subsystem. The token assembly the neural network technologies and produce the complete package for using. The package is shown on the Fig. 5.1.

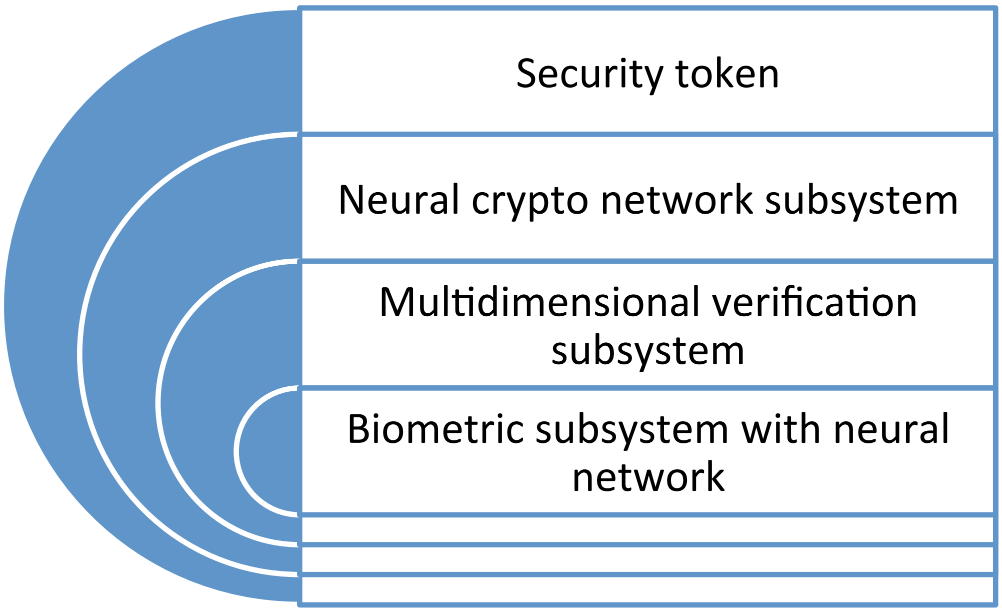

Security token
Neural crypto network subsystem
Multidimensional verification subsystem
Biometric subsystem with neural network

Fig. 5.1 - Security token structure

The token can work in 2 different states. The first is the user authentication token, which produces the user credentials to secured system. The second function might be the whole security authentication system, which has trust relations with the master security system. The trust relationship is secure with asymmetric key pair sharing between the token security system and the master security system [22]. The key exchange is shown on the Fig. 5.2.

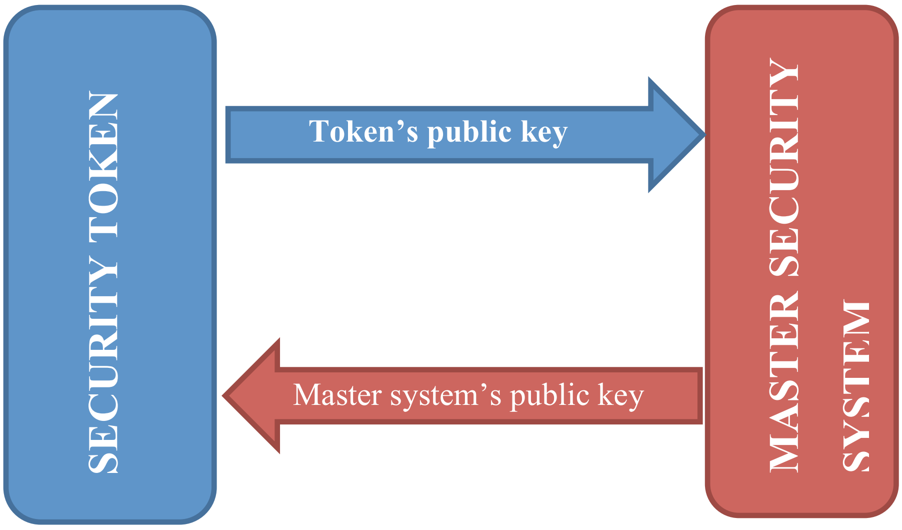


Fig. 5.2 - Duplex trust relationship

The relationship is trust model between the security token and the master security system. The relationship might be duplex or the one way. It is possible to have one way trust model between the security token and the master system which mean, the master system trust user verification produced by the security token. The one-way relationship is on the Fig. 5.3.

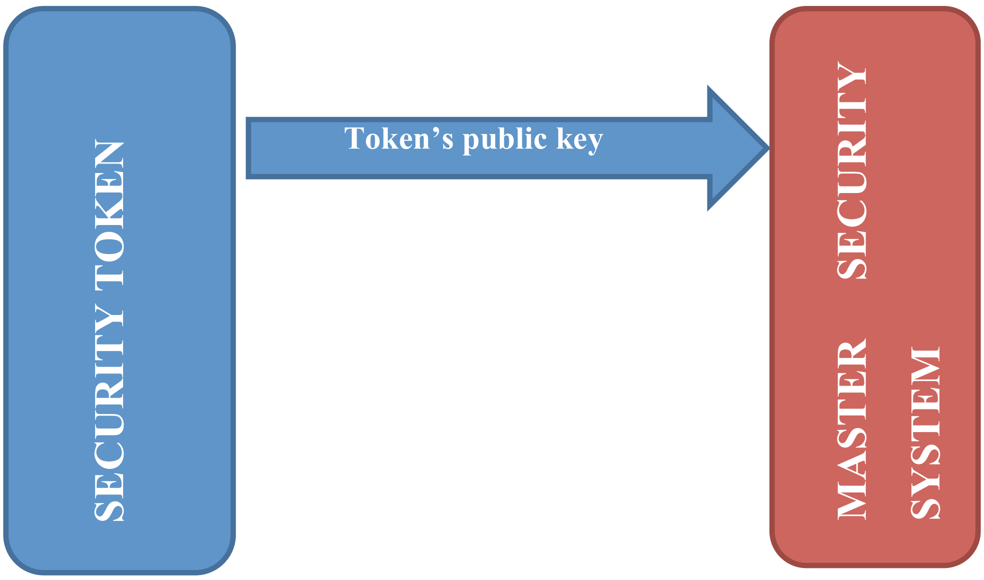


Fig. 5.3 - One way trust model

The token might content several key pairs for trust relationships with various security systems. The token with relationship works as the external database with user credentials and general rights.

PRACTICAL PART

6 SELF REPAIR ABILITIES

6.1 Hopfield neural network - fingerprints

Fig. 6.1 shows the tested samples. There were 3 different samples of the real fingerprint. Source pictures were digitalized and cleaned from noise and graphically mistakes. This cleaning and repairing wasn't effect the self-repair function of the Hopfield neural network. The cleaning operation was made only for showing the pictures.

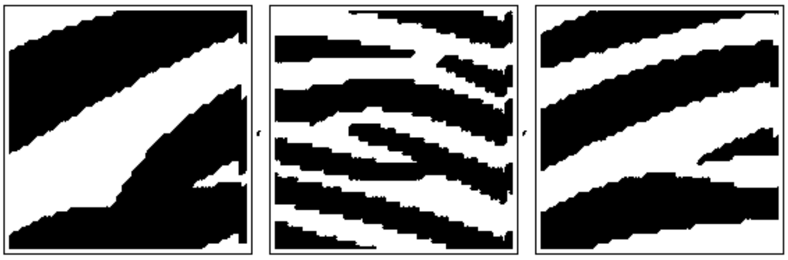


Fig. 6.1 - Testing samples

The second step was digitalization of used samples to monochromatic binary neuron structures. These processes was realized by custom C# language program which open the bmp picture and produce the data string contains structures binary values which represented each pixel of source image. The structured output shows Fig. 6.2.

{{0,1,1,1,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1…},{0,0,1,1,1,1,1,0,0,0,1,0,1,0,…},{…}}

Fig. 6.2 - Output string format

The third step was teaching of the Hopfield neural network with 3 testing samples.

After upper described steps, there was the fitted neural network, which is able to repair and recognize 3 samples of fingerprints. The simulation of critical factor, which produce noise or lower quality of the scanned finger, was realized with generated noise. The noise was generated as the uniform resolution noise. The next step was adding this noise to testing samples. Fig. 6.3 shows testing samples with 75% of noise. The left side of the picture shown below represents testing samples with 75% of noise. The right side represented the original clear samples for testing the self-repair function of the Hopfield neural network.

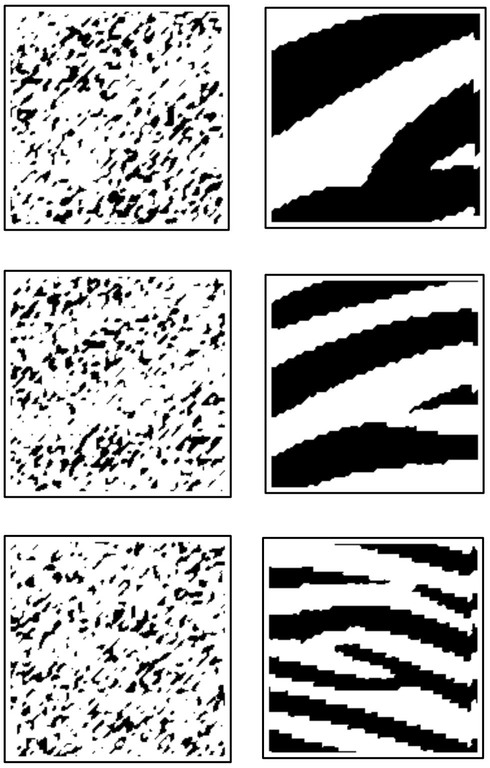


Fig. 6.3 -75% noise, testing samples

Tests were realized in the Wolfram Mathematica 8 simulation environment. Testing results shows Fig. 6.4. Samples with added noise are on the left part of picture shown below. The right part of picture shows the original clear sample. The middle part of this result shows repaired samples with Hopfield neural network.

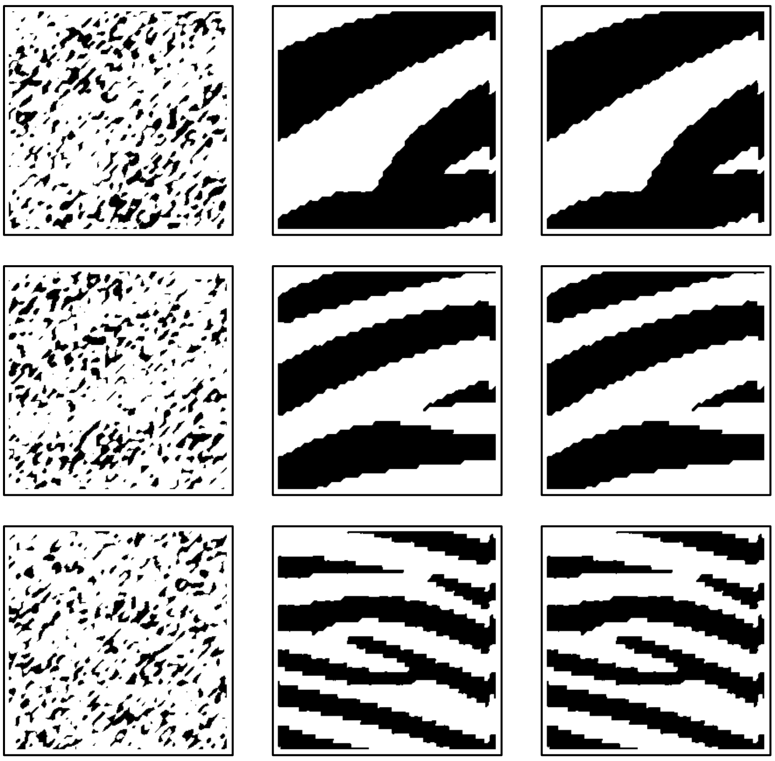


Fig. 6.4 - Testing result

The result shows, that the NN has a huge potential in self-repairing procedures. This potential might be used as the entry point to the next authentication procedure. The NN is able to pre-prepare the damaged sample for the main identification procedure. This step reduces the amount of data that must be inside the sample. The NN might repair unreadable data and the verification procedure receives a clear sample for the next identification process.

The self-repair ability is successful until the noise level is lower than 75%. This is flowing from the experimental testing with real fingerprints.

6.2 Hopfield neural network – face

The neural network for the picture reconstruction is based on the Hopfield neural network. The testing samples contained pictures of six different faces. There were some woman's and man's faces captured from different positions. These faces are shown on Fig. 6.5.

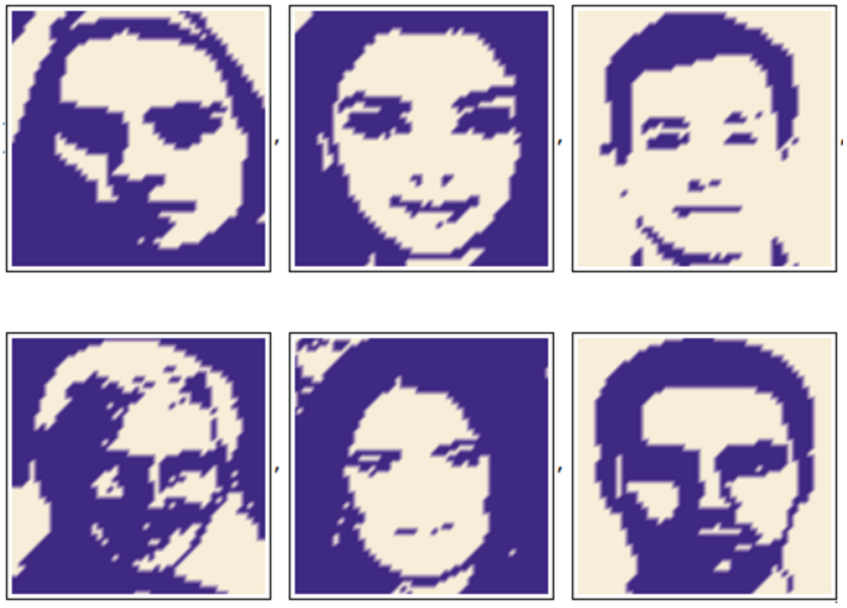


Fig. 6.5 - Source faces

6.2.1 Face reconstruction

The next was producing the images with the lower quality. There was used a noise for simulation the nature problems such as the rain, the fog and the limited visibility. The testing procedure contained the learned neural network (network was learned with the samples of faces). These experiments were realized 100 times and the followed picture represents the reconstructed samples. The Fig. 6.6 represents the minority noise. The noise does not affect each sample. Some samples are still relatively clear and processing system might identify it.

The next figure describes the samples with 40% of the noise. This level made marginally changes in some samples, especially the first and the second from the left in the second line. The identification of these samples is quite complicated, because these samples lost some of the specific biometric characteristics.

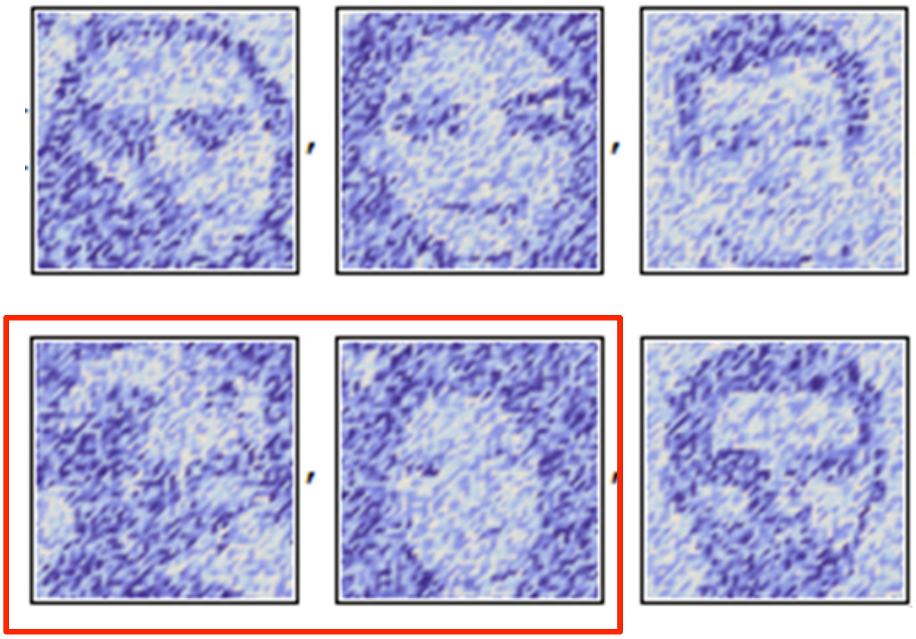


Fig. 6.6 - 40% on noise

The reconstruction showed in the Fig. 6.7 represents, that the neural network might repair these damaged samples to the original state.

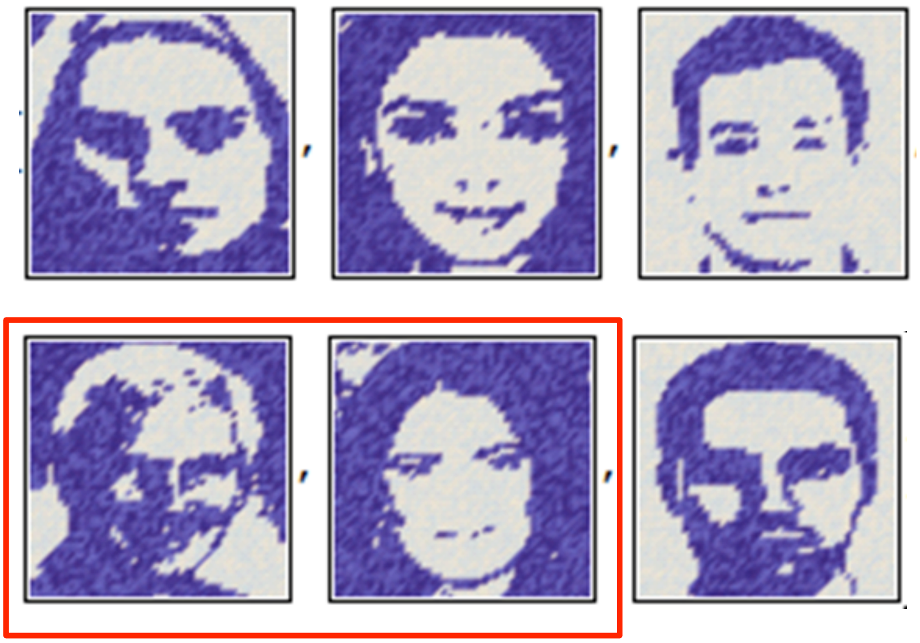


Fig. 6.7 - 40% of noise – reconstruction

The Fig. 6.8 shows the medium level of the noise at the 60% level. The source samples are damaged and the human operator does not recognize the original samples. The pixel comparison method does not work, because the samples lost marginal numbers of biometric identification markants [14]. The pictures are damaged.

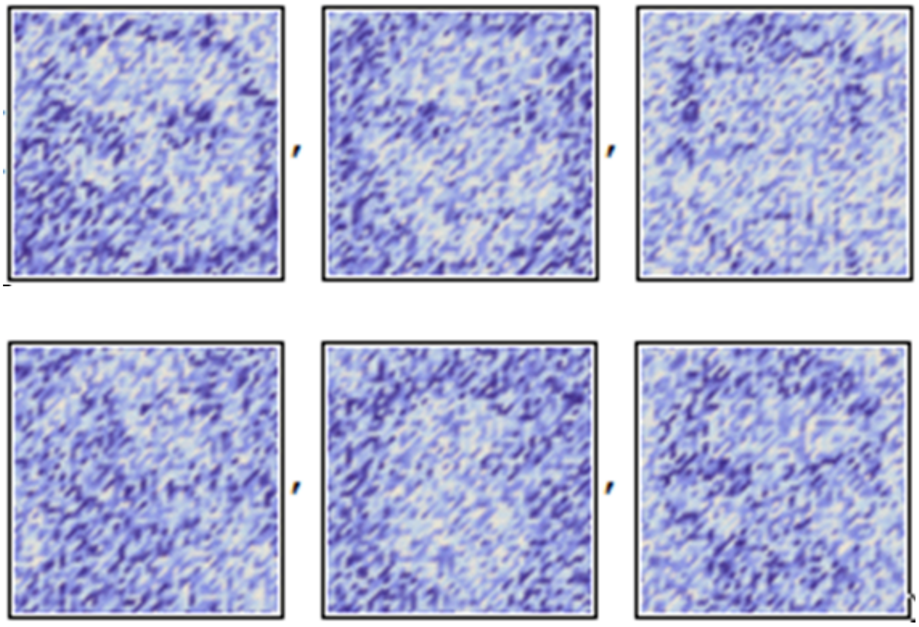


Fig. 6.8 - 60% of noise

The result from the neural network repair procedure is shown on the Fig. 6.9. The network could repair the original samples with extremely accuracy. The pictures were reconstructed to their original state. The identification system might analyse these samples and provide the positive or negative identification of selected users/peoples. The huge potential of the neural network self-repair function is representing by these results.

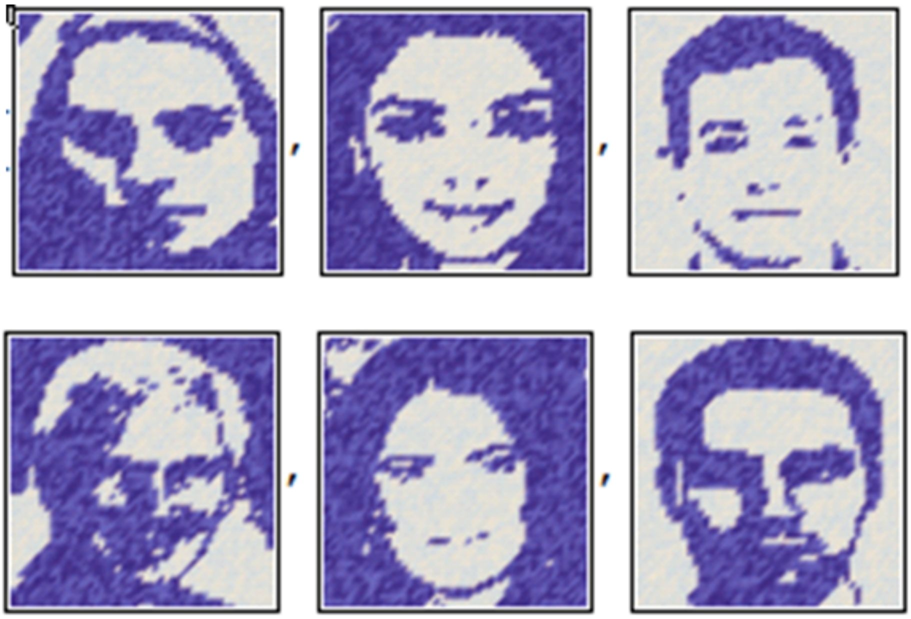


Fig. 6.9 - 60% of noise - reconstruction

7 CLASSIFICATION OF SAMPLES

7.1 Non-secure classification

The Neural network might be used as an identification element. This meant that this NN has output layer that specify which sample was at the entry layer. The non-secure classification is based on the 1 bit code differences between samples.

Table 7.1 - Non-secure classification

Input binary layer	Output layer
{1,1,1,0,0,1,0,1,0,1,0,0,1,1,0,1,0,1,1,0,…..}	**{0,0,0,1}**
{1,0,0,0,0,1,0,1,0,1,1,1,1,1,0,1,0,1,1,0,…..}	**{0,0,1,0}**
{1,1,0,0,0,1,0,1,0,1,0,1,1,1,0,1,0,0,1,0,…..}	**{0,0,1,1}**
{1,1,1,0,0,1,0,1,0,1,1,1,1,0,0,1,0,1,0,0,…..}	**{0,1,0,0}**
{0,1,1,0,0,0,0,1,0,1,1,1,1,1,0,1,0,0,1,0,…..}	**{0,1,0,1}**

This method for classification has a big security problem. The problem is inside the code differences between neighbour samples. If there is a huge damage or the sample is quite similar (not the same), the classification might jump to the known sample and verification process is corrupted. The solution of this issue is in securing the output code by some redundant code.

7.2 Securing the code by BCH code

Upper described problem is solved by the adding the redundant information to the output from the layer. There was chosen the BCH code with 31 bit length and 15 bit redundant bits, known as BCH 31/15. This code might detect the corrupted string and if the amount of the corrupted bits is in limit, the code might correct the sample. So the output layer does not have only 1 bit differences and the verification process could not be corrupted by the 1 corrupted bit in the output layer of NN.

Table 7.2 - Secure classification by the BCH code

Output layer BCH
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

Table 7.2 shows the incremental counter for generation user ID number. The differences between the number is 1(in decimal non-secure notification), but inside the NN there is greater bit difference.

8 NEURAL NETWORK FINGERPRINT IDENTIFICATION

This identification is based on the multi-layer neural network. This network is trained to recognize specific samples of fingerprint pictures. The pictures are converted to the B/W colour pictures. The next step is conversion of picture pixels to their binary neuron representation. The process is adapted to the Double value neurons. Each 64-bit neuron is converted to one Double value. This value represented the one input neuron to the designed network. The samples are scanned by the Biomini fingerprint scanner. The size of samples is fixed to 280x320 pixels. This provide 89 600 binary pixel with value 0 or 1. There were 1400 Double value neurons after conversion raw binary neuron to the Double value testing neurons. After this, there was a training set to neural network. The output is generated as the ID code secured by BCH code. This process is shown on Fig. 8.1.

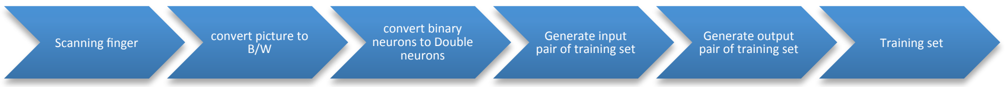

Scanning finger
convert picture to B/W
convert binary neurons to Double neurons
Generate input pair of training set
Generate output pair of training set
Training set

Fig. 8.1 - Training set generation process

The training set contains the input vectors represented by the Double neurons loaded from the fingerprint and the BCH code of ID.

Type of the network was chosen as a multilayer network with variable number of layers. The minimal number of layers is 3 that represented the input layer for the picture information, one hidden layer and output layer with one Double neuron with the secured ID of fingerprint. Practical part of this process is shown on the Fig. 8.2.

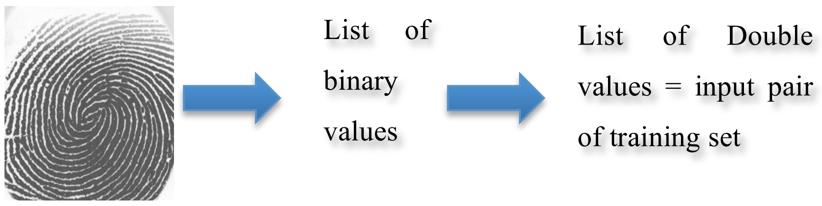


Fig. 8.2 - Input pair generation process

8.1 Training process definition

Training process has 2 ending mechanisms; the first of them is the acceptable error of training. This value s super user defined during the training process. The second stopping mechanism is the difference between training errors. The system supported the set of this difference for the emergency stopping mechanism. If there are too les number of neurons inside the hidden layer or layers, this stopped the infinite loop of the training process. Thus this mechanism must receive 30 differences with less value than user defined in a row. After this, the training is stopped.

The default training error was set to 0. 000 000 1 or less in each tests described inside this work. This value represented the allowed differences between the binary value of the ideal value and the value received from the neural network.

8.2 Training progress

The training progress is represented by the graphical sequence of training error during the process. The progress is visualised as a graph with the training error during the training process. The second graph represented the differences between steps. This differences was computed a difference between the last training error and the actual training error.

The last part of training process information is the table with the time and statistic information which is flowing from the simulation.

The next chapters represent the experimental output affording on the number of layers and number of neurons in hidden layers. Each experiment has own table with the settings and statistic information and graphical output from the testing procedures.

8.2.1 Progress – one hidden layer with 30 neurons

Table 8.1 - Training L30

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	30 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 ($1 . 10^{-7}$)
Acceptable training error difference	0.0000001 ($1 . 10^{-7}$)
No. training epoch	417
Training duration	00:00:06.23 (H:M:S)
Last training error value	$1.8624 . 10^{-9}$

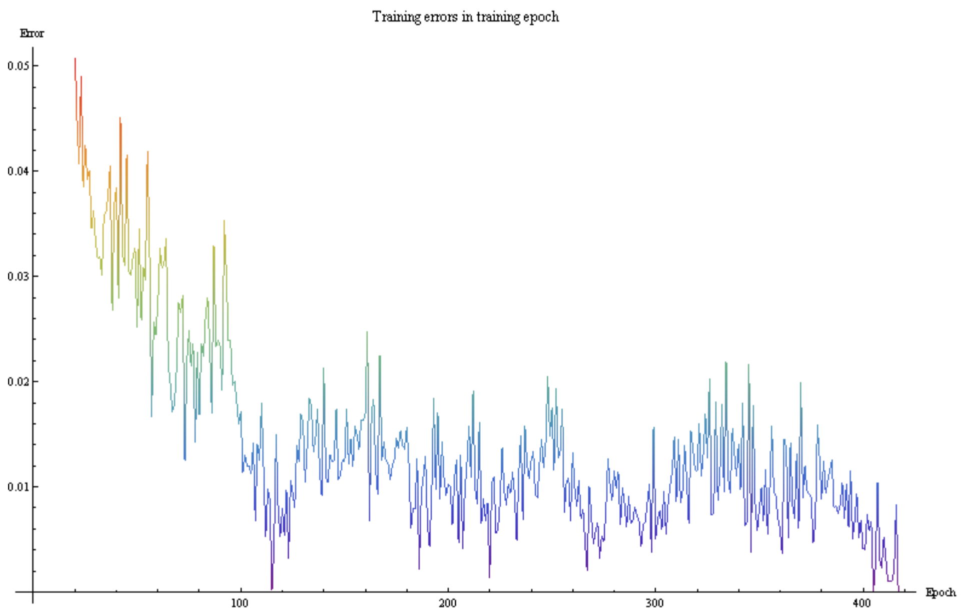


Fig. 8.3 - Training error L30

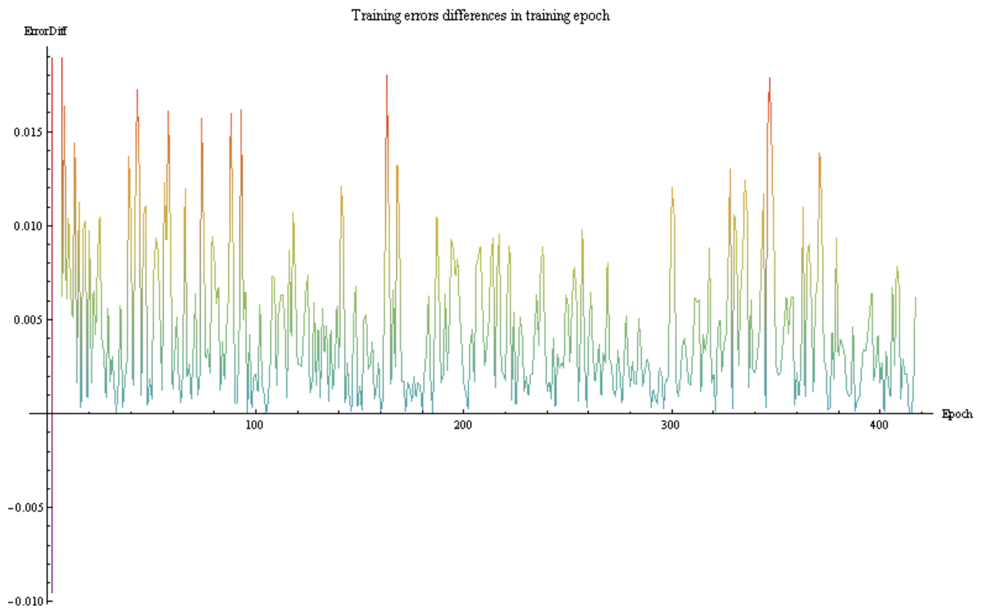


Fig. 8.4 - Training error difference L30

8.2.2 Progress – one hidden layer with 50 neurons

Table 8.2 - Training L50

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	50 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 (1 . 10 $^{-7}$)
Acceptable training error difference	0.0000001 (1 . 10 $^{-7}$)
No. training epoch	136
Training duration	00:00:03.47 (H:M:S)
Last training error value	7.2884 . 10^{-8}

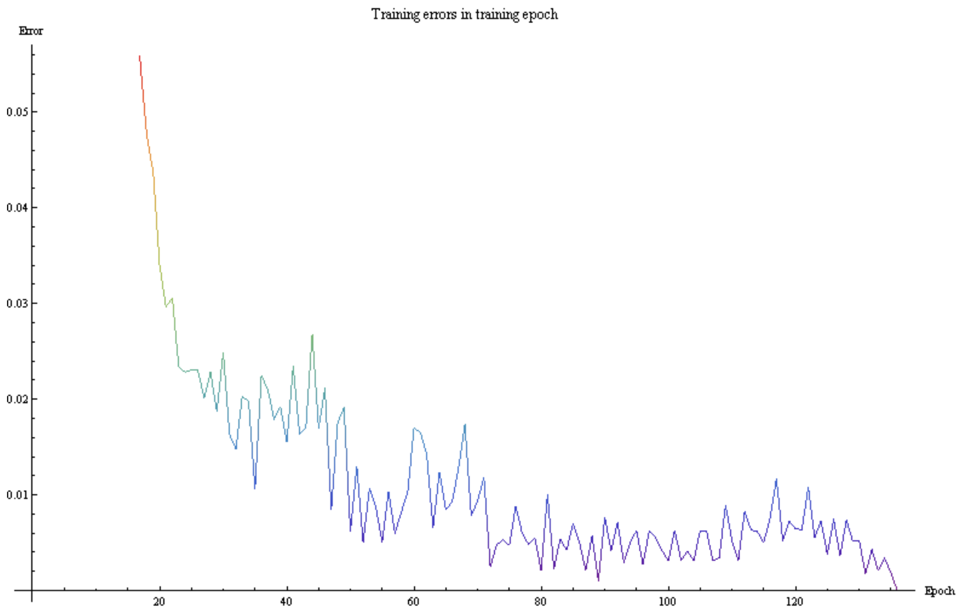


Fig. 8.5 - Training error L50

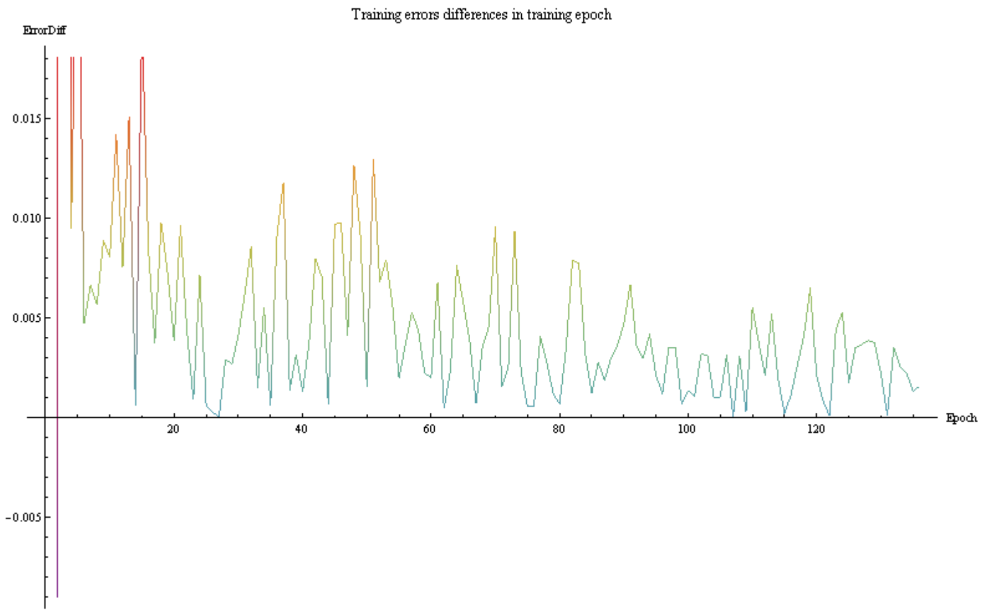


Fig. 8.6 - Training error difference L50

8.2.3 Progress – one hidden layer with 100 neurons

Table 8.3 - Training L100

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	100 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 (1 . 10 $^{-7}$)
Acceptable training error difference	0.0000001 (1 . 10 $^{-7}$)
No. training epoch	42
Training duration	00:00:02.37 (H:M:S)
Last training error value	6.9461 . 10^{-11}

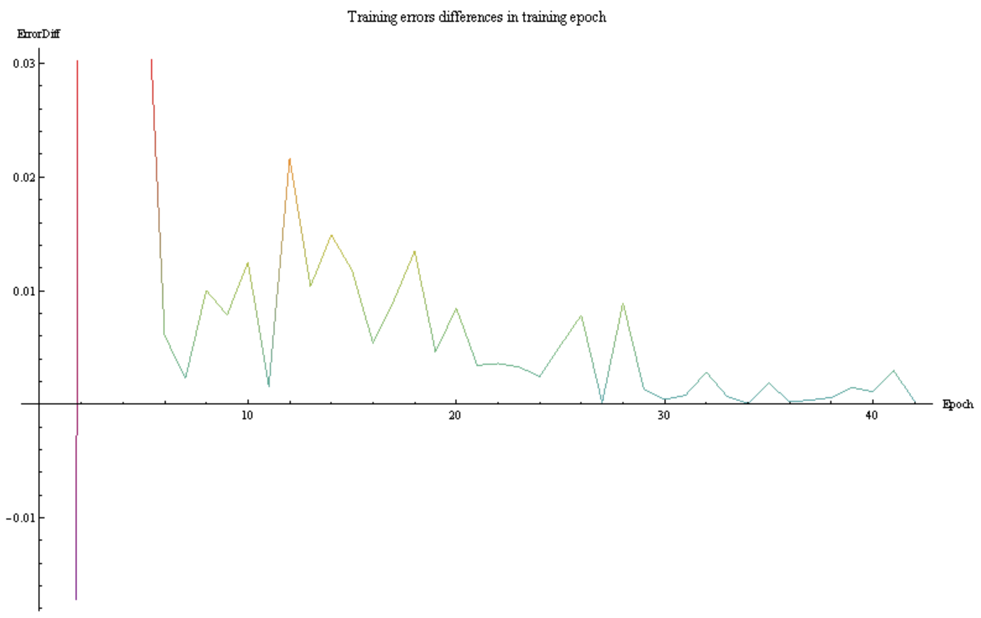


Fig. 8.7 - Training error L100

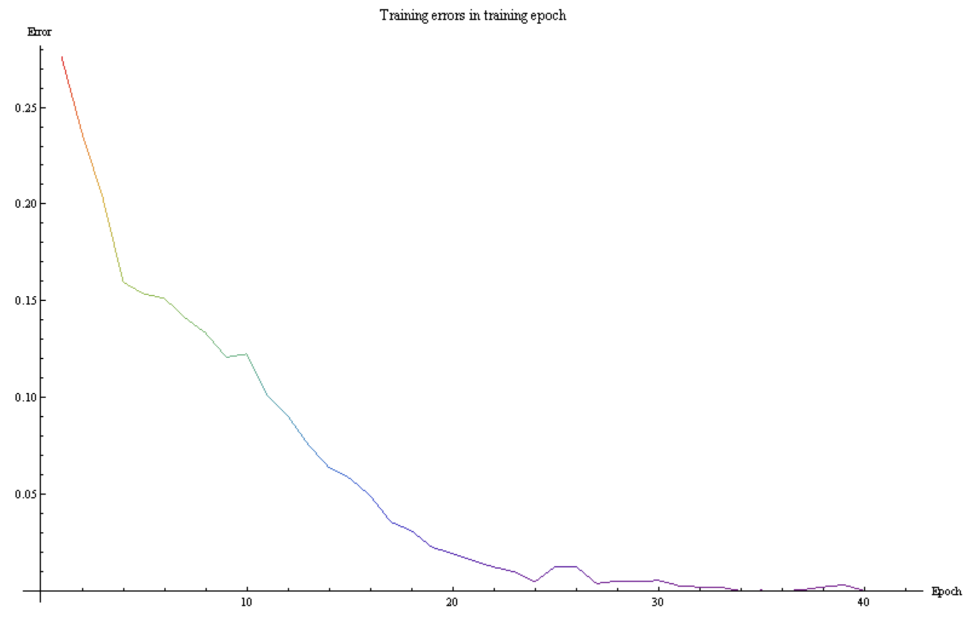


Fig. 8.8 - Training error differences L100

8.2.4 Progress – one hidden layer with 1000 neurons

Table 8.4 - Training L1000

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	1000 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 (1 . 10 $^{-7}$)
Acceptable training error difference	0.0000001 (1 . 10 $^{-7}$)
No. training epoch	32
Training duration	00:00:06.2273562 (H:M:S)
Last training error value	9.4832 . 10^{-8}

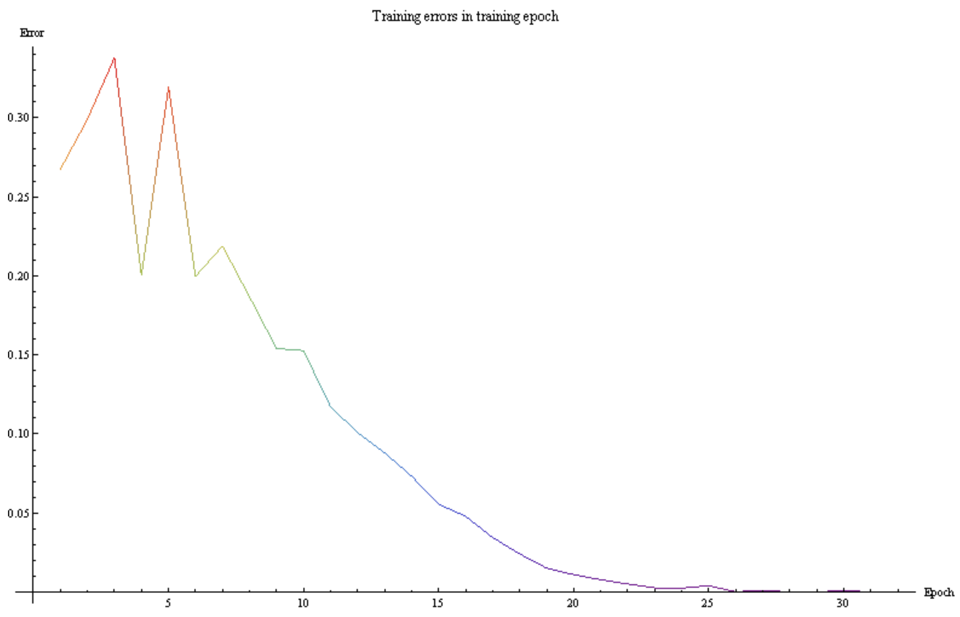


Fig. 8.9 - Training error L1000

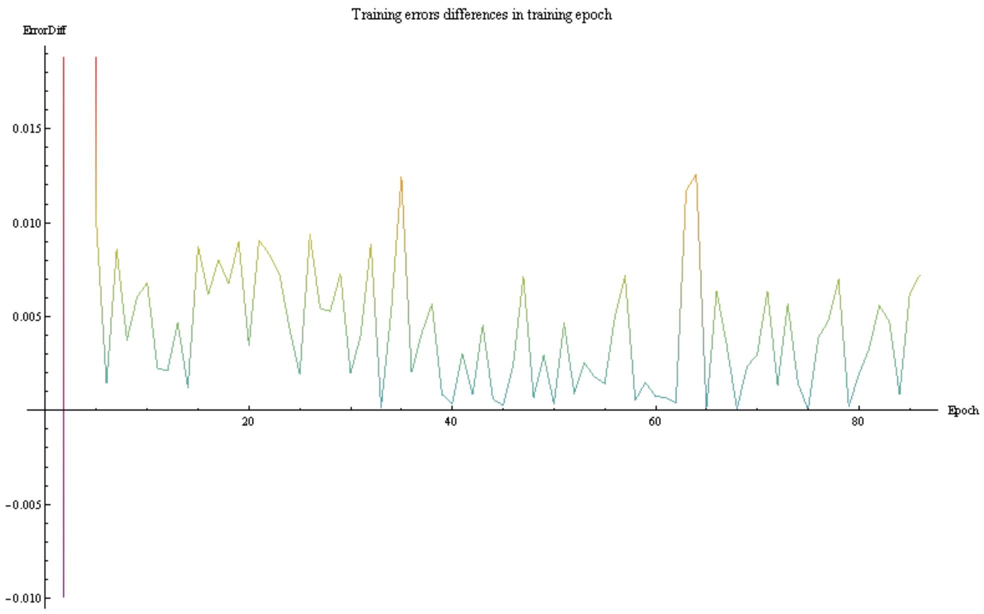


Fig. 8.10 - Training error differences L1000

8.2.5 Progress – two hidden layer with 500,100 neurons

Table 8.5 - Training L500, L100

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	500 neurons
2. hidden layer	100 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 (1 . 10 $^{-7}$)
Acceptable training error difference	0.0000001 (1 . 10 $^{-7}$)
No. training epoch	86
Training duration	00:00:49.30 (H:M:S)
Last training error value	1.0095 . 10^{-8}

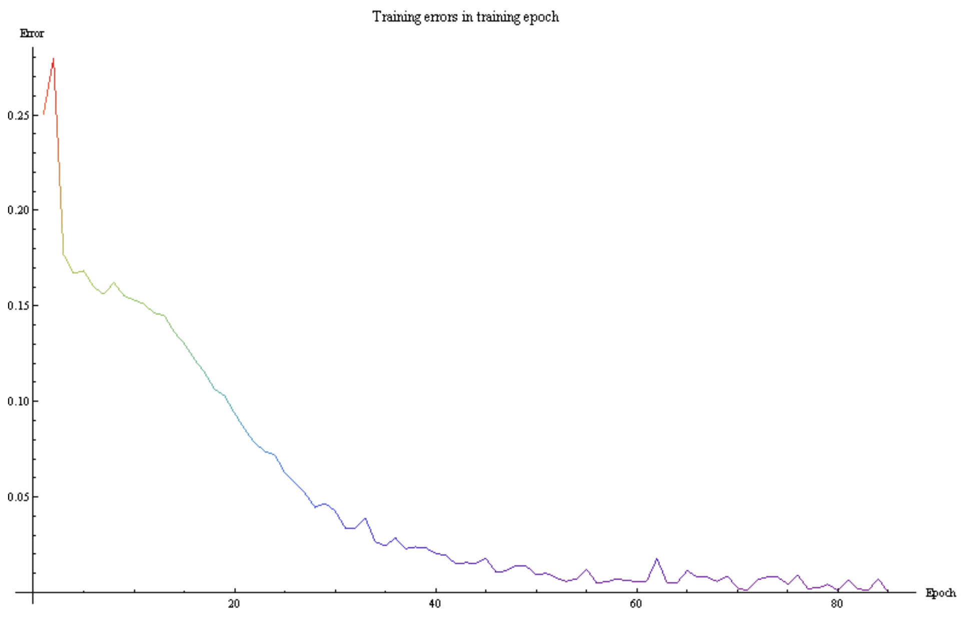


Fig. 8.11 - Training error L500, L100

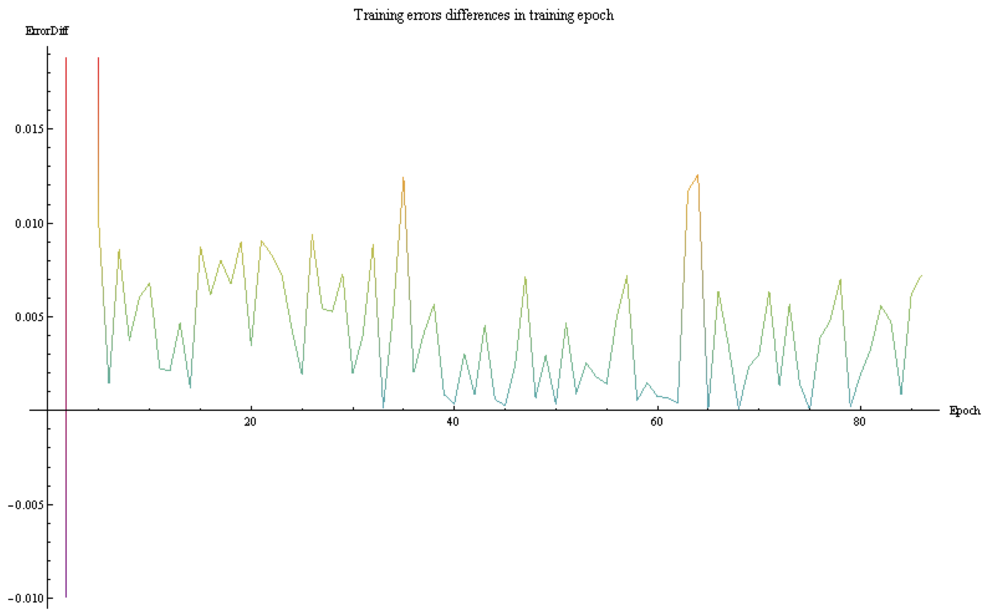


Fig. 8.12 - Training error differences L500, L100

8.2.6 Progress – two hidden layer with 750,300 neurons

Table 8.6 - Training L750, L300

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	750 neurons
2. hidden layer	300 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 (1 . 10 $^{-7}$)
Acceptable training error difference	0.0000001 (1 . 10 $^{-7}$)
No. training epoch	81
Training duration	00:01:21.46 (H:M:S)
Last training error value	6.3390 . 10^{-12}

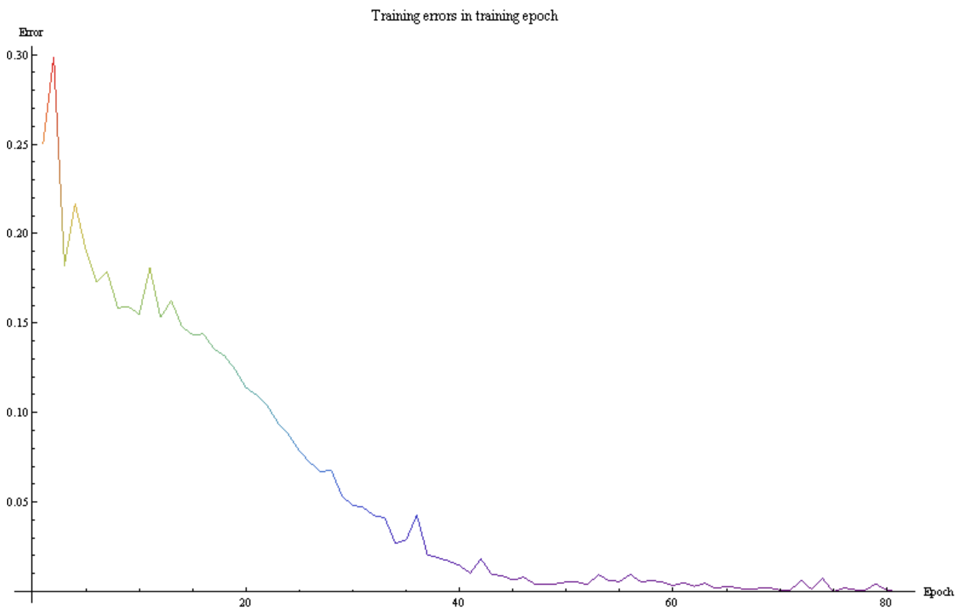


Fig. 8.13 - Training error L750, L300

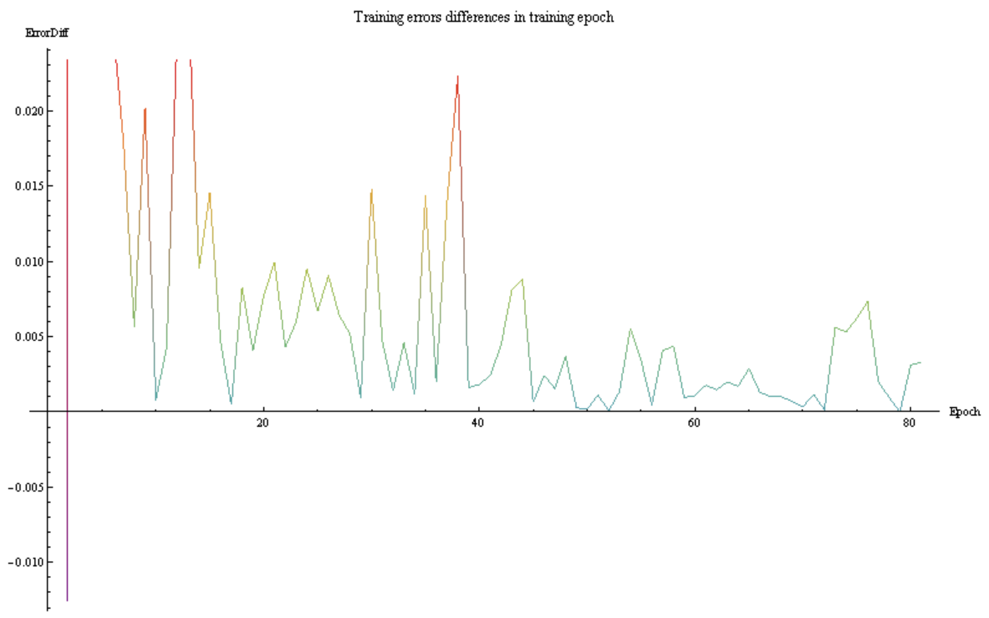


Fig. 8.14 - Training error differences L750, L300

8.2.7 Progress – two hidden layer with 1000,500 neurons

Table 8.7 - Training L1000, L500

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	1000 neurons
2. hidden layer	500 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 ($1 . 10^{-7}$)
Acceptable training error difference	0.0000001 ($1 . 10^{-7}$)
No. training epoch	97
Training duration	00:02:29.17 (H:M:S)
Last training error value	$3.2802 . 10^{-16}$

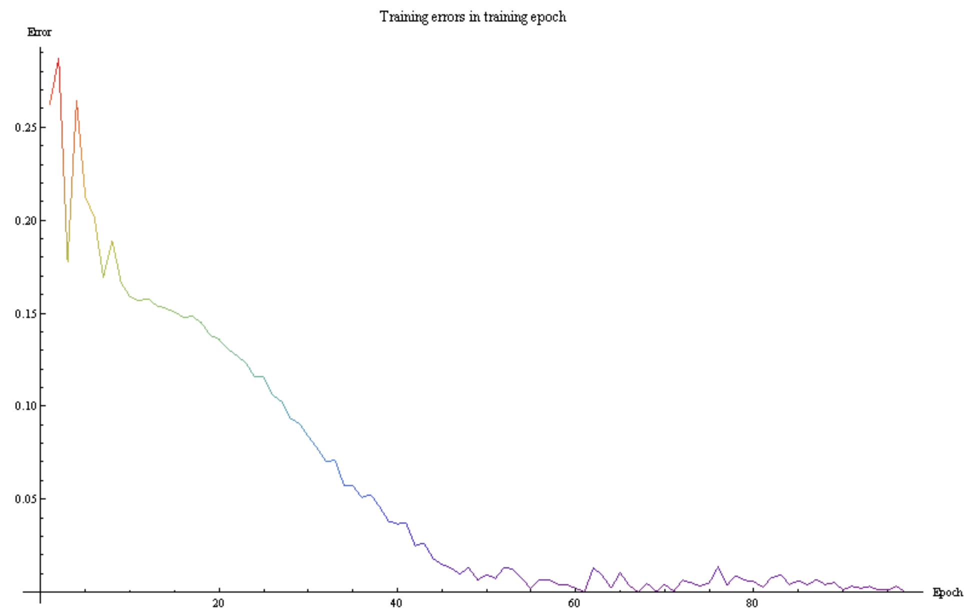


Fig. 8.15 - Training error L1000, L500

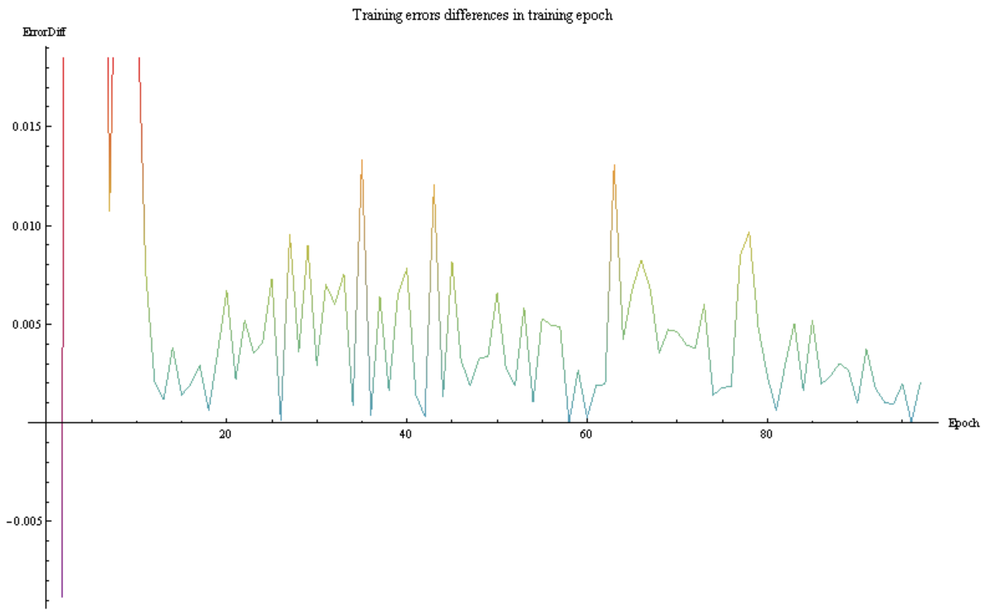


Fig. 8.16 - Training error differences L1000, L500

8.2.8 Progress – three hidden layer with 1000,1500,1000 neurons

Table 8.8 - Training L1000, L1500, L1000

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	1000 neurons
2. hidden layer	1500 neurons
3. hidden layer	1000 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 $(1 . 10^{-7})$
Acceptable training error difference	0.0000001 $(1 . 10^{-7})$
No. training epoch	602
Training duration	00:35:07.87 (H:M:S)
Last training error value	$1.3241 . 10^{-14}$

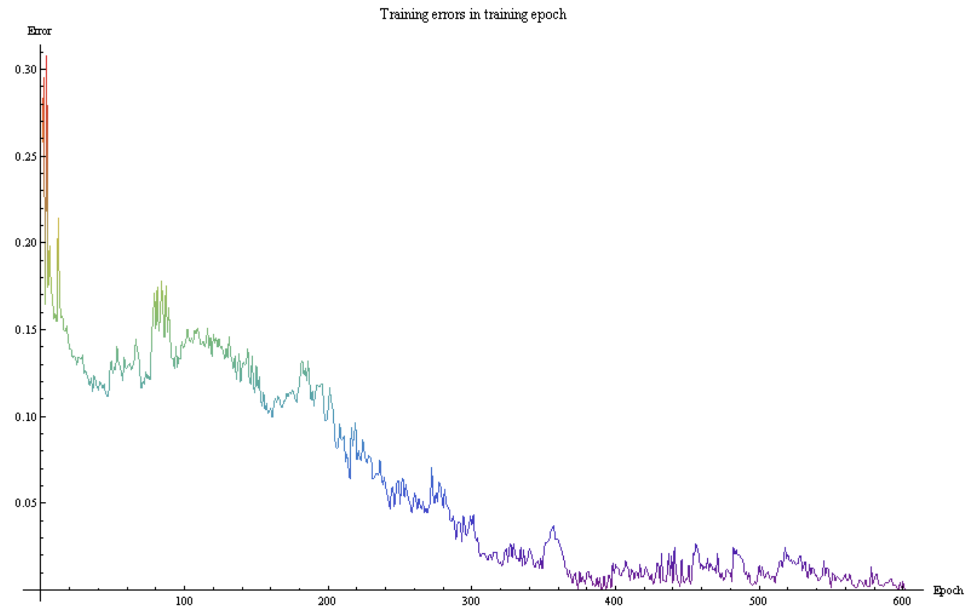


Fig. 8.17 - Training error L1000, L1500, L1000

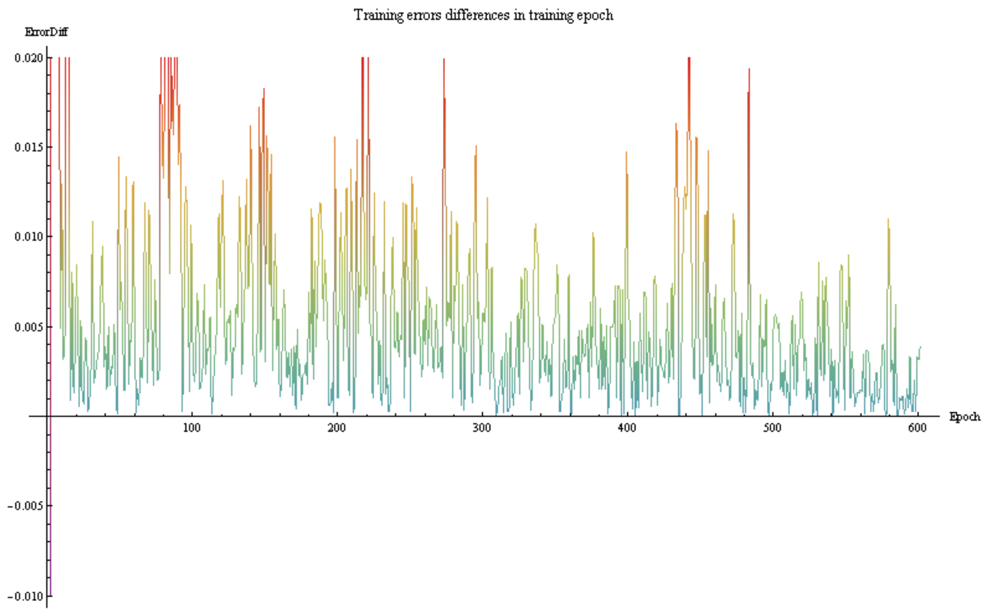


Fig. 8.18 - Training error differences L1000, L1500, L1000

8.2.9 Progress – three hidden layer with 1000,3000,1000 neurons

Table 8.9 - Training L1000, L3000, L1000

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	1000 neurons
2. hidden layer	3000 neurons
3. hidden layer	1000 neurons
Output layer	31 neurons
Acceptable training error	0.0000001 ($1 . 10^{-7}$)
Acceptable training error difference	0.0000001 ($1 . 10^{-7}$)
No. training epoch	148
Training duration	00:15:02.23 (H:M:S)
Last training error value	$1.3981 . 10^{-8}$

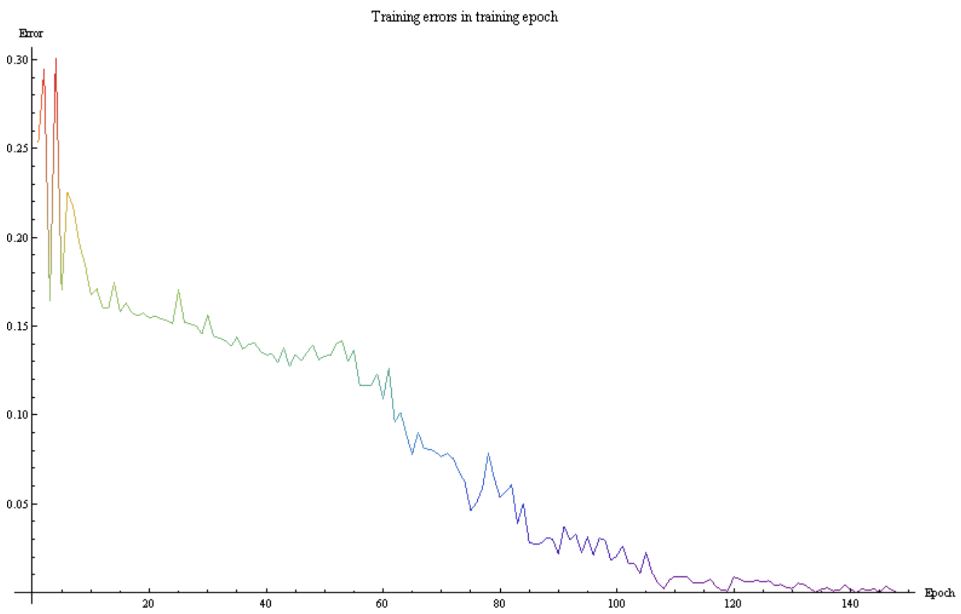


Fig. 8.19 - Training error L1000, L3000, L1000

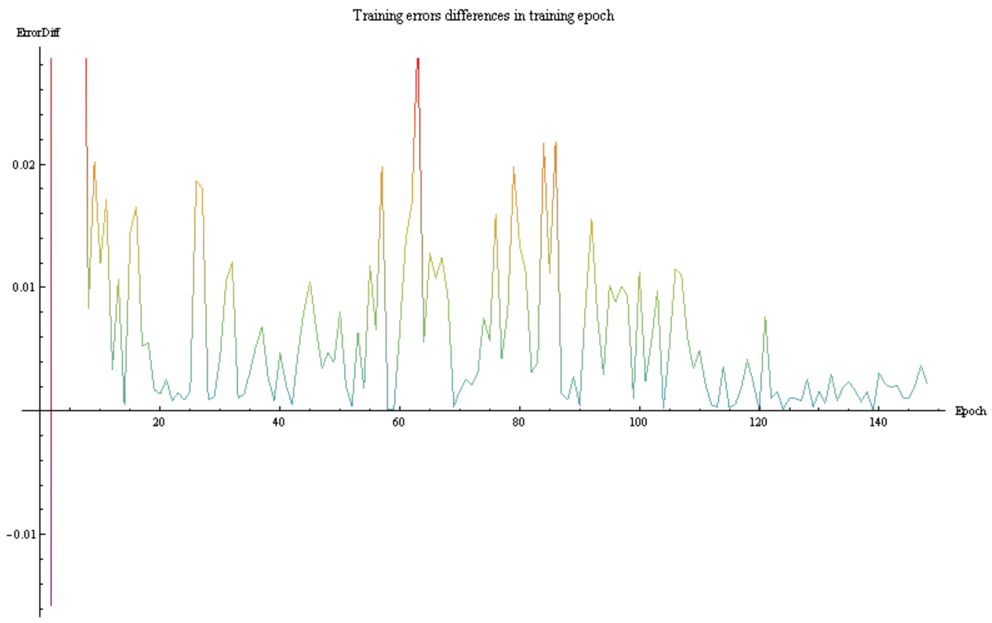


Fig. 8.20 - Training error differences L1000, L3000, L1000

8.3 Results – usability of NN

The output from the previously described steps is the trained neural network with high level of self-repairing function. This NN is able to identify the known fingerprint samples. This network is stored into SQLite database and used for the verification of people. There are two different usability of the NN; the first is inside using the one neural network for all fingerprint samples, which system must detect and successfully identify. This process required that system administrator must have the whole database of fingerprint identification pictures. Next there was a network that is trained with these specific samples. This process is applicable in user environment with zero or minimal user migration.

The second usability is flowing from the using one neural network for each user. The network is able to identify only the owner of this NN. This allows using the network as a portable verification token. Each user has specific neural network that will be trained with each finger. The user security token contains the user neural network. Verification network is the portable. The results of the identification process are detailed described in APPENDIX B.

9 NEURAL NETWORK FACE IDENTIFICATION

The face identification process is particularly similar than the fingerprint identification. The source data are similar. There is a picture captured by the camera. This picture represents the face of the scanned user. If there is a noise or lower light environment, the picture cannot be compare with the original sample stored in internal database pixel by pixel. The first potentially issues is flowing from the partial changes of the face; for example the beard. The next problem of identification is the different position of scanned face. The neural network might reconstruct this face and it is less addicted on the quality of pictures. There is a place for a few error limits, which cannot change the result of the verification procedure.

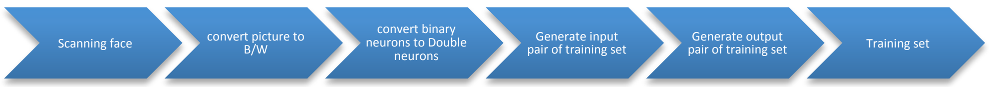

Scanning face
convert picture to B/W
convert binary neurons to Double neurons
Generate input pair of training set
Generate output pair of training set
Training set

Fig. 9.1 - Training set generation - face

The identification engine uses the similar principle as the fingerprint identification process. Firstly, there is a scanning of face; the webcam build in notebook or other portable device realizes this operation. This process produces the colour picture with the face. The next operation is conversion to the B/W colour pallet and creating binary neurons. Binary neurons are converted to the double input neurons and added to the training set as inputs. The ID number secured with the BCH code represents the outputs. The procedure of creating training set is shown on Fig. 9.2.

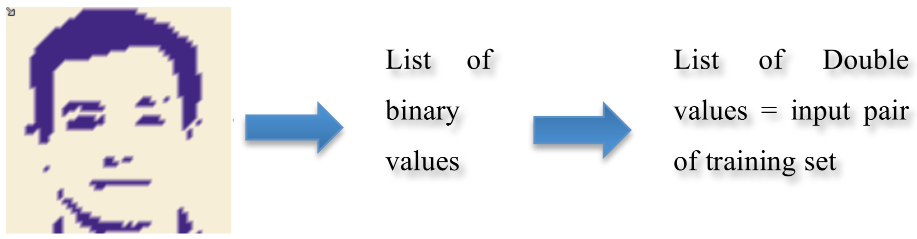


Fig. 9.2 - Input pair generation process

9.1 Training process definition

Training process has 2 ending function; the first of them is the acceptable error of training. This value s super user defined during the training process. The second stopping mechanism is the difference between training errors. The system supported the set of this difference for the emergency stopping mechanism. If there are too les number of neurons inside the hidden layer or layers, this stopped the infinite loop of the training process. Thus this mechanism must receive 30 differences with less value than user defined in a row. After this, the training is stopped.

The default training error was set to 0. 000 000 01 or less in each tests described inside this work. This value represented the allowed differences between the binary value of the ideal value and the value received from the NN.

9.2 Training progress

The training progress is represented by the graphical sequence of training error during the process. The progress is visualised as a graph with the training error during the training process. The second graph represented the differences between steps. This differences was computed a difference between the last training error and the actual training error.

The last part of training process information is the table with the time and statistic information which is flowing from the simulation.

The next chapters represent the experimental output affording on the number of layers and number of neurons in hidden layers. Each experiment has own table with the settings and statistic information and graphical output from the testing procedures.

9.2.1 Progress – one hidden layer with 30 neurons

Table 9.1 - Training L30

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	30 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 (1 . 10 $^{-8}$)
Acceptable training error difference	0.00000001 (1 . 10 $^{-8}$)
No. training epoch	887
Training duration	00:00:15.10 (H:M:S)
Last training error value	3.2967 . 10^{-17}

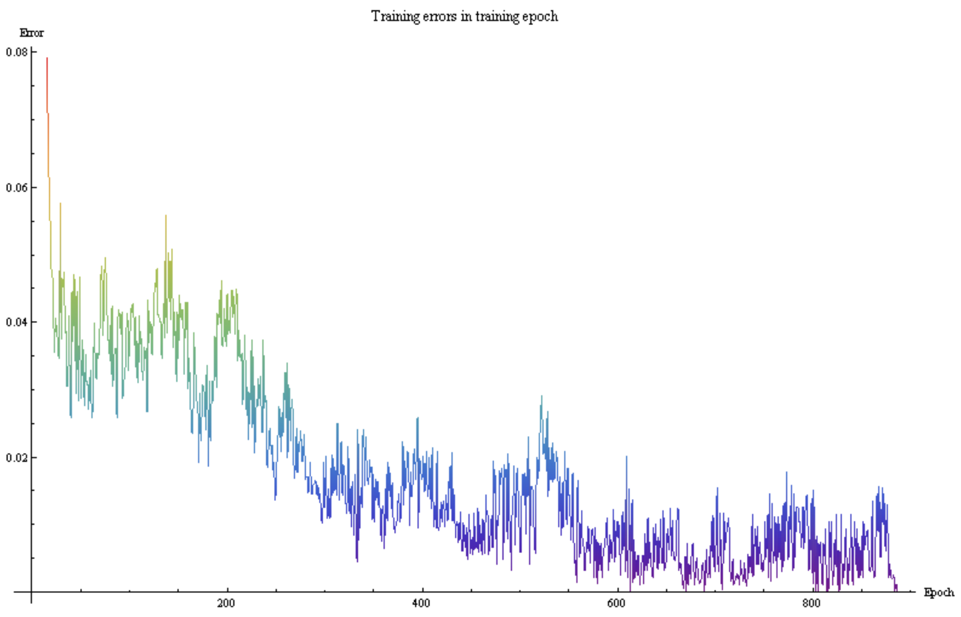


Fig. 9.3 - Training error L30

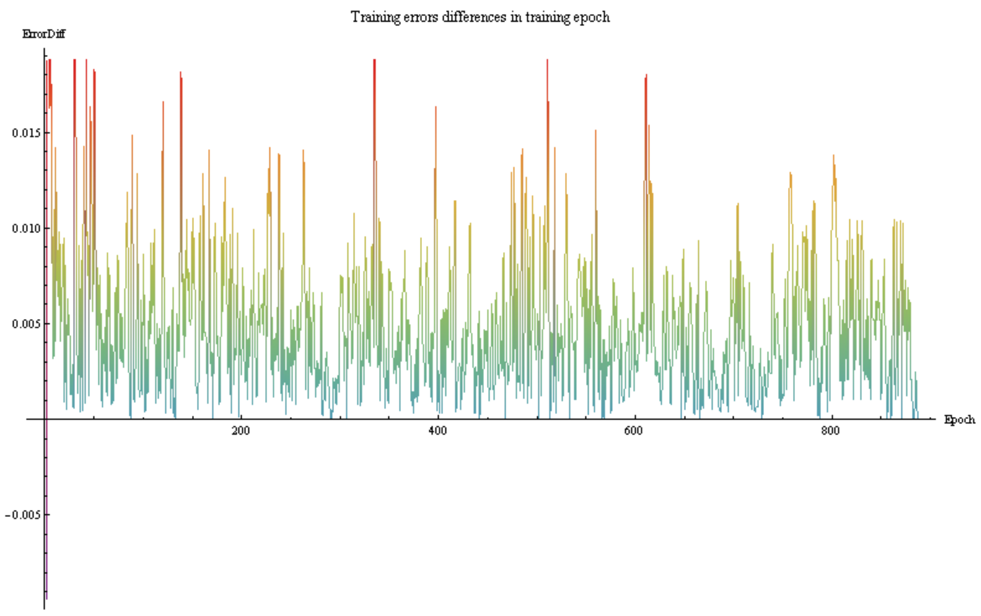


Fig. 9.4 - Training error differences L30

9.2.2 Progress – one hidden layer with 50 neurons

Table 9.2 - Training L50

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	50 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 ($1 . 10^{-8}$)
Acceptable training error difference	0.00000001 ($1 . 10^{-8}$)
No. training epoch	104
Training duration	00:00:03.24 (H:M:S)
Last training error value	$7.0081 . 10^{-18}$

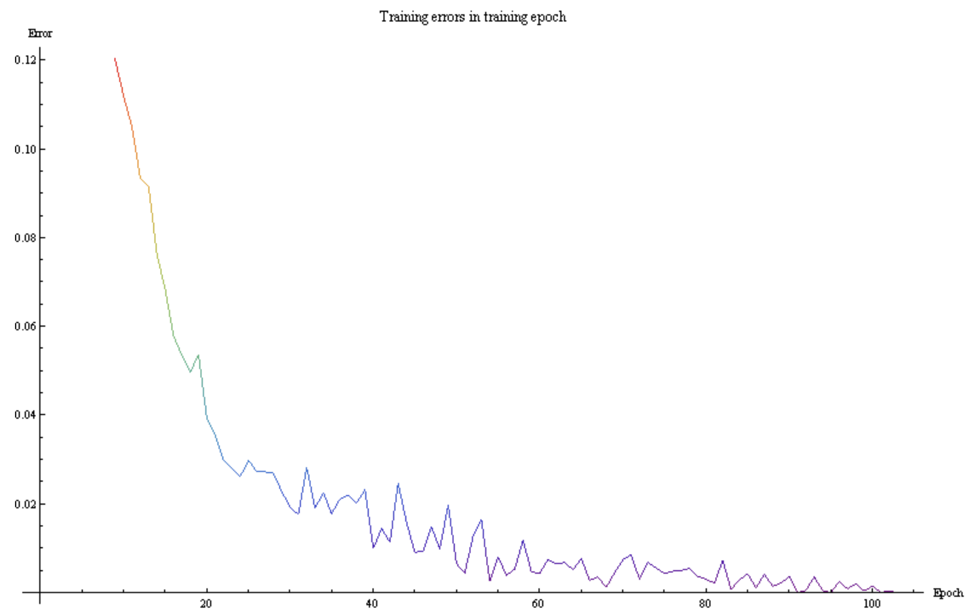


Fig. 9.5 - Training error L50

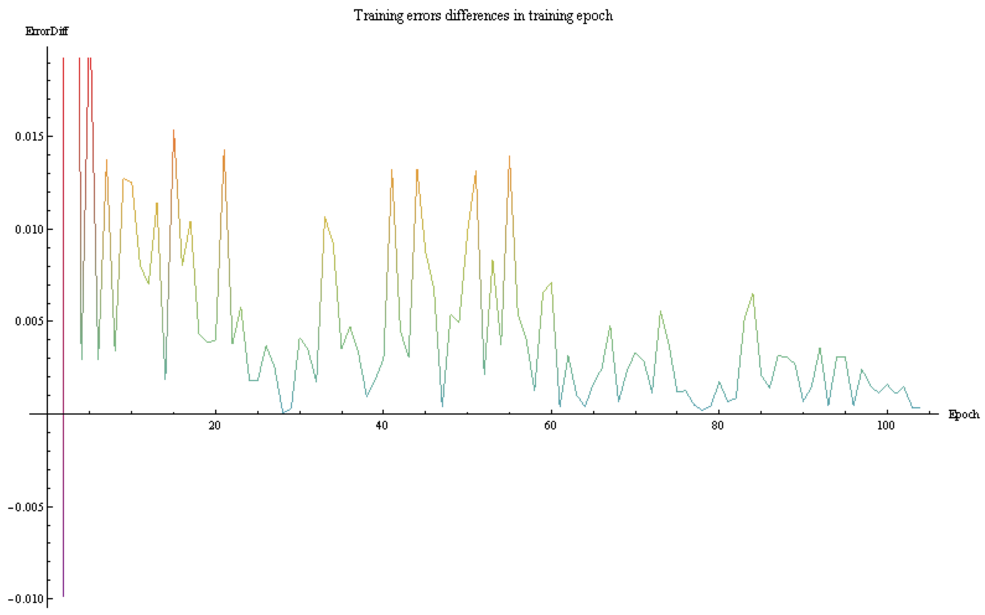


Fig. 9.6 - Training error differences L50

9.2.3 Progress – one hidden layer with 100 neurons

Table 9.3 - Training L100

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	100 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 ($1 . 10^{-8}$)
Acceptable training error difference	0.00000001 ($1 . 10^{-8}$)
No. training epoch	55
Training duration	00:00:03.33 (H:M:S)
Last training error value	$3.4690 . 10^{-10}$

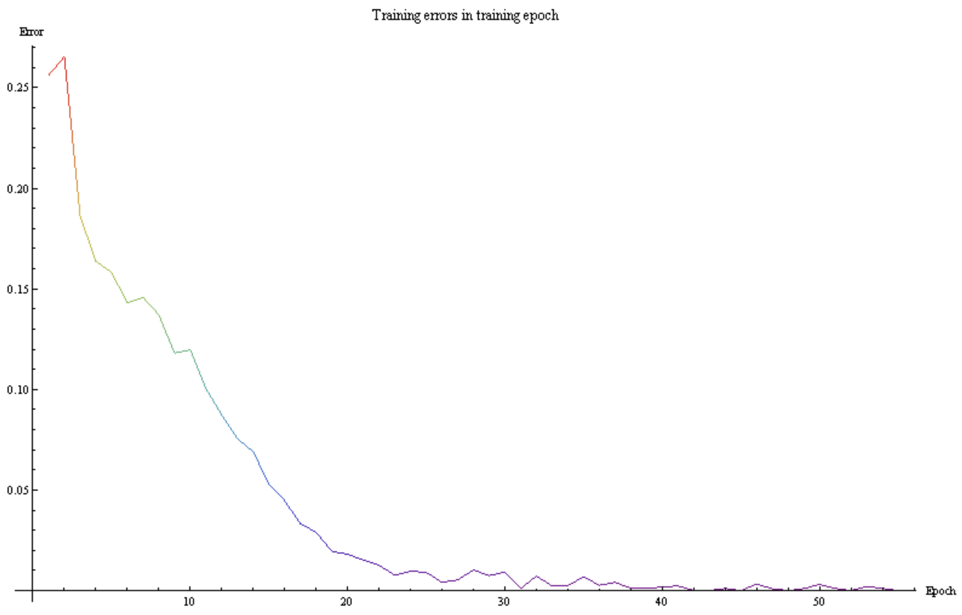


Fig. 9.7 - Training error L100

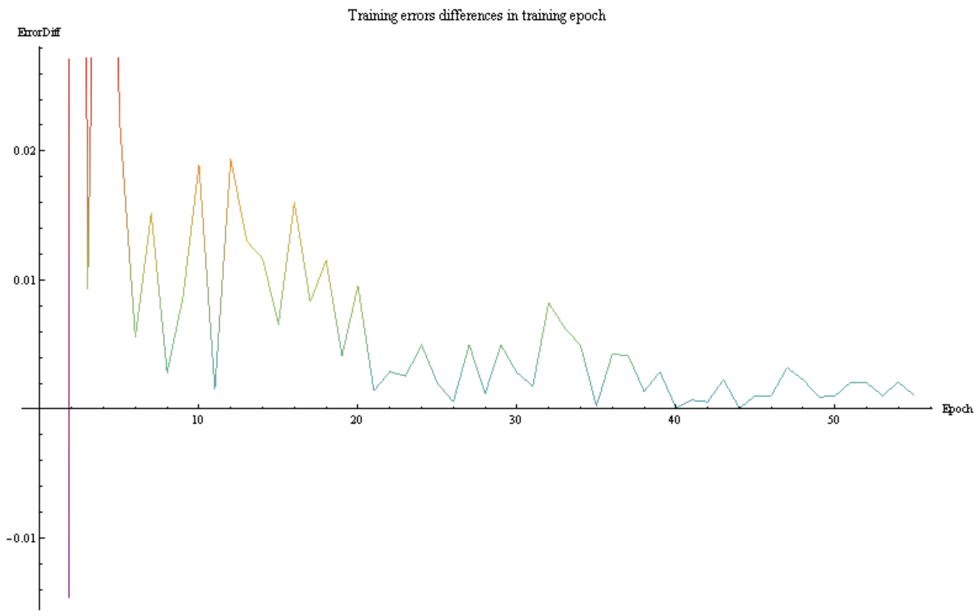


Fig. 9.8 - Training error differences L100

9.2.4 Progress – one hidden layer with 1000 neurons

Table 9.4 - Training L1000

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	1000 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 (1 . 10 $^{-8}$)
Acceptable training error difference	0.00000001 (1 . 10 $^{-8}$)
No. training epoch	30
Training duration	00:00:41.79 (H:M:S)
Last training error value	5.4860 . 10^{-9}

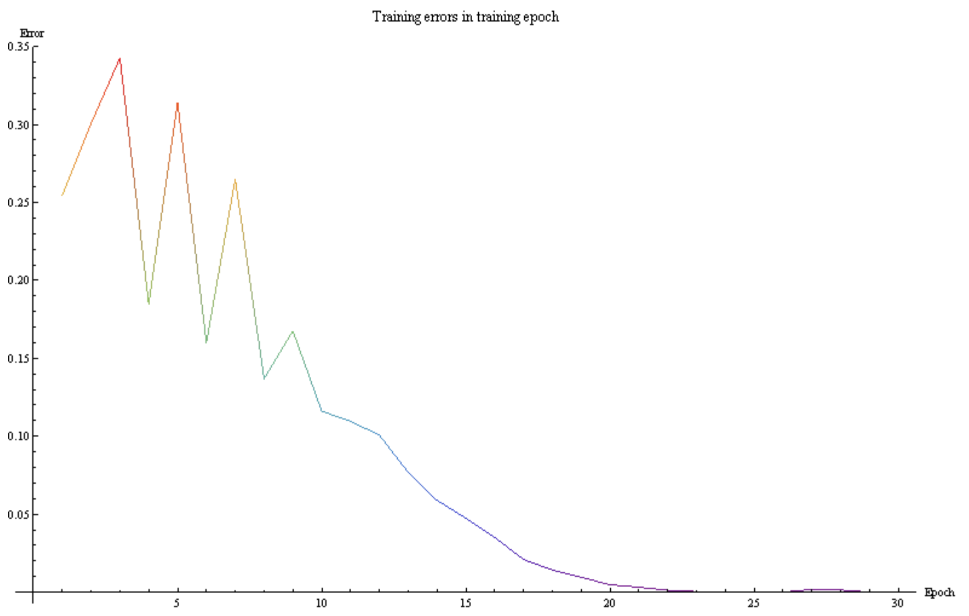


Fig. 9.9 - Training error L1000

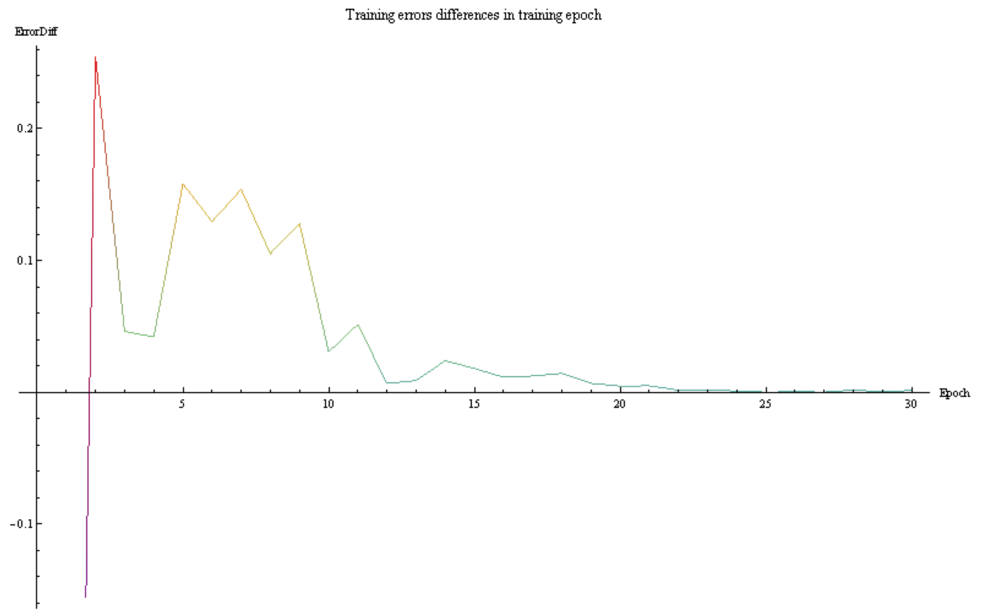


Fig. 9.10 - Training error differences L1000

9.2.5 Progress – two hidden layer with 500,100 neurons

Table 9.5 - Training L500, L100

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	500 neurons
2. hidden layer	100 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 (1 . 10 $^{-8}$)
Acceptable training error difference	0.00000001 (1 . 10 $^{-8}$)
No. training epoch	123
Training duration	00:01:16.46 (H:M:S)
Last training error value	1.2855 . 10^{-10}

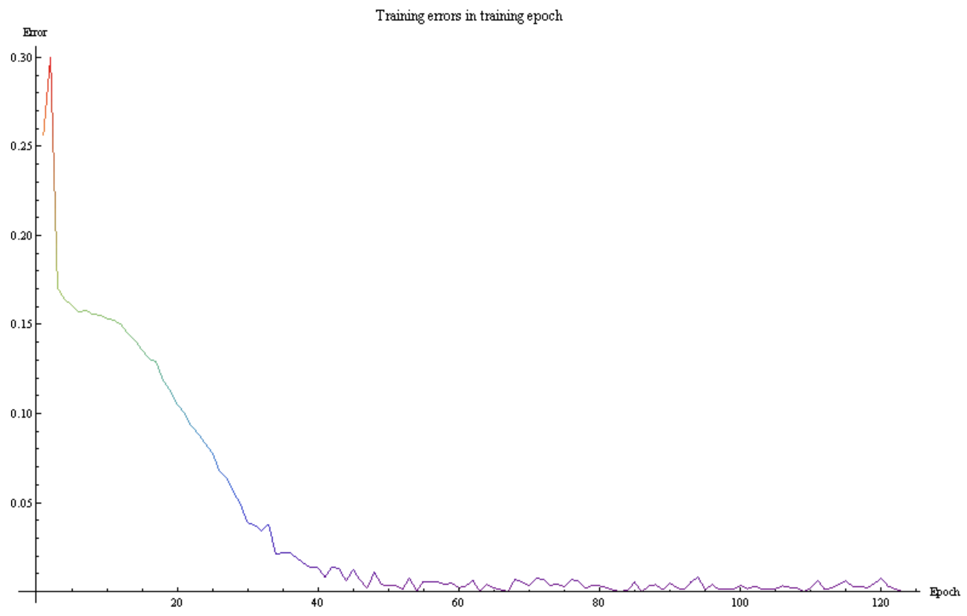


Fig. 9.11 - Training error L500, L100

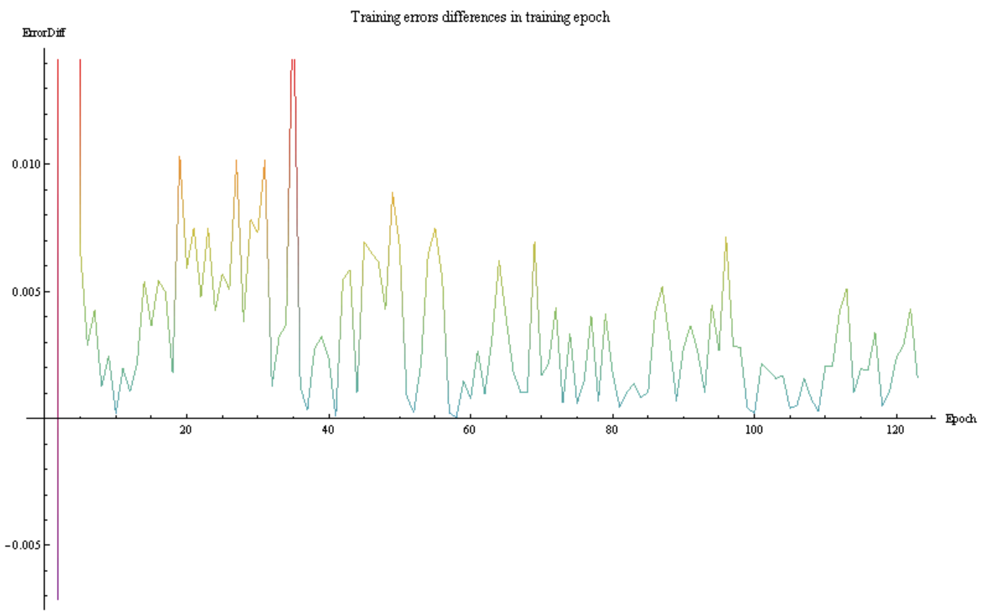


Fig. 9.12 - Training error differences L500, L100

9.2.6 Progress – two hidden layer with 750,300 neurons

Table 9.6 - Training L750, L300

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	750 neurons
2. hidden layer	300 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 ($1 . 10^{-8}$)
Acceptable training error difference	0.00000001 ($1 . 10^{-8}$)
No. training epoch	60
Training duration	00:01:05.21 (H:M:S)
Last training error value	$2.0808 . 10^{-10}$

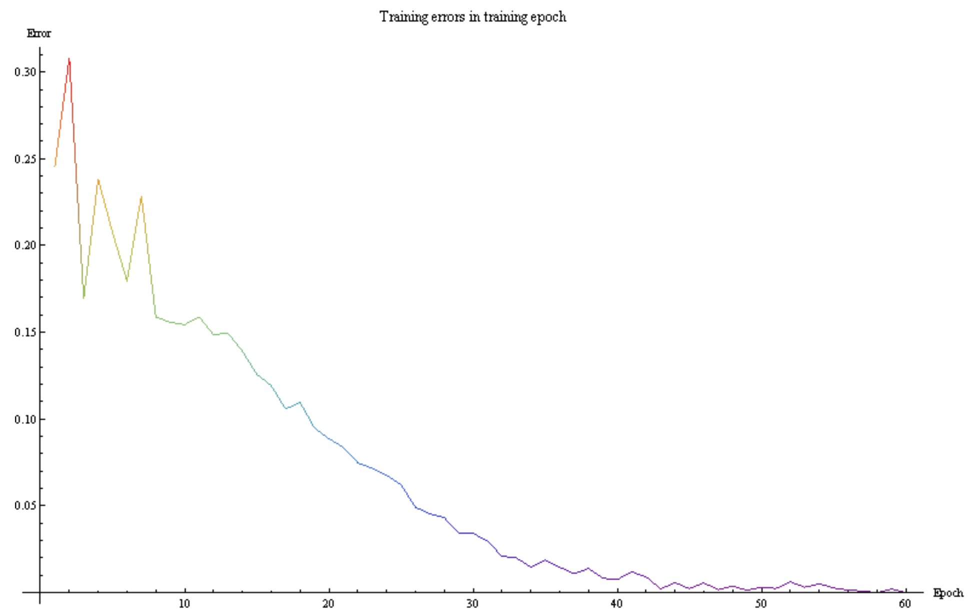


Fig. 9.13 - Training error L750, L300

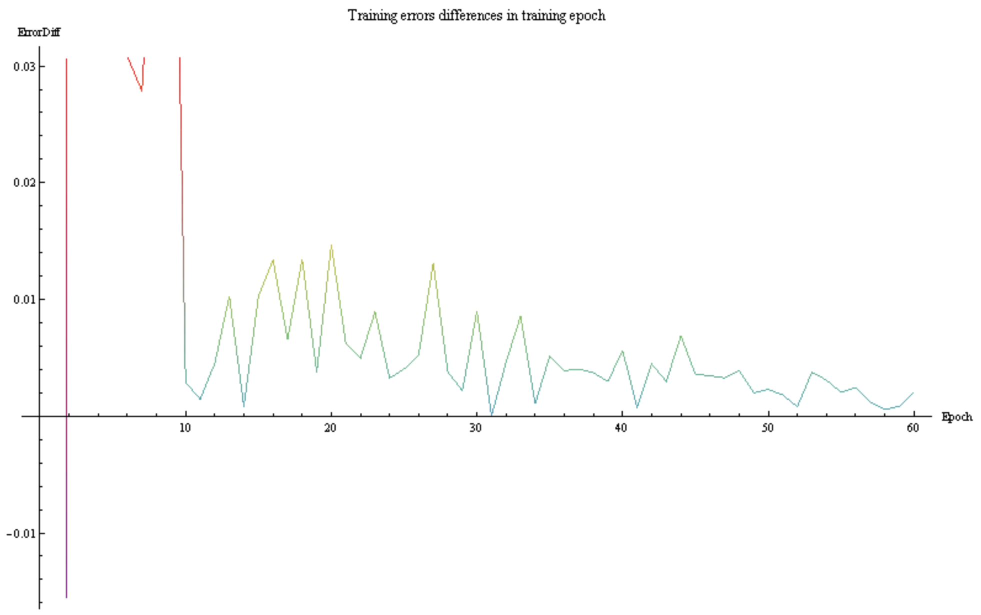


Fig. 9.14 - Training error differences L750, L300

9.2.7 Progress – two hidden layer with 1000,500 neurons

Table 9.7 - Training L1000, L500

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	1000 neurons
2. hidden layer	500 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 (1 . 10 $^{-8}$)
Acceptable training error difference	0.00000001 (1 . 10 $^{-8}$)
No. training epoch	64
Training duration	00:01:55.80 (H:M:S)
Last training error value	7.5042 . 10^{-10}

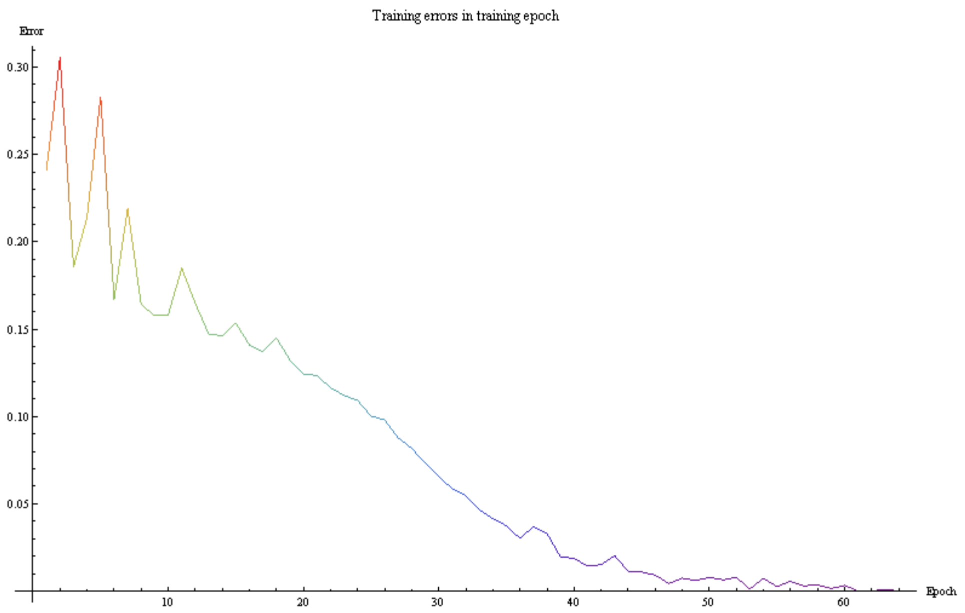


Fig. 9.15 - Training error L1000, L500

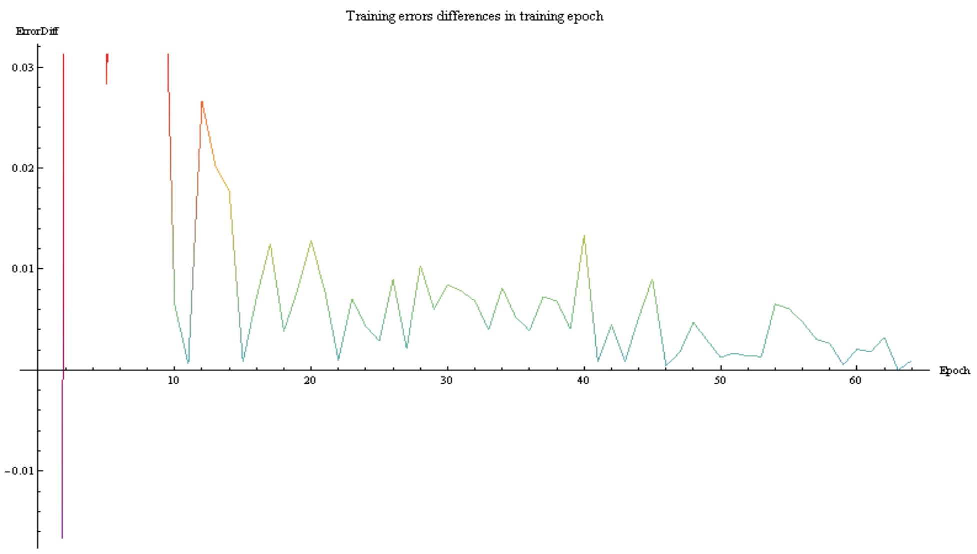


Fig. 9.16 - Training error differences L1000, L500

9.2.8 Progress – three hidden layer with 1000,1500,1000 neurons

Table 9.8 - Training L1000, L1500, L1000

Number of fingerprints	31
Input layer	1400 neurons
1. hidden layer	1000 neurons
2. hidden layer	1500 neurons
3. hidden layer	1000 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 (1 . 10^{-8})
Acceptable training error difference	0.00000001 (1 . 10^{-8})
No. training epoch	1070
Training duration	01:14:46.56 (H:M:S)
Last training error value	5.6115 . 10^{-9}

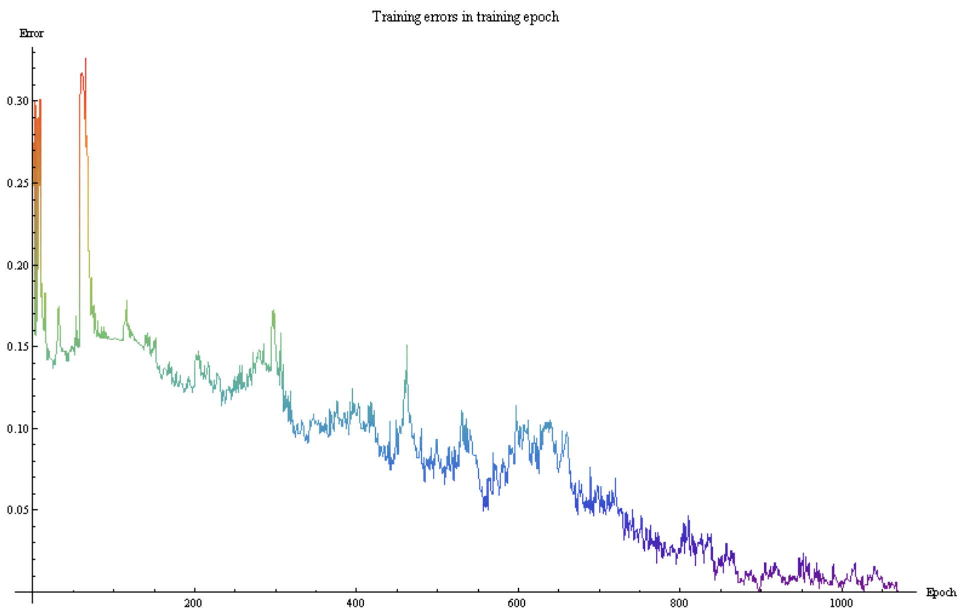


Fig. 9.17 - Training error L1000, L1500, L1000

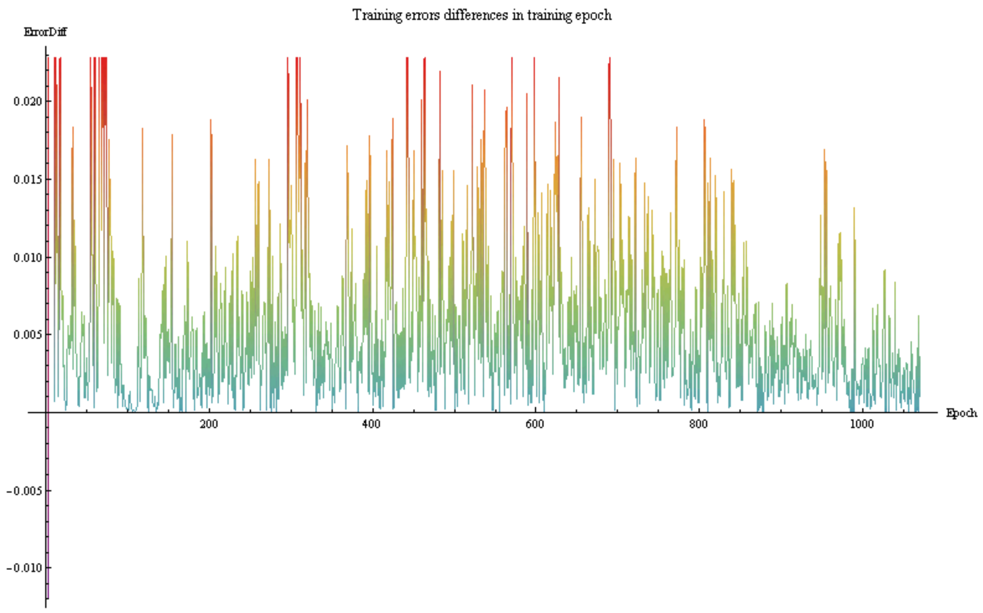


Fig. 9.18 - Training error differences L1000, L1500, L1000

9.2.9 Progress – three hidden layer with 1000,3000,1000 neurons

Table 9.9 - Training L1000, L3000, L1000

Number of fingerprints	**31**
Input layer	1400 neurons
1. hidden layer	1000 neurons
2. hidden layer	3000 neurons
3. hidden layer	1000 neurons
Output layer	31 neurons
Acceptable training error	0.00000001 ($1 . 10^{-8}$)
Acceptable training error difference	0.00000001 ($1 . 10^{-8}$)
No. training epoch	178
Training duration	00:21:34.32 (H:M:S)
Last training error value	$6.6971. 10^{-9}$

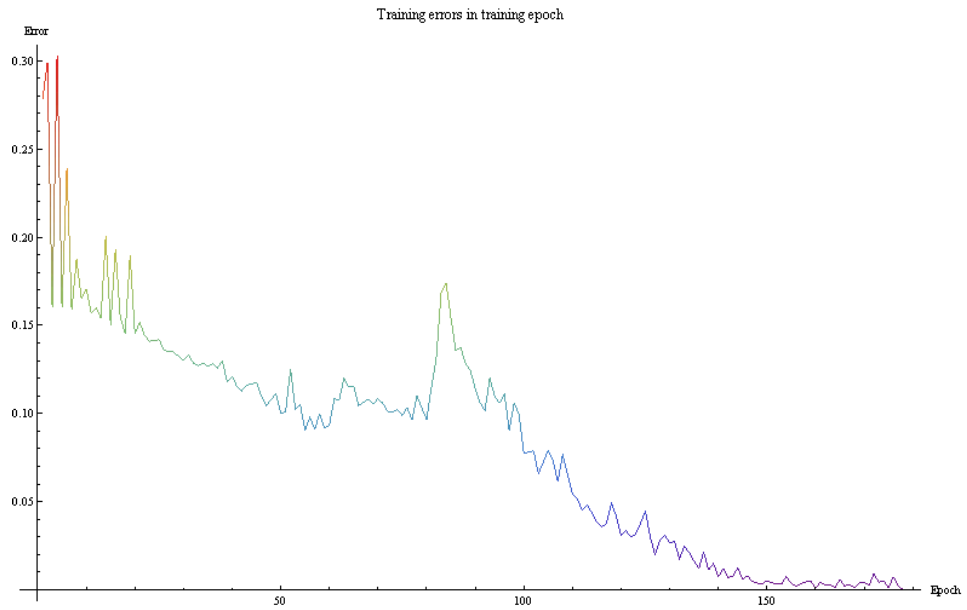


Fig. 9.19 - Training error L1000, L3000, L1000

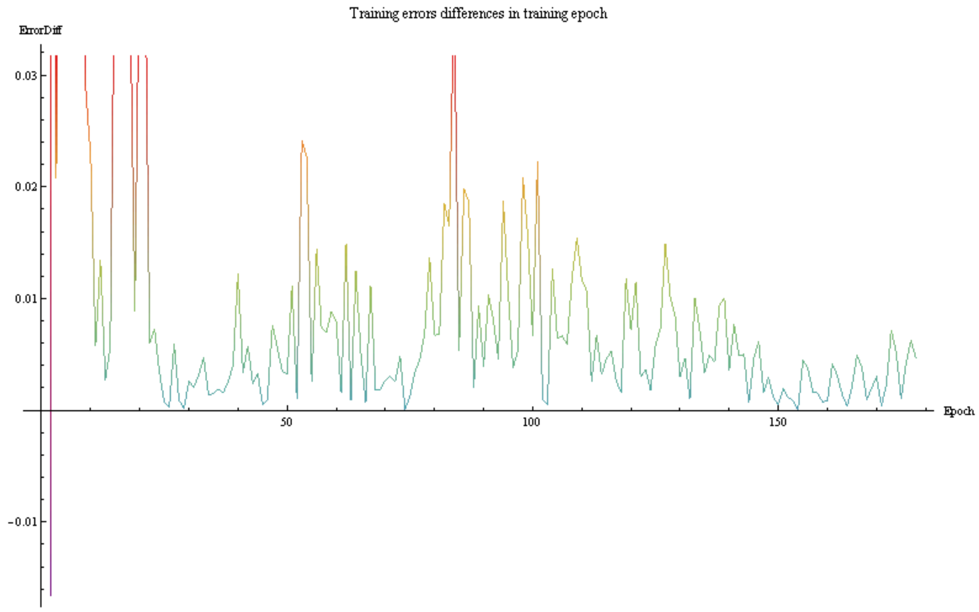


Fig. 9.20 - Training error differences L1000, L3000, L1000

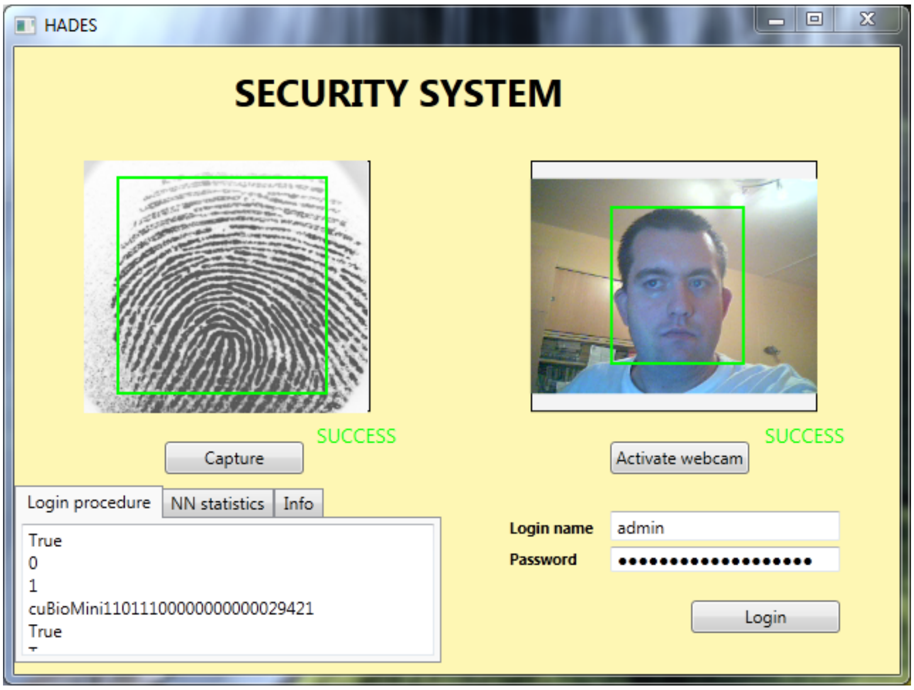
9.3 **Results – usability of NN**

The output from the previously described steps is the trained neural network with high level of self-repairing function. This NN is able to identify the known face samples. This network is stored into SQLite database and used for the verification of people. There are two different usability of the NN; the first is inside using the one neural network for all possible face samples, which system must detect and successfully identify. This process required that system administrator must have the whole database of faces identification pictures. Next there was a network that is trained with these specific samples. This process is applicable in user environment with zero or minimal user migration.

The second usability is flowing from the using one neural network for each user. The network is able to identify only the owner of this NN. This allows using the network as a portable verification token. Each user has specific neural network that will be trained with each view and angle. The user security token contains the user neural network. Verification network is the portable. The results of the identification process are detailed described in APPENDIX B.

10 SECURITY SYSTEM

The designed security system is divided to two independent applications. Each of these applications has own user interface and specific purpose. The first application is only for the verification process. The prototype user interface is shown on Fig. 10.1.

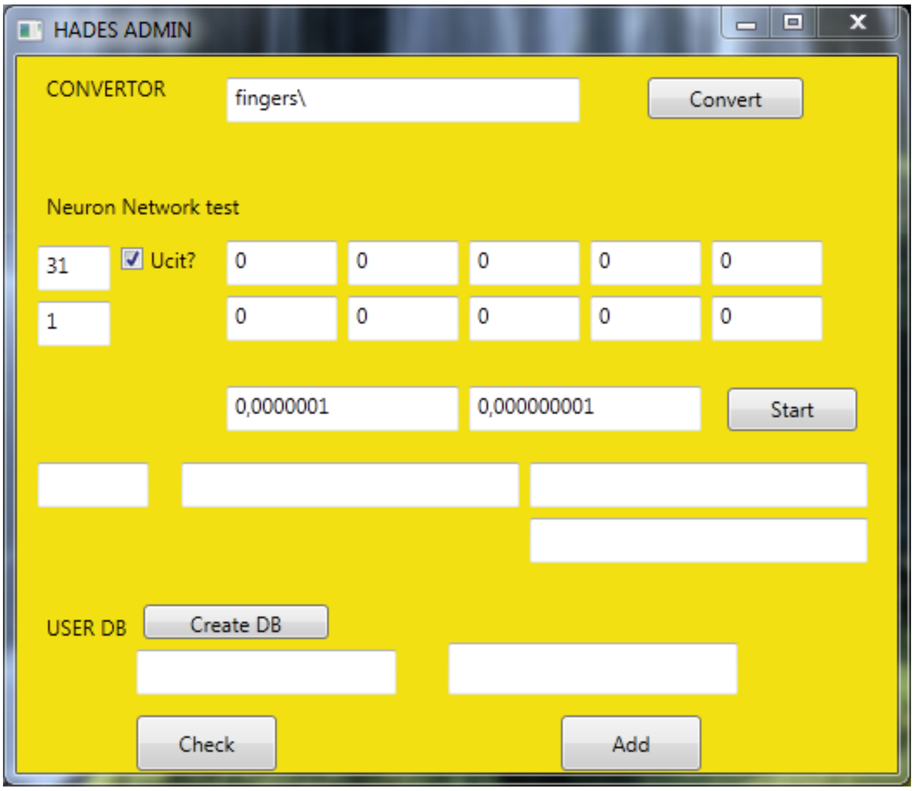


```
True
0
1
cuBioMini1101110000000000029421
True
```

Login name: admin

Fig. 10.1 - Security system UserGUI

The prototype of the admin part of application is shown on Fig. 10.2.



fingers\

31

1

0 0 0 0 0

0 0 0 0 0

0,0000001 0,000000001

Fig. 10.2 - Security system Admin part

10.1 Verification GUI

This part of security application is developer for the verification process. The application used GUI with 2 picture parts. The left one is for fingerprint; this part has highlighted area for the fingerprint. The fingerprint must have the central point inside the highlighted area. The second part is for the user face picture. The application provides live view to the camera; the user might correct the position of face and press Capture button below the picture.

The next part is represented by the username and password. These credentials is using for the generation of the specific userhash. The userhash is combination of hash values from the username and the password. The detail description is in chapter 3.1.3.

10.1.1 User identification process

The user identification process is flowing from the Fig. 10.1. The identification procedure is shown on Fig. 10.3.

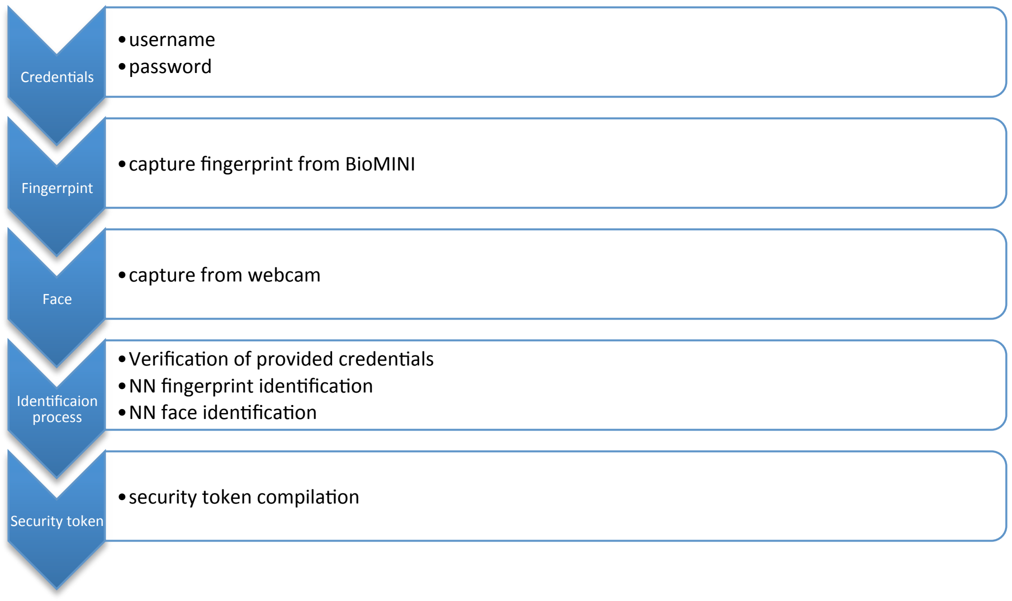

Credentials
•username
•password
Fingerrpint
•capture fingerprint from BioMINI
Face
•capture from webcam
Identificaion process
•Verification of provided credentials
•NN fingerprint identification
•NN face identification
Security token
•security token compilation

Fig. 10.3 - User verification process

The first step is providing the credential information; the valid username and the password. The next part contains the biometric identification procedures including the neural network for the fingerprint and face identification. The user manages the capture process. The user performs click action inside the GUI and captures the fingerprint picture. Next the user clicks onto Capture button below the webcam screen. The verification process continues with the verification of created userhash and two neural networks for identification. The first is for the fingerprint identification and the second one is for the face identification process.

10.1.2 *Class and function documentation*

10.1.2.1 Interface UserGUI

The UserGUI represents the interface and the main part of verification security subsystem. This class contains security token for user verification. The token contains two neural networks for user biometric verification and generated userhash from successful login operation. This token has default value as null. The next structure inside this class is UserCharacteristic structure which assembly the list of controlled user specification.

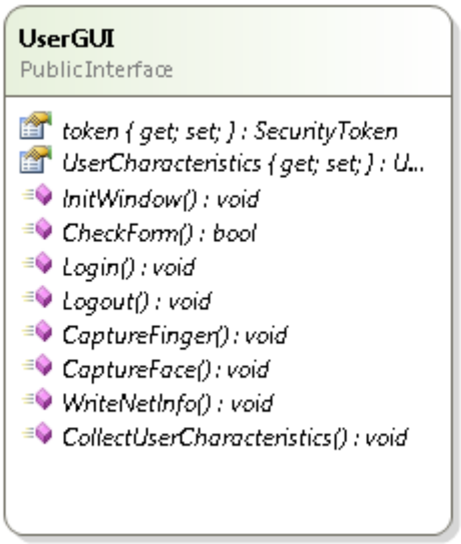


Fig. 10.4 - Interface UserGUI

The UserGUI interface also contains methods for the basic verification operations such as: Login and Logout actions, supported operation for capturing fingerprint and face picture. The checking operation which control the right format of provided credentials is represented by the method CheckForm(). The specific user characteristic is collected by the CollectUserCharacteristics() function. The function also provide the intelligent comparison between stored characteristic and the new one. The statistic information about neural network stored inside the security token shows the function WriteNetInfo().

10.1.2.2 Class SecurityToken

This class represented the security center of designed solution. The class wraps user verification information. The TokenID is the same value that user has inside internal credential database. The ValidHash is the userhash value. The certificate array contains certificates of external systems with trust relationship; system which used this security token as a trust user token. The FingerIdentification is a class with the neural network for the fingerprint identification. The variable FaceIdentification is the neural network for face identification process.

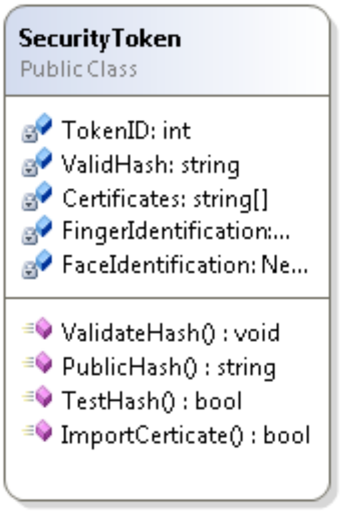


Fig. 10.5 - Class SecurityToken

The class also wraps the method for user verification process. The first is the ValidateHash method for periodical check of userhash value. This is security system disallow any changes of userhash; this value does not be changed without login logout procedure. The PublicHash get the public part of user certificate; these certificates might be imported to external systems for the trust relationship. The ImportCertificate contains function for certificate import. This function is for available two way trust relationship.

10.1.2.3 Class UserCharacteristics

The UserCharacteristcs class wrap the functions for operations with the array of each user characteristics. The CharacteristicArray contains the array of each Characteristic.

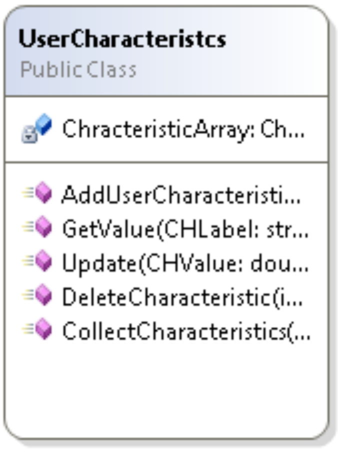


Fig. 10.6 - Class UserCharacteristics

The methods contain the basic operation with the user characteristics such as add user characteristics represented by the method called AddUserCharacterists. Function GetValue is for return of specific user characteristics identify by the characteristic label. The method called as Update contains update function for the user characteristics, this update must be allowed by the supervisor function which determined which change is inside the limit and which might represents the potential security issue. The periodical function CollectCharacteristics running at the background and periodically collecting and updating user characteristics. The periodical check prevents the user-switching without correct user logout and login process.

10.1.2.4 Class Characteristic

This class wrap the function for operation with each of the user characteristics. It represents the atomic part of user characteristics. The class contains variable with the original label of characteristic (variable Label), the specific value of this characteristic (variable Value) and the formal description of characteristic (variable Description). The description contains the text description of characteristic in “human” based style: for example “average no. alphabetic chars click per second”. The method provides functions for the Set and Get value of each variables in this class. The function Update provides updating of class variables.

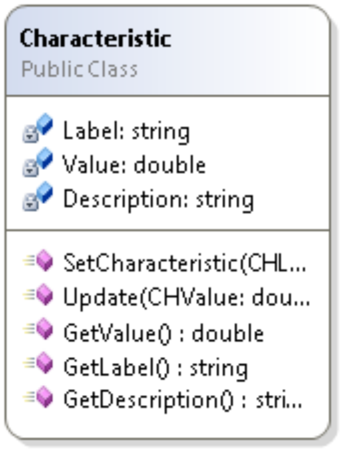


Fig. 10.7 - Class Characteristic

10.1.2.5 Class DatabaseClass

The class wraps the SQLite database functionalities. The local variable represents the connection string to existing SQLite database. This solution was chosen for his high availability. The database type switching will be realized by creation of new Class with the operation with other database type: for example MySQL, MSSQL or other.

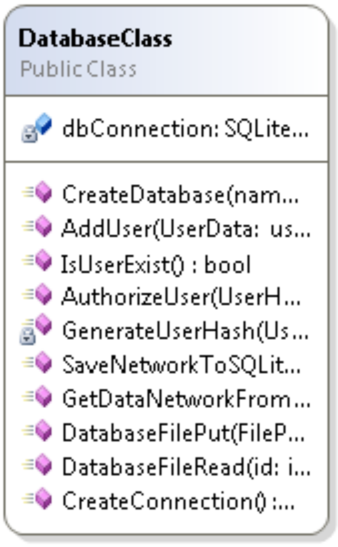


Fig. 10.8 - Class DatabaseClass

The methods inside this class wraps provide operation with the SQLite database. There are methods for generate connection string (CreateConnection), for creating database (CreateDatabase), user operations (AddUser – add new user to database), user exist check (IsUserExist), functions for userhash generation (GenerateUserHash). The method SaveNetworkToSQLite provides functionality for saving the neural network to the SQLite database.

The method GetDataNetworkFromDB provides operations for the loading neural network from SQLite database. Methods DatabaseFilePut and

DatabaseFileRead collect function for the export or import neural network to or from binary file.

10.1.2.6 Class ImageOperations

This class simplifies the build in image operation. The class provide the functions for the image colour conversion to B/W colour model. The method convertImages provide batch conversion operation on array of files.

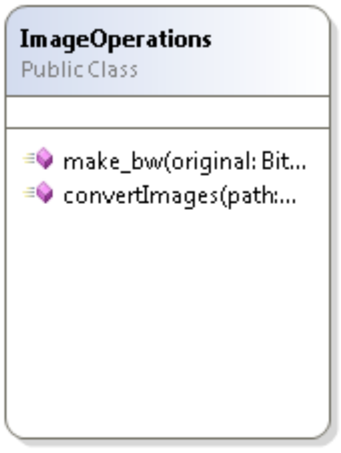


Fig. 10.9 - Class ImageOperations

10.1.2.7 Class NeuralNet

This class simplify the neural network creation and basic operation with neural network. It provides some simple interface for the Encog neural network framework.

The class constructor builds the structure of the neural network. The LoadNetwork loads serialized neural network to the Encog framework. The function SaveNetwork serialize Encog network and export it to the database or file.

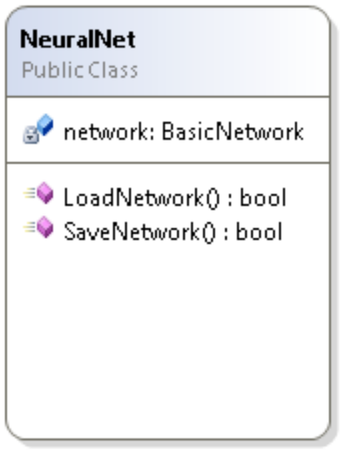
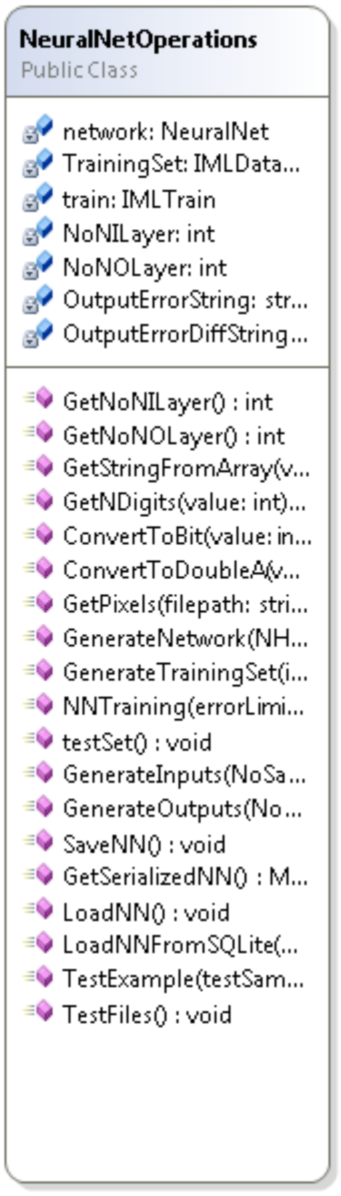


Fig. 10.10 - Class NeuralNet

10.1.2.8 NeuralNetOperations

This class wraps he operation with the neural network. It represented the bridge between the security system and the Encog neural network framework. The variable netwotk represents the neural network. The TrainingSet is the training set for the neural network training process. The Train is the progress and status from training operation. The NoNILayer represents the number of neurons in input layer (based on the size of input vector). The variable NoNOLayer contain the number of neurons in output layer; 31 neurons in this application. The OutputErrorString is the log string with training error progress. The OutputErrorDiffString represents the log string with error difference during training.



```
NeuralNetOperations
Public Class

network: NeuralNet
TrainingSet: IMLData...
train: IMLTrain
NoNILayer: int
NoNOLayer: int
OutputErrorString: str...
OutputErrorDiffString...

GetNoNILayer() : int
GetNoNOLayer() : int
GetStringFromArray(v...
GetNDigits(value: int)...
ConvertToBit(value: in...
ConvertToDoubleA(v...
GetPixels(filepath: stri...
GenerateNetwork(NH...
GenerateTrainingSet(i...
NNTraining(errorLimi...
testSet() : void
GenerateInputs(NoSa...
GenerateOutputs(No...
SaveNN() : void
GetSerializedNN() : M...
LoadNN() : void
LoadNNFromSQLite(...
TestExample(testSam...
TestFiles() : void
```

Fig. 10.11 - Class NeuralNetOperations

The methods included inside this class provide the functionalities for the operations with the neural network model. There are some methods for returning the private value from the call container: GetNoNILayer, GetNoNOLayer, The next is the support function for the operation with the string values and double arrays: GetStringFromArray and GetNDigits, ConvertToBit, ConvertToDoubleA, GetPixels. The other function is implemented for the operations with neural network: GenerateNetwork, GenerateTrainingSet, NNTraining, TestSet, GenerateInputs, GenerateOutputs, LoadNN, LoadNNFromSQLite, SaveNN, GetSerializedNN, TestExample and TestFile.

10.2 Admin GUI

10.2.1 Creating security token

The next picture describes the process of security token generation. The Admin GUI realizes this process.

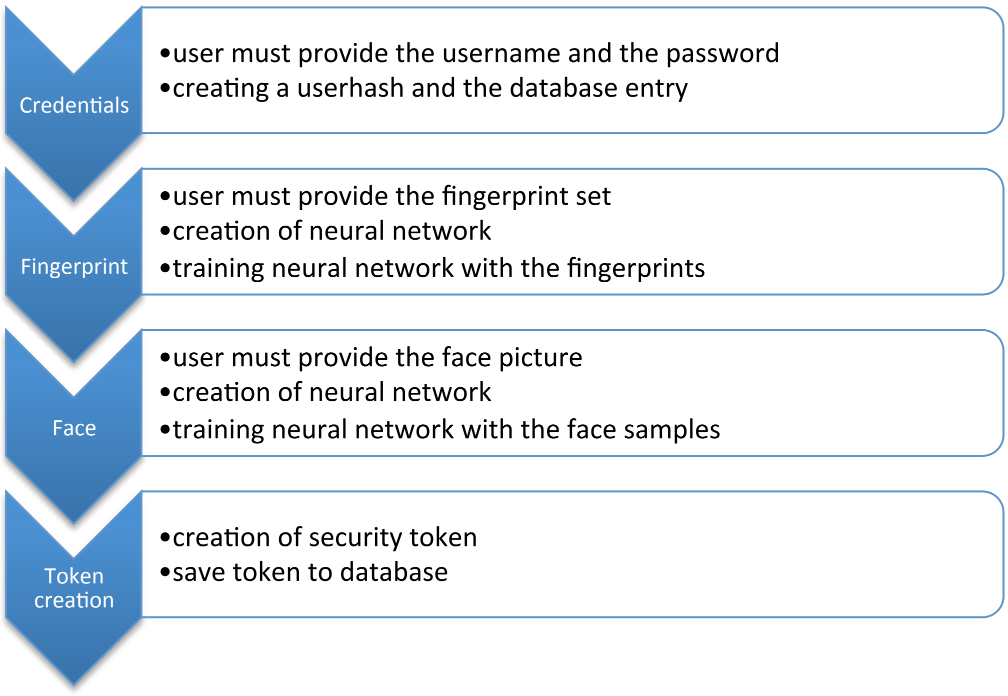

Credentials
•user must provide the username and the password
•creating a userhash and the database entry
Fingerprint
•user must provide the fingerprint set
•creation of neural network
•training neural network with the fingerprints
Face
•user must provide the face picture
•creation of neural network
•training neural network with the face samples
Token creation
•creation of security token
•save token to database

Fig. 10.12 - Creating of security token

The first step of token creation operation is collecting standard user credentials such as username and password. Then there is created a userhash value. The userhash represents the one verification parameter of security process. The next step is providing the fingerprint sample; it is possible to collect more than one sample as the sample set. The created neural network depends on provided samples; the effect of variant number of hidden layer and number of neurons inside these hidden layer is shown in chapter 8.2. The number of hidden layers affects the quality and the time of training procedure. The output from the training process is the trained neural network that is able to identify learned samples.

The next procedure needs the face sample pictures. These pictures are used for the training process of face identification process. The affect of various combinations of hidden layer and number of neurons is shown in chapter 9.2.

The finish operation creates the original token. This token might be portable or standalone.

10.2.2 Class and function documentation

10.2.2.1 Interface AdminGUI

The AdminGUI interface is built for the creating a security token for the users. This console is using by the administrator. The administrator might add user to the internal database and create the security token for him.

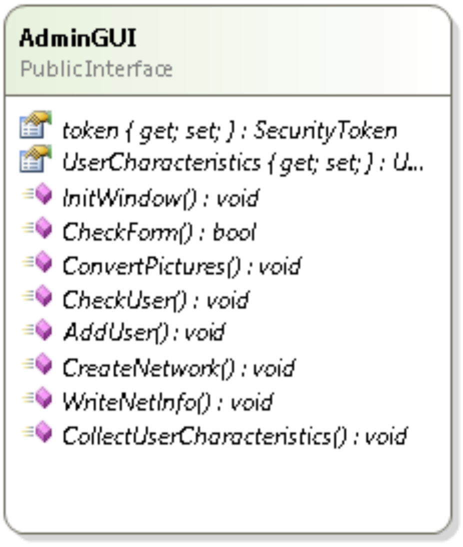


Fig. 10.13 - Class AdminGUI

The interface includes the variable for the security token (token) and user characteristics (UserCharacteristic). The methods are following: InitWindow for creating a GUI window of application. The CheckForm is the standard function for control the form data before executing the main procedure. The ConvertPictures is the method for converting colour pictures to B/W pictures. The CheckUser is the method for verification if the user exists in the database. This is for the preventing duplicity entries in database. The procedure AddUser provides operation for adding new user to database. The editing function is implemented ad AddUser with special parameter. The CreateNetwork function is for creating the neural network; this is abstract function, which might generate neural network for finger and face. The WriteNetInfo returns the information from training process to application. The CollectUserCharacteristic specify the file of user characteristic which will be start after login procedure.

10.2.2.2 Basic Class

The basic/core classes are the same that in UserGUI interface. The detailed list of these method and classes is described between chapter 10.1.2.2 and 10.1.2.8.

10.2.2.3 Class BCHCodes

This class is implemented for the operation with BCHCodes. The class represented the simplification of BCHCode usage. The static variable represents the generated binary string for the BCH code securing. The method implements the basic operation with the BCH code. The GetBCHCode returns the 31/15 BCHCode from the user-defined number. The method CheckBCHCode provide test function for the checking the BCHCode, this is the method for detecting errors. The DecodeBCHCode returns the corrected BCH code from damaged samples. The method used the private method TryRepair for the test or reconstruct able; if the repair is impossible, the TryRepair function returns error value.

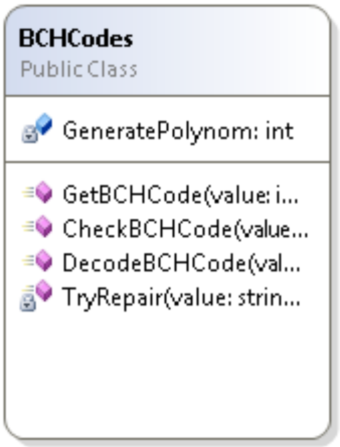


Fig. 10.14 - Class BCHCodes

10.2.2.4 Class FileOperations

This class implement the basic operations with files. The class represented the bridge between the system file function and security system. This class wraps the method for the get file list from specific folder (method GetFiles). The method

WriteOutPutData is for writing test progress to the backup txt file. The backup file example is in APPENDIX C. The CreateTXTOutput is the function for creating txt file and prepare it for writing. The CloseTXTOutput finalize the backup file and close the connection stream.

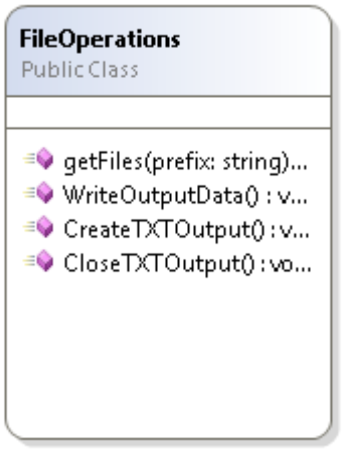


Fig. 10.15 - Class FileOperations

11 CONCLUSION AND DISCUSSIONS

The token represents the complete security solution. This token is the multipurpose routine for user verification, which combines several subsystem based on artificial neural network features. The previously described part of the final solution of the thesis aim represents the usage of the artificial neural network as a result of the modern security verification system for user authentication. The artificial neural network works as the basic feature of all described subsystems, which make the complete solution; the security token. The designed solution combines two totally different views to user verification systems. The classical verification system uses some internal database with users and verification information. The next level is roaming profile such as the European academic wireless network Eduroam. The designed solution uses both methods. It is usable as a standalone system for the user verification process. But it is usable as the autonomy security verification system, which uses the asymmetric key exchange for creating trust relationships with other systems. The token is hybrid solution, which combines the standalone and the roaming system of verification.

The thesis presents the modern usability of artificial neural networks in the security focus. The usability is really wide, because the network represents the similar system as a human brain; in very simply approximation. The network might react to problems, which are not the binary clear. It might repair some damaged data, it might learns (especially dynamically learning) the human behaviour and specific characteristics. The security system described in this work used neural network for fingerprint and face identification; the process of identification is significantly easy and realized by the common technologies, but the neural network has better results with the non-ideal samples, which are affected by the outside environment. The samples might be damaged. The network might applicable his self-repair function and successfully identify/classify the sample from database. The next advantage is flowing from the structure and functionalities of neural network. There is not necessary to save the user biometric data such as fingerprint and face picture inside database. The neural network does not save any sensitive personal data of verified user.

The previously described features of the artificial neural network represent the future usability of these networks. Now, there is one huge problem with usage of neural networks. The learning process is limited by speed of the present computer system. The intelligent behaviour of the security systems represents the future of identification systems. The pure standard algorithm is dummy computer program, which does not support any adaptability. The samples are verified bit by bit. The neural network might bring the artificial intelligence inside this process. The neural network is commonly using in IDS security system [18]. The user verification is still waiting for the intelligent application.

The aim goals started at the beginning of this thesis are fulfilled in previous chapters in the following way.

- **To prove that there is the place for the assembly of the neural networks to the user verification processes**
    - The chapter 1 described the state of art; there is a very large place for application of neural networks. The practical application is detailed described in chapter 8 and chapter 9.
- **To prove that the neural network might successfully verified the user or might repaired the sample used for the user verification**
    - This problem is described in chapter 3.1.2, chapter 8 and chapter 9.
- **To prove that the speed of the solution based on the neural network is inside the practical limited (for example: the time for the learning, identification, etc.)**
    - This problem is showed in chapter 8 and chapter 9, the speed of identification and self-repair procedure is addicted to the

structure and size of the neural network. The basic test consume a few second for the training process. The identification process is fast.

- **To define the part of standard user verification procedure that might be replaced or proved by the neural network.**
  - The definition is showed in chapter 8, chapter 9 and chapter 10. The chapters represent the practical usage of this neural network identification process.
- **To define the limited of usage for the neural network inside the computer security application.**
  - The potentially limits of the neural network identification is flowing from the size of the neural network and from the time which identification process consumes. This limited id showed in chapters in chapter 8 and chapter 9.
- **To develop a system prototype for testing neural network application inside the computer security processes.**
  - The developed prototype is described in chapter 10.

These points seem to be fulfilled too.

12 LIST OF AUTHOR'S PUBLICATION ACTIVITIES

2008

1. MALANÍK, D., DULÍK, T., DRBÁLEK, Z., ČERVENKA, M.: System for capturing, streaming and sharing video files, WSEAS Press (IT), Proceedings of the 8th WSEAS International Conference on DISTANCE LEARNING and WEB ENGINEERING, Venice, 2008, 86-91, ISBN-ISSN 978-960-474-005-5

2009

1. MALANÍK, D.: Bezpečnost přihlašování v moderní síti Internet, Informační a datová bezpečnost ve vazbě na strategické rozhodování ve znalostní společnosti
   Zlín, 24. – 25. 3. 2009, ISBN-ISSN 80-238-6782-7
2. MALANÍK, D., KOUŘIL, L., HECZKO, M. Disease- Simulation Environment.,České národní finále Imagine Cup 2009, Sborník soutěžních prací. Egypt 09 imagine cup Microsoft. 2009
3. MALANÍK, D.: Possibilities of usage single login technologies to different web services, Information and Data Security, Crisis Managementand Strategic Decision-making in Knowledge Society Zlin, Czech Republic, 24. – 25. 3. 2009, ISBN-ISSN 80-238-6782-7

2010

1. MALANÍK, D. Nature Behavior in Stochastic Extreme Finding Methods (2010). 1205-1207, Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, ISBN 978-3-901509-73-5, ISSN 1726-9679, pp 0603, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2010
2. MALANÍK D., Possible issues in extreme finding algorithms, XXXV. Seminar ASR '2010 " Instruments and Control" Ostrava 2010, ISBN 978-80-248-2198-7
3. MALANÍK D., JAŠEK R., POSSIBILITIES OF USAGE NEURAL NETWORKS IN CRYPTOGRAPHY, Internet, bezpečnost a konkurenceschopnost organizací, Zlín 2010 ISBN 978-83-61645-16-0

4. MALANÍK D., JAŠEK R., PHYSICAL SECURITY IN IT SYSTEMS, Internet, bezpečnost a konkurenceschopnost organizací, Zlín 2010 ISBN 978-83-61645-16-0

5. JASEK R., MALANIK D.: Speed Differences between Mathematica and C# Languages in Identiying Extreme Values in Applications, Proceedings of the Workshop: Methods and Applications of Artificial intelligence, 23-24 September 2010, The College of Informatics and Management, Bielsko-Biava, pp. 27-37, ISBN 978-83-62466-02-3.

6. MALANÍK D., Význam fyzického zabezpečení IT systémů, Security Revue - ISSN 1336-9717 September 28th, 2010

2011

1. MALANÍK D.,THE PENETRATION TESTING OF WIFI NETWORKS, Internet, bezpečnost a konkurenceschopnost organizací, ISBN: 978-80-7454-012-7, pp 263 - 170 , Published by Tomas Bata University in Zlín, 2011, Faculty of Applied Informatics.

2. MALANÍK D.,THE SECURITY ISSUES DURING DEVELOPING WIFI NETWORKS, Internet, bezpečnost a konkurenceschopnost organizací, ISBN: 978-80-7454-012-7, pp 270 - 275 , Published by Tomas Bata University in Zlín, 2011, Faculty of Applied Informatics.

3. BOUDNÁ H., MALANÍK D., KVANTOVÁ KRYPTOGRAFIE, Internet, bezpečnost a konkurenceschopnost organizací, ISBN: 978-80-7454-012-7, pp 57 - 62 , Published by Tomas Bata University in Zlín, 2011, Faculty of Applied Informatics.

4. MALANÍK D.,SYSTEMS FOR THE IDENTIFICATION OF ENTRY AND BIOMETRIC SYSTEMS, Internet, bezpečnost a konkurenceschopnost organizací, ISBN: 978-80-7454-012-7, pp 57 - 62 , Published by Tomas Bata University in Zlín, 2011, Faculty of Applied Informatics.

5. MALANIK D., JASEK R., NEURAL NETWORK FACE IDENTIFICATION, The 11th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC '11), ISBN 978-1-61804-028-2, pp 129 - 134, Published by WSEAS Press 2011

13 REFERENCES

1. BÍLA, J. *Umělá inteligence a neuronové sítě v aplikacích*. Vyd. 2., přepracované. Praha : ČVUT, 1998. ISBN: 9788001017692.
2. BITTO, O. *Šifrování a biometrika, aneb, Tajemné bity a dotyky*. Vyd. 1. Kralice na Hané : Computer Media, 2005. ISBN: 9788086686486.
3. BOXER, L. *Algorithms sequential & parallel : a unified approach*. 2nd ed. Hingham Mass. : Charles River Media, 2005. ISBN: 9781584504122.
4. COLE, E. *Network security bible*. 2nd ed. Indianapolis IN ;Chichester : Wiley ;;John Wiley, 2009. ISBN: 9780470502495.
5. DELFS, H. *Introduction to cryptography principles and applications*. Berlin ;;New York : : Springer,, 2007. ISBN: 9783540492436.
6. DROZDOWSKI, M. *Scheduling for parallel processing*. Dordrecht : : Springer,, 2009. ISBN: 9781848823105.
7. GRAHAM, B. *Security analysis : principles and technique*. 6th ed. New York : McGraw-Hill, 2009. ISBN: 9780071592536.
8. HEATON, J. *Introduction to neural networks for C#*. 2nd ed. St. Louis : Heaton Research Inc., 2008. ISBN: 9781604390094.
9. HUANG, S. *Network security*. New York ;;London : : Springer,, 2010. ISBN: 9780387738208.
10. KATZ, J. *Introduction to modern cryptography*. Boca Raton : Chapman & Hall/CRC, 2008. ISBN: 9781584885511.
11. LEIWO, J. et al. A Security Design for a Wide-Area Distributed System. In SONG, J. (ed.). *Information Security and Cryptology - ICISC'99*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2000 [cit. 2011-10-28], p. 236-256. URL <http://www.springerlink.com/index/10.1007/10719994_19>. ISBN: 978-3-540-67380-4, 978-3-540-45568-4.
12. LOCKHART, A. *Network security hacks*. 2nd ed. Sebastopol CA : O'Reilly, 2007. ISBN: 9780596527631.
13. MA, J. *Security Access in Wireless Local Area Networks From Architecture and Protocols to Realization*. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2009. ISBN: 9783642009419.
14. MALTONI, D. *Biometric authentication : ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15th, 2004 : proceedings*. Berlin : : Springer,, 2004. ISBN: 9783540224990.
15. MCCLURE, S. *Hacking bez záhad*. 1. vyd. Praha : Grada, 2007. ISBN: 9788024715025.
16. MICCIANCIO, D.; PANJWANI, S. Adaptive Security of Symbolic Encryption. In KILIAN, J. (ed.). *Theory of Cryptography*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005 [cit. 2011-10-28], p.

169-187. URL <http://www.springerlink.com/index/10.1007/978-3-540-30576-7_10>. ISBN: 978-3-540-24573-5, 978-3-540-30576-7.

17. NASH, T. *C# 2010 : rychlý průvodce novinkami a nejlepšími postupy*. Vyd. 1. Brno : Computer Press, 2010. ISBN: 9788025130346.

18. PALOMO, E. J. et al. A Competitive Neural Network for Intrusion Detection Systems. In LE THI, H. A.; BOUVRY, P.; PHAM DINH, T. (eds.). *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Berlin, Heidelberg : Springer Berlin Heidelberg, [cit. 2011-10-28], p. 530-537. URL <http://www.springerlink.com/index/10.1007/978-3-540-87477-5_56>. ISBN: 978-3-540-87476-8, 978-3-540-87477-5.

19. SALOMON, D. *Foundations of computer security*. London, UK : : Springer,, 2006. ISBN: 9781846281938.

20. SOLANAS, A. *Advances in artificial intelligence for privacy protection and security*. Singapore ;;Hackensack NJ : World Scientific, 2010. ISBN: 9789812790323.

21. ŠNOREK, M. *Neuronové sítě a neuropočítače*. Vyd. 1. Praha : Vydavatelství ČVUT, 2002. ISBN: 9788001025499.

22. TIPTON, H. *Information security management handbook*. Boca Raton, FL : : Auerbach Publications,, 2007. ISBN: 9781420013580.

23. VALAGUSSA, F.; VALAGUSSA, L. [Community cardiology and health promotion through schools]. *Italian Heart Journal: Official Journal of the Italian Federation of Cardiology*. 2004-11, vol. 5 Suppl 8, p. 42S-44S; discussion 52S-53S, 116S-121S. PMID: 15615359.

24. XIAO, Y. *Handbook of security and networks*. Hackensack, NJ : : World Scientific,, 2011. ISBN: 9789814273046.

25. ZELINKA, I. *Aplikovaná informatika, aneb, Úvod do fraktální geometrie, buněčných automatů--*. Vyd. 2. Ve Zlíně : Univerzita Tomáše Bati Fakulta technologická, 2005. ISBN: 9788073182755.

26. ZELINKA, I. *Aplikace umělé inteligence*. Vyd. 1. Zlín : Univerzita Tomáše Bati ve Zlíně, 2010. ISBN: 9788073188986.

27. ZELINKA, I. *Umělá inteligence, aneb, Úvod do neuronových sítí, evolučních algoritmů--*. Vyd. 2. Ve Zlíně : Univerzita Tomáše Bati, 2005. ISBN: 9788073182779.

28. ZELINKA, I. *Umělá inteligence - hrozba nebo naděje?* 1. vyd. Praha : BEN - technická literatura, 2003. ISBN: 9788073000684.

29. ZHU, X.; YU, Y.; WANG, H. Research of Immune Neural Network Model Based on Extenics. In LI, K. et al. (eds.). *Bio-Inspired Computational Intelligence and Applications*. Berlin, Heidelberg : Springer Berlin Heidelberg, [cit. 2011-10-28], p. 18-27. ISBN: 978-3-540-74768-0, 978-3-540-74769-7.

APPENDIX

APPENDIX A: BSA testing results

APPENDIX B: : Neural network– finger/face identification

APPENDIX C: Backup file example

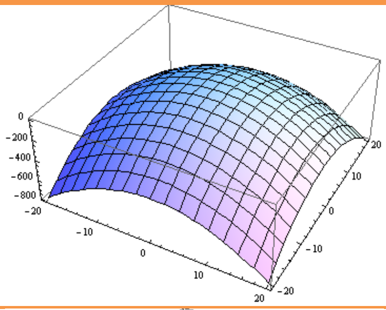
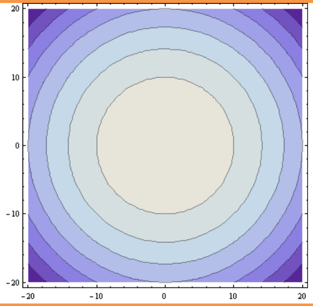
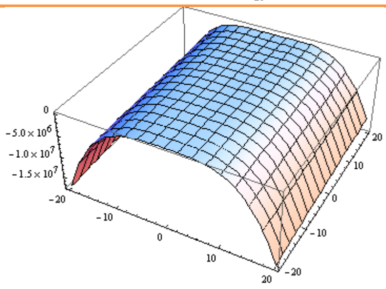
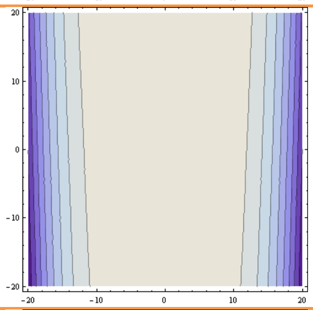
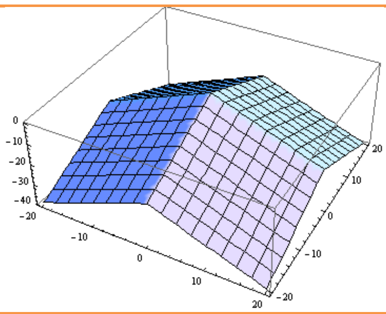
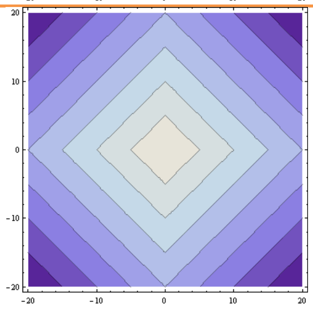
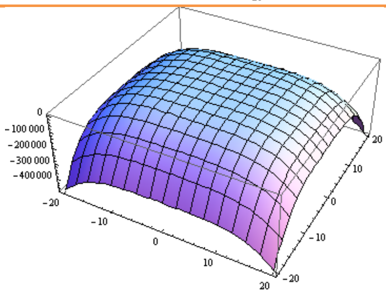
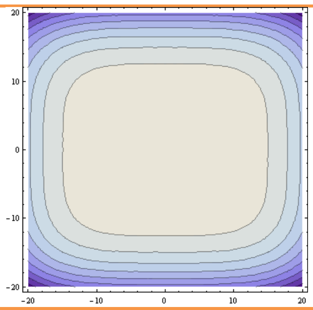
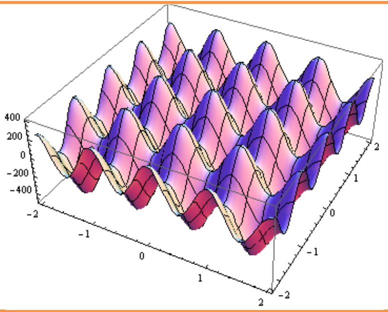
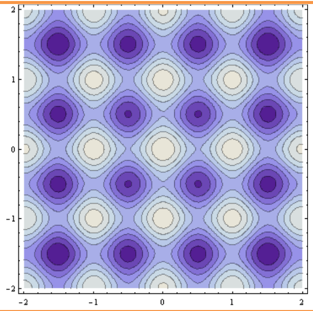
APPENDIX D: Curriculum vitae

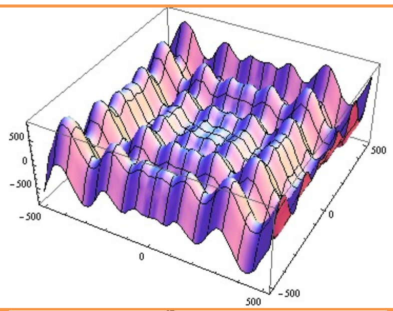
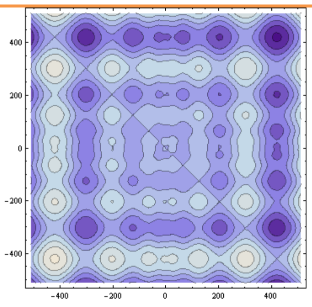
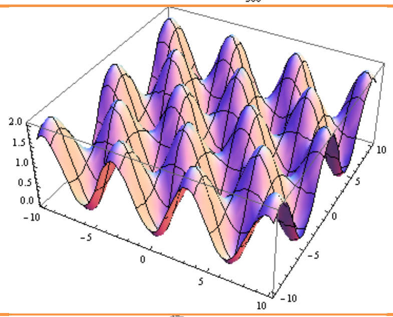
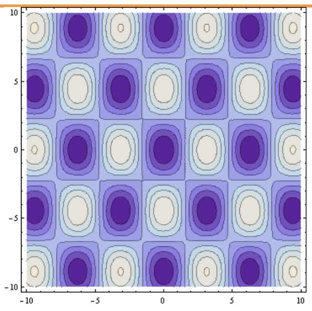
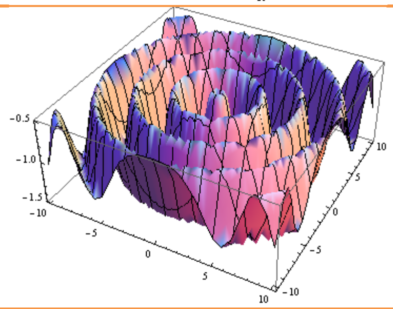
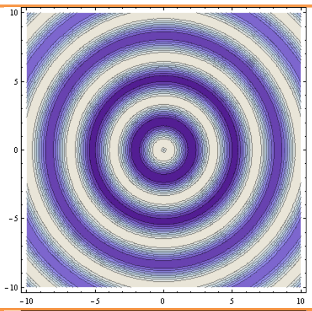
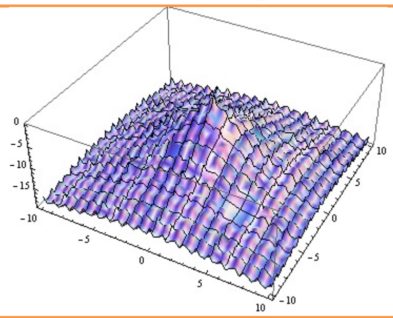
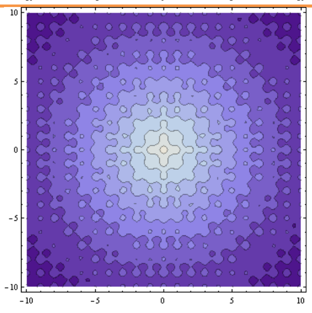
APPENDIX A: BSA testing results

Function	Equation	Extremes
DeJong1st	$-\sum_{i=1}^{\text{dinemsions}} x[\![i]\!]^2$	1 GE[1]
DeJong2nd	$-\sum_{i=1}^{\text{Dimensions}-1} ((1-x[\![i]\!])^2+100(x[\![i]\!]^2-x[\![i+1]\!])^2)$	1 GE
DeJong3rd	$-\sum_{i=1}^{\text{dinemsions}} \lvert x[\![i]\!]\rvert$	1 GE
DeJong4th	$-\sum_{i=1}^{\text{Dimensions}} ix[\![i]\!]^4$	1GE
Rastrigin	$-20\sum_{i=1}^{\text{Dimensions}} (x[\![i]\!]^2-10\cos(2\pi x[\![i]\!]))$	? GE, n LE[2]
Schwefel	$\sum_{i=1}^{\text{dinemsions}} -x[\![i]\!]\sin\left(\sqrt{\lvert x[\![i]\!]\rvert}\right)$	1 GE, n LE
Griewangk	$1-\prod_{i=1}^{\text{Dimensions}} \cos\left(\frac{x[\![i]\!]}{\sqrt{i}}\right)+\sum_{i=1}^{\text{Dimensions}} \frac{x[\![i]\!]^2}{4000}$	? GE, n LE
SineWave	$-\sum_{i=1}^{\text{Dimensions}-1}\left(\frac{\sin^2\left(0.5-\sqrt{x[\![i]\!]^2+x[\![i+1]\!]^2}\right)}{\left(0.001\left(x[\![i]\!]^2+x[\![i+1]\!]^2\right)+1\right)^2}+0.5\right)$	? GE, n LE
Ackley	$-\sum_{i=1}^{\text{Dimensions}-1}\left(-20e^{-0.141421\sqrt{x[\![i]\!]^2+x[\![i+1]\!]^2}}-e^{0.5(\cos(2\pi x[\![i]\!])+\cos(2\pi x[\![i+1]\!]))}+20+e\right)$	1 GE, n LE

[1] Global extreme

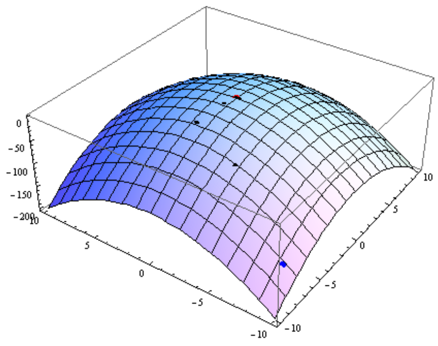
[2] Local extreme

Function		Visualisation
DeJong1st		
DeJong2nd		
DeJong3rd		
DeJong4th		
Rastrigin		

Schwefel		
Griewangk		
SineWave		
Ackley		

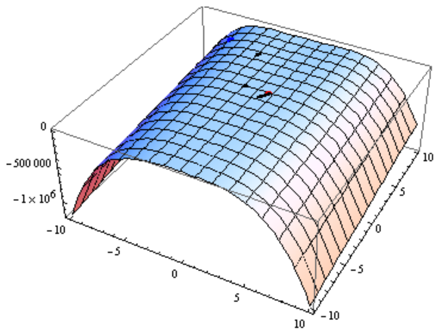
DeJong1st

Start point(the blue point)	[-7.70071 ; -9.02031]
Start value(fitness)	-140.667
Number of iteration	131
Number of fitness calculation	4 250
Found extreme position	[6.83359×10^{-13} ; 4.0505×10^{-13}]
Found extreme value(fitness)	-6.31044×10^{-25}
Known extreme position	[0 ; 0]
Known extreme value(fitness)	0



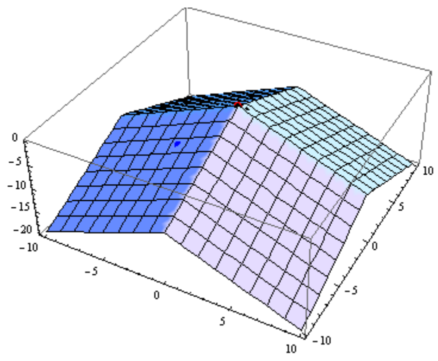
DeJong2nd

Start point(the blue point)	[-5.83628 ; 7.36614]
Start value(fitness)	-7 1314.8
Number of iteration	7 281
Number of fitness calculation	218 750
Found extreme position	[1 ; 1]
Found extreme value(fitness)	-1.96979×10^{-15}
Known extreme position	[1 ; 1]
Known extreme value(fitness)	0



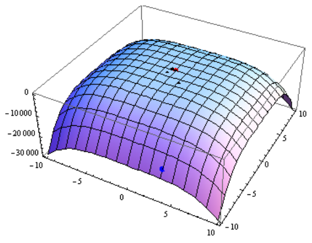
DeJong3rd

Start point(the blue point)	[-3.08531 ; -3.06706]
Start value(fitness)	-6.15237
Number of iteration	195
Number of fitness calculation	6 149
Found extreme position	[6.1169×10^{-14} ; 7.03989×10^{-13}]
Found extreme value(fitness)	-7.65158×10^{-13}
Known extreme position	[-1.97503×10^{-11} ; -8.97822×10^{-11}]
Known extreme value(fitness)	-1.09532×10^{-10}



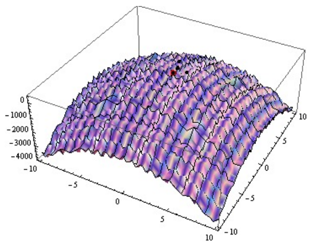
DeJong4th

Start point(the blue point)	[3.72643 ; -9.5665]
Start value(fitness)	-16 943.9
Number of iteration	102
Number of fitness calculation	3 366
Found extreme position	[1.60697×10^{-6} ; -1.74361×10^{-6}]
Found extreme value(fitness)	-2.51541×10^{-23}
Known extreme position	[5.75571×10^{-9} ;1.03158×10^{-8}]
Known extreme value(fitness)	-2.37464×10^{-32}

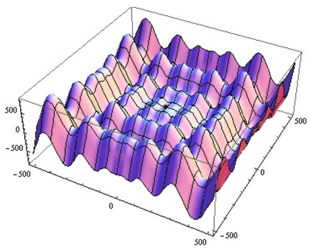


Rastrigin

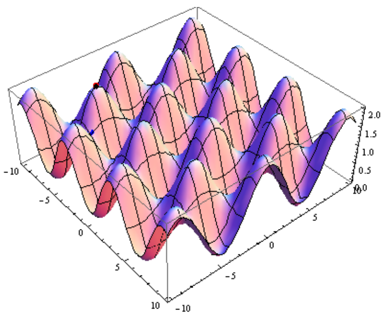
Start point(the blue point)	[-2.74829 ; 2.71306]
Start value(fitness)	-346.421
Number of iteration	103
Number of fitness calculation	3 410
Found extreme position	[1.15187×10^{-6} ; 7.50246×10^{-7}]
Found extreme value(fitness)	400
Known extreme position	[0.994959 ; 0.994959]
Known extreme value(fitness)	360.202



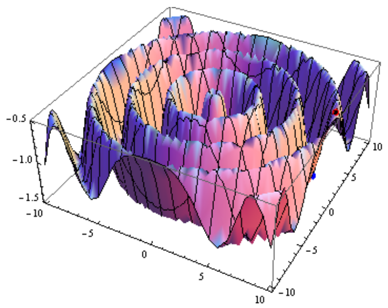
Schwefel

Start point(the blue point)	[216.4 ; -314.912]	
Start value(fitness)	-465.985	
Number of iteration	511	
Number of fitness calculation	15 810	
Found extreme position	[-5.2392 ; -5.2392]	
Found extreme value(fitness)	7.8906	
Known extreme position	[-5.2392 ; -5.2392]	
Known extreme value(fitness)	7.8906	

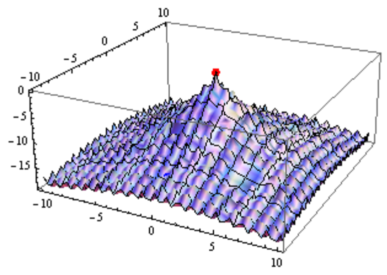
Griewangk

Start point(the blue point)	[-5.08967 ; -6.12683]	
Start value(fitness)	1.15253	
Number of iteration	36	
Number of fitness calculation	1 393	
Found extreme position	[-6.28636 ; -4.44736]	
Found extreme value(fitness)	2.01481	
Known extreme position	[3.14316 ; -1.57772×10^{-30}]	
Known extreme value(fitness)	2.00247	

SineWave

Start point(the blue point)	[8.68038 ; 0.890169]	
Start value(fitness)	-1.24946	
Number of iteration	15	
Number of fitness calculation	756	
Found extreme position	[9.73699 ; 1.92922]	
Found extreme value(fitness)	-0.500002	
Known extreme position	[-0.252603 ; -0.431499]	
Known extreme value(fitness)	-0.5	

Ackley

Start point(the blue point)	[-0.421008 ; -7.2625]	
Start value(fitness)	-14.9501	
Number of iteration	182	
Number of fitness calculation	5 780	
Found extreme position	[8.16448×10^{-15} ; -7.77028×10^{-16}]	
Found extreme value(fitness)	-2.13163×10^{-14}	
Known extreme position	[-1.7045×10^{-16} ; -1.76405×10^{-16}]	
Known extreme value(fitness)	0	

n-D space

Tests with n-D space were simulated in C# framework version 4.0.
As examples were choose DeJong1st and DeJong3rd 20D space.

Function	DeJong1st
Dimension	20
Start position	[2.29; -6.46; 8.99; -9.79; -2; 1.85; 3.47; 9.47; -5.58; 4.38; -5.13; 6.96; 8.55; -6.95; -0.22; 3.07; -1.42; 3.29; 6.53; -9.29]
Start value	-730.5273
Localized extreme position	[0.02069; -0.02549; 0.01073; 0.0062; -0.00207; -0.00333; 0.00918; 0.00413; -0.00014; 0.02934; 0.00782; -0.02393; -0.0278; -0.01543; -0.01986; 0.01003; 0.01591; 0.00935; 0.01579; -0.00932]
Localized extreme value	-0.005025
Known extreme position	[0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0]
Known extreme value	0

Function	DeJong3rd
Dimension	20
Start position	[-9.78; -2.22; -4.79; 1.69; -4.51; 8.87; -6.97; 4.22; -8.62; -0.78; -7.29; -7.37; 0.18; -8.88; 6.89; 5.92; 8.82; 4.56; 9.82; 6.83]
Start value	-119.01
Localized extreme position	[-0.01329; 0.00035; 0.0007; -0.0182; 0.00522; 0.0209; -0.00167; 0.00986; -0.00941; -0.00655; 0.00444; -0.04244; 0.02599; 0.00288; 0.00852; -0.0023; 0.00201; -0.00297; 0.01511; 0.00463]
Localized extreme value	-0.197437
Known extreme position	[0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0]
Known extreme value	0

Function	Rastrigin
Dimension	20
Start position	[-5.96; -7.62; -5.91; -1.79; -5.07; 6.99; -1.91; 4.23; 6.94; 8.87; 8.3; -0.36; 0.33; -5.39; -0.33; -7.48; 3.62; 2.68; -0.15; -8.97]
Start value	-193 840
Localized extreme position	[0.123554; 0; 0.345197; 0.853444; -0.094958; -0.114789; 0.652301; 0.200874; -0.320911; -0.390018]
Localized extreme value	339.879
Known extreme position	[0.;2.01948×10^{-28};0.;6.16298×10^{-33};-0.994959;6.46235×10^{-27};2.06795×10^{-25};0.994959;-0.421216;0.]
Known extreme value	340.202

APPENDIX B: Neural network training process – finger/face identification

Network with one hidden layer – 30 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 1

No. neuron in 1st hidden layer = 30

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 417

Duration = 00:00:06.22

Final training error = 1.8624 . 10^{-9}

```
############################ test output ################################

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
```

```
actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,0,1,1,0,0,0,0,1,1,0,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
```

```
actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,1,0,0,0,1,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with one hidden layer – 50 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 1

No. neuron in 1st hidden layer = 50

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 136

Duration = 00:00:03.47

Final training error = 7.2884 . 10^{-8}

```
############################ test output ################################

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
```

```
actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
```

```
actual  =   0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with one hidden layer – 100 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 1

No. neuron in 1st hidden layer = 100

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 42

Duration = 00:00:02.37

Final training error = 6.9461 . 10^{-11}

############################ test output ################################

```
actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
```

```
actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
```

```
actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with one hidden layer – 1000 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 1

No. neuron in 1st hidden layer = 1000

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 32

Duration = 00:00:37.39

Final training error = 9.4832 . 10^{-8}

```
########################## test output ##############################

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
```

```
actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
```

```
actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with two hidden layer – 500,100 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 2

No. neuron in 1st hidden layer = 500

No. neuron in 2nd hidden layer = 100

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 86

Duration = 00:00:49.30

Final training error = 1.0095 . 10^{-8}

############################ test output ################################

```
actual  =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual  =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual  =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual  =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
```

```
actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
```

```
actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual   =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal    =        0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with two hidden layer – 750,300 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 2

No. neuron in 1st hidden layer = 750

No. neuron in 2nd hidden layer = 300

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 81

Duration = 00:01:21.46

Final training error = $6.3389 . 10^{-12}$

```
############################ test output ################################

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
```

```
actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
```

```
actual  =     0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual  =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal   =     0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with two hidden layer – 1000,500 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 2

No. neuron in 1st hidden layer = 1000

No. neuron in 2nd hidden layer = 500

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 97

Duration = 00:02:29.17

Final training error = 3.2802 . 10^{-16}

########################### test output ###############################

```
actual   =      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
```

actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
actual	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal	=	0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
actual	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
actual	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
actual	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
actual	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal	=	0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
actual	=	0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal	=	0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0

```
actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with three hidden layer – 1000,1500,100 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 3

No. neuron in 1st hidden layer = 1000

No. neuron in 2nd hidden layer = 1500

No. neuron in 3rd hidden layer = 1000

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 602

Duration = 00:35:07.87

Final training error = 1.3241 . 10^{-14}

############################ test output ################################

```
actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
```

```
actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual   =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal    =      0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual   =      0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal    =      0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
```

```
actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual  =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal   =    0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

Network with three hidden layer – 1000,3000,100 neurons

No. samples = 31

No. neuron in input layer = 1400

No. hidden layer = 3

No. neuron in 1st hidden layer = 1000

No. neuron in 2nd hidden layer = 3000

No. neuron in 3rd hidden layer = 1000

No. neuron in output layer = 31

Acceptable error = 0.0000001

Accept error difference = 0.0000001

No. epoch = 148

Duration = 00:15:02.23

Final training error = 1.3980 . 10^{-08}

```
########################### test output ###############################

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

actual   =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal    =     0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
```

```
actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

actual  =   0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal   =   0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
```

```
actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

actual  =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal   =       0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

APPENDIX C: Backup file example

```
NEURAL NETWORK TEST
No. Fingerprints = 31
No. neurons in input layer = 1400
NN with 1 hidden layer
No. neurons in 1st hidden layer = 100
No. neurons in output layer = 31
Acceptable error value = 0.0000001
Acceptable error difference = 0.0000001

Start 28.10.2011 10:09:27

Epoch #1 Error : 0,276193424208195 errorDiff :-1(0)
Epoch #2 Error : 0,23659436390292 errorDiff :0,276193424208195(0)
Epoch #3 Error : 0,203445691317325 errorDiff :0,0395990603052754(0)
Epoch #4 Error : 0,159611919403142 errorDiff :0,0331486725855951(0)
Epoch #5 Error : 0,153573721884102 errorDiff :0,0438337719141832(0)
Epoch #6 Error : 0,151280502404022 errorDiff :0,00603819751903983(0)
Epoch #7 Error : 0,141251460469447 errorDiff :0,00229321948007999(0)
Epoch #8 Error : 0,133398253955433 errorDiff :0,0100290419345748(0)
Epoch #9 Error : 0,120885775516566 errorDiff :0,00785320651401433(0)
Epoch #10 Error : 0,122450448198426 errorDiff :0,0125124784388664(0)
Epoch #11 Error : 0,100807148580961 errorDiff :0,00156467268185974(0)
Epoch #12 Error : 0,0904302886922183 errorDiff :0,021643299617465(0)
Epoch #13 Error : 0,0755124988567182 errorDiff :0,0103768598887428(0)
Epoch #14 Error : 0,0637496323313809 errorDiff :0,0149177898355002(0)
Epoch #15 Error : 0,0583620538983033 errorDiff :0,0117628665253372(0)
Epoch #16 Error : 0,0492518640361818 errorDiff :0,00538757843307761(0)
Epoch #17 Error : 0,0357677339578212 errorDiff :0,00911018986212158(0)
Epoch #18 Error : 0,0311578994327217 errorDiff :0,0134841300783605(0)
Epoch #19 Error : 0,0226998043987406 errorDiff :0,0046098345250995(0)
Epoch #20 Error : 0,0192710360782683 errorDiff :0,00845809503398115(0)
Epoch #21 Error : 0,0156943403654152 errorDiff :0,00342876832047227(0)
Epoch #22 Error : 0,0124040068161883 errorDiff :0,00357669571285313(0)
Epoch #23 Error : 0,00995843561623019 errorDiff :0,00329033354922689(0)
Epoch #24 Error : 0,00479339286604866 errorDiff :0,00244557119995808(0)
Epoch #25 Error : 0,0126347616842386 errorDiff :0,00516504275018153(0)
Epoch #26 Error : 0,0127731298805257 errorDiff :0,00784136881818993(0)
Epoch #27 Error : 0,00385760212283293 errorDiff :0,000138368196287115(0)
Epoch #28 Error : 0,00517754476853115 errorDiff :0,00891552775769278(0)
Epoch #29 Error : 0,00476449168669168 errorDiff :0,00131994264569822(0)
Epoch #30 Error : 0,00554732546554842 errorDiff :0,000413053081839474(0)
Epoch #31 Error : 0,00271992345873342 errorDiff :0,00078283377885674(0)
Epoch #32 Error : 0,0020394888213752 errorDiff :0,002827402006815(0)
Epoch #33 Error : 0,00198295892495039 errorDiff :0,000680434637358225(0)
Epoch #34 Error : 0,000130167945201663 errorDiff :5,65298964248051E-05(0)
Epoch #35 Error : 0,000352371530487376 errorDiff :0,00185279097974873(0)
Epoch #36 Error : 3,65265738327438E-06 errorDiff :0,000222203585285713(0)
Epoch #37 Error : 0,000602590869251439 errorDiff :0,000348718873104102(0)
Epoch #38 Error : 0,0020814734840477 errorDiff :0,000598938211868165(0)
Epoch #39 Error : 0,00316802640733244 errorDiff :0,00147888261479626(0)
Epoch #40 Error : 0,000216022085657566 errorDiff :0,00108655292328474(0)
Epoch #41 Error : 4,0537586700043E-07 errorDiff :0,00295200432167488(0)
Epoch #42 Error : 6,94610593698264E-11 errorDiff :0,000215616709790565(0)



Test------------------------------------
 actual=0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0
ideal=0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0

 actual=0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0
ideal=0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0

 actual=0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0
ideal=0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0

 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1
```

```
 actual=0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0
ideal=0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0

 actual=0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1,1,1,1,0,0,0,0

 actual=0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,1,1,0

 actual=0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0

 actual=0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,0,1,1,0,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,0,0,1,0,1,0

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,1,0,1,0,1

 actual=0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
ideal=0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,1,0,0
```

```
End of training 28.10.2011 10:09:29

Duration 00:00:02.3671354

 -------------------------  Data errors  -------------------------------------
{0.276193424208195,0.23659436390292,0.203445691317325,0.159611919403142,0.15357
3721884102,0.151280502404022,0.141251460469447,0.133398253955433,0.12088577551
6566,0.122450448198426,0.100807148580961,0.0904302886922183,0.0755124988567182,
0.0637496323313809,0.0583620538983033,0.0492518640361818,0.0357677339578212,0.0
311578994327217,0.0226998043987406,0.0192710360782683,0.0156943403654152,0.012
4040068161883,0.00995843561623019,0.00479339286604866,0.0126347616842386,0.012
7731298805257,0.00385760212283293,0.00517754476853115,0.00476449168669168,0.00
554732546554842,0.00271992345873342,0.0020394888213752,0.00198295892495039,0.0
00130167945201663,0.000352371530487376,3.65265738327438 * 10 ^ -
06,0.000602590869251439,0.0020814734840477,0.00316802640733244,0.0002160220856
57566,4.0537586700043 * 10 ^ -07,6.94610593698264 * 10 ^ -11}

-------------------------  Data errors diff   -------------------------------
{-
1,0.276193424208195,0.0395990603052754,0.0331486725855951,0.0438337719141832,0.
00603819751903983,0.00229321948007999,0.0100290419345748,0.0078532065140143,0.
0125124784388664,0.00156467268185974,0.021643299617465,0.0103768598887428,0.01
49177898355002,0.0117628665253372,0.00538757843307761,0.0091101898621215,0.01
34841300783605,0.00460983452509950,0.00845809503398115,0.00342876832047227,0.00
357669571285313,0.00329033354922689,0.00244557119995808,0.00516504275018153,0.
00784136881818993,0.000138368196287115,0.00891552775769278,0.0013199426456982
2,0.000413053081839474,0.00078283377885674,0.002827402006815,0.000680434637358
225,5.65298964248051 * 10 ^ -
05,0.00185279097974873,0.000222203585285713,0.000348718873104102,0.00059893821
1868165,0.00147888261479626,0.00108655292328474,0.00295200432167488,0.00021561
6709790565}
```

APPENDIX D: Curriculum vitae

europass

Europass – Curriculum Vitae

Personal data

First name(s) / Surname(s)	**Malaník David**
Address(es)	Dřevnická 4126, 76001 Zlín (Česká republika)
Telephone(s)	+420 607 823 175
E-mail	david.malanik@gmail.com
Nationality	ČR
Date of birth	01. March 1984
Gender	Male

Work experience

Dates	01. June 2011 →
Occupation or position held	Ph.D. student - CEBIA-Tech
Main activities and responsibilities	Research team member
Name and address of employer	Tomas Bata University in Zlin
Dates	01. September 2008 → planned end of study 12/2011
Occupation or position held	Ph.D. student
Main activities and responsibilities	Teaching: Computer viruses and security, Security of information systems, Data security, Operation systems and

	his security. Supervision a bachelor and master thesis with aim to computer security Project accompanying
Name and address of employer	Tomas Bata University in Zlin
Dates	2006 - 2008
Occupation or position held	Lector
Main activities and responsibilities	Teaching the courses for the certification (Windows XP, Server 2003)
Name and address of employer	APS Brno s.r.o., Brno
Dates	2006 - 2007
Occupation or position held	Student researcher
Main activities and responsibilities	Creating of SW models for subject TPA. Software command tools for MAXON motors. Project supervisor: assoc. prof. Ing. František Hruška, Ph.D.
Name and address of employer	Tomas Bata University in Zlin

Education and training

Dates	2008 →
Principal subjects/occupational skills covered	Information technologies, Ph.D. IT Dissertation theme: Usability of the artificial intelligence and modern techniques for securing computer systems
Name and type of organisation providing education and training	Tomas Bata University in Zlin, Faculty of applied informatics (Ph.D.) Zlin
Dates	2003 - 2008

Title of qualification awarded	Bc., Ing.
Principal subjects/occupational skills covered	Information technologies
Name and type of organisation providing education and training	Tomas Bata University in Zlin, Faculty of applied informatics Zlin
Dates	1999 - 2003
Name and type of organisation providing education and training	Střední průmyslová škola Zlín Specialisation: Technical lyceum, IT
Dates	15/05/2011 - 17/05/2011
Title of qualification awarded	Absolvent
Principal subjects/occupational skills covered	Wireless Hacking
Name and type of organisation providing education and training	VERGILIUS IT Expert, s.r.o. Praha
Dates	15/11/2010 - 16/11/2010
Title of qualification awarded	Absolvent
Principal subjects/occupational skills covered	Hacking Unlimited
Name and type of organisation providing education and training	VERGILIUS IT Expert, s.r.o. Praha
Dates	01/10/2009 - 28/02/2010
Title of qualification awarded	
Principal subjects/occupational	Researcher/programmer

skills covered	
Name and type of organisation providing education and training	Haute Ecole de la province Liege Liege (Belgium)
Dates	2007 - 2007
Title of qualification awarded	Microsoft Certified Professional (MCP: 70-290)
Principal subjects/occupational skills covered	Managing and Maintaining a Microsoft Windows Server 2003 Environment
Name and type of organisation providing education and training	APS Brno s.r.o. Brno
Dates	2007 - 2007
Title of qualification awarded	Microsoft Certified Professional (MCP: 70-291)
Principal subjects/occupational skills covered	Implementing, Managing, and Maintaining a Microsoft Windows Server 2003 Network Infrastructure
Name and type of organisation providing education and training	APS Brno s.r.o. Brno
Dates	2007 - 2007
Title of qualification awarded	Microsoft Certified Professional (MCP: 70-270)
Principal subjects/occupational skills covered	Installing, Configuring, and Administering Microsoft Windows XP Professional
Name and type of organisation providing education and training	APS Brno s.r.o. Brno

Personal skills and competences

Other language(s)

Self-assessment	Understanding				Speaking				Writing	
European level (*)	Listening		Reading		Spoken interaction		Spoken production			
English	B2	independent user	B2	independent user	B2	independent user	B2	independent user	B2	independent user
Slovak	B2	independent user	B2	independent user	B2	independent user	B2	independent user	B2	independent user
French	A1	basic user	A1	basic user	A1	basic user	A1	basic user	A1	basic user
German	A1	basic user	A1	basic user	A1	basic user	A1	basic user	A1	basic user

(*) Common European Framework of Reference for Languages

Social skills and competences: communicativeness, time flexibility, thoroughness, creativity, responsibility

Organisational skills and competences: team management, project management, crisis management, coordination of e-learning projects, communication and cooperation with foreign team, coordination of projects implemented in different locations online EU

Computer skills and competences: Many years of experience with the administration and deployment of operating systems: Microsoft Windows, Linux, MAC OSX
Office applications: Microsoft Office, Open Office
Programming languages: C + +, C #, PHP, HTML, MYSQL, .NET
Security: OS Hacking, Wireless Hacking, penetration tests, Social engineering, Exploiting

Driving licence: B