DEMOCRITUS UNIVERSITY OF THRACE
SCHOOL OF HEALTH SCIENCES
DEPARTMENT OF MOLECULAR BIOLOGY & GENETICS

Application of Artificial Neural Networks to the crystallographic phase problem

Dionysia Moschou (1262)
Advisor: Nicholas M. Glykos

Final year thesis
Alexandroupolis 2017

Chapter 0
Abstract

X-ray crystallography is a tool used in order to determine small structures such as proteins. The atoms in a crystal under study cause a beam of incident X-rays to diffract and give a diffraction pattern. In order to produce a 3-dimensional picture of the density of electrons within a crystal, a crystallographer needs the amplitudes and also the phases of the diffracted waves. The information of those phases is lost resulting in the *phase problem*. Algorithmic operations such as *direct methods* have been developed to overcome this obstacle. These methods suggest direct relationships between the structure factors of a crystal. Artificial Neural Networks are computational models. They are presented as systems of interconnected *neurons* which exchange messages between each other. These connections have numeric *weights* that can be tuned based on experience, making them adaptive to *inputs* and capable of learning, therefore estimate the phases of the diffracted waves using only the observed intensities. The objective was for a network to be constructed in order that it learns the necessary algorithmic operations to estimate the crystallographic phase by means of only the directly observed experimental data. If neural networks could learn relationships as those used in direct methods, it would greatly enhance the work of crystallographers.

Περίληψη

Η κρυσταλλογραφία ακτίνων Χ αποτελεί ένα σημαντικό εργαλείο για τον προσδιορισμό μικρών δομών όπως οι πρωτεΐνες και τα νουκλεϊκά οξέα. Τα κρυσταλλογραφικά πειράματα χρησιμοποιούν κρυστάλλους από την υπό μελέτη δομή. Τα άτομα του κρυστάλλου προκαλούν μια δέσμη προσπίπτουσων ακτίνων Χ να εκτραπεί και να αποδώσει ένα πρότυπο περίθλασης. Το πρότυπο αυτό αποτελεί έναν έμμεσο τρόπο απεικόνισης μιας μικρής δομής. Προκειμένου να παραχθεί μια τρισδιάστατη απεικόνιση της πυκνότητας των ηλεκτρονίων στον κρύσταλλο, ο κρυσταλλογράφος χρειάζεται τα πλάτη και τις φάσεις των ανακλάσεων. Η πληροφορία των φάσεων δεν είναι δυνατό να αποκτηθεί άμεσα λόγω της πολύ μεγάλης συχνότητας των σκεδαζόμενων ακτινών. Η απώλεια αυτής της πληροφορίας αποτελεί το *πρόβλημα φάσεων.* Αλγοριθμικές εξισώσεις όπως οι *άμεσες μέθοδοι* έχουν αναπτυχθεί με σκοπό να ξεπεραστεί το εμπόδιο. Οι μέθοδοι αυτές συνιστούν μαθηματικές σχέσεις μεταξύ των παραγόντων μιας δομής, το πλάτος και τη φάση. Τα τεχνητά νευρωνικά δίκτυα αποτελούν υπολογιστικά μοντέλα βασισμένα στις λειτουργίες των βιολογικών νευρωνικών δικτύων που φυσικώς συναντώνται στους πολυκύτταρους οργανισμούς και συνιστούν το νευρικό σύστημα. Παρουσιάζονται ως συστήματα διασυνδεδεμένων *νευρώνων* που ανταλλάσσουν μηνύματα μεταξύ τους. Αυτές οι διασυνδέσεις αντιπροσωπεύουν τις συνάψεις των βιολογικών δικτύων και φέρουν αριθμητικές τιμές, τα *βάρη*, που μπορεί να συντονιστούν με βάση την εμπειρία, καθιστώντας τα δίκτυα προσαρμοζόμενα στις *εισόδους* και ικανά εκμάθησης. Ως εκ τούτου, θεωρείται δυνατή η εκπαίδευσή τους ώστε να εκτιμούν τις φάσεις των περιθλασμένων κυμάτων χρησιμοποιώντας μόνο τις παρατηρούμενες εντάσεις. Η ιδέα βασίζεται στις άμεσες μεθόδους που, όπως αναφέρθηκε πριν, συνδέουν άμεσα αριθμητικά τους παράγοντες του πλάτους (είσοδος) με τις φάσεις (έξοδος του δικτύου). Η εύκολη προσαρμογή των τεχνητών νευρωνικών δικτύων βασίζεται στην δυνατότητά τους να προσομοιώνουν συναρτήσεις που συσχετίζουν τις εισόδους με αντίστοιχες εξόδους κατά τη διαδικασία της εκπαίδευσης. Αυτή προϋποθέτει την προσαρμογή των βαρών των νευρώνων σε τιμές τέτοιες ώστε το δίκτυο να παράγει τις προτεινόμενες εξόδους. Στόχος της έρευνας είναι το δίκτυο να μάθει τις απαραίτητες αλγοριθμικές πράξεις μέσω αυτής της διαδικασίας ώστε να αποκτήσει την ικανότητα να προβλέπει τις φάσεις μιας κρυσταλλογραφικής δομής όταν μόνο μέσω των άμεσα παρατηρούμενων πειραματικών δεδομένων. Αν τα νευρωνικά δίκτυα μπορούσαν να μάθουν σχέσεις όπως αυτές που χρησιμοποιούνται στις άμεσες μεθόδους, το έργο των κρυσταλλογράφων θα ενισχυθεί σημαντικά.

Contents

Chapter 1

Introduction

The following thesis deals with the phase problem of crystallography and artificial neural networks' applications in terms of the defined problem. For understanding the purpose of this research and the correlation of these two quite distinct concepts, the section of the Introduction prefacing this work is consisted of two parts: the first will refer to the origins of artificial neural networks and the adaptive use of them, while the second will analyze the question of what crystallography and the phase problem are. The development of these basic concepts will prepare the reader for understanding the aim of this research explained in a third part of the introduction, and the method of experimental tests accompanying this work.

Part A: Artificial Neural Networks

1.1 Biological Neural Networks

Throughout the years, biological systems have been able to adapt, survive and flourish. Evolution has given then human brain features that include massive parallelism, distributed representation and computation, learning ability, generalization ability, adaptability, inherent contextual information processing, fault tolerance and low energy consumption *(Jain et al., 1996)*. These properties have impressed people who seek to emulate and adapt natural processes for application in the artificial world (*Lewis et al., 2009*). Advances in depeloping intelligent machines and the understanding of biological nervous systems leaded researchers to design artificial neural networks (ANNs) to solve a variety of problems in pattern recognition, prediction etc.

A detailed analysis of how ANNs work will require principles of neurophysiology, cognitive science, physics, computer science, statistics, pattern recognition, parallel processing and hardware (*Jain et* *al., 1996*). Artificial neural networks are in general a result of research in artificial intelligence and cognitive science. As crude approximations to biological neural systems, ANNs contribute to a new paradigm for dialogue among many of the disciplines of cognitive science. As suggested by their name, ANNs were initially introduced as computational models of simplified biological neural networks. For that reason, the principles of neuronal physiology and functions should be obtained before trying to understand how ANNs work.

1.1.1 Neuron

A neuron (etymology < ancient Greek *νεῦρον* sinew, cord, nerve, *Oxford English Dictionary, 2005*) is the fundamental unit which, together with glial cells, constitutes a neural system. Neurons' main importance is due to their ability to transmit and receive messages in the nervous system *(Lodish et al., 2000).* Glial cells consist of astrocytes and oligodendrocytes. Astrocytes are support cells in the mammalian central nervous system (CNS). Their main function is structural and metabolic support of the neurons *(Kimelberg & Nedergaard, 2010)*. Oligodendrocytes are also support cells and provide trophic support by the production of essential growth factors *(Bradl & Lassmann, 2010).*

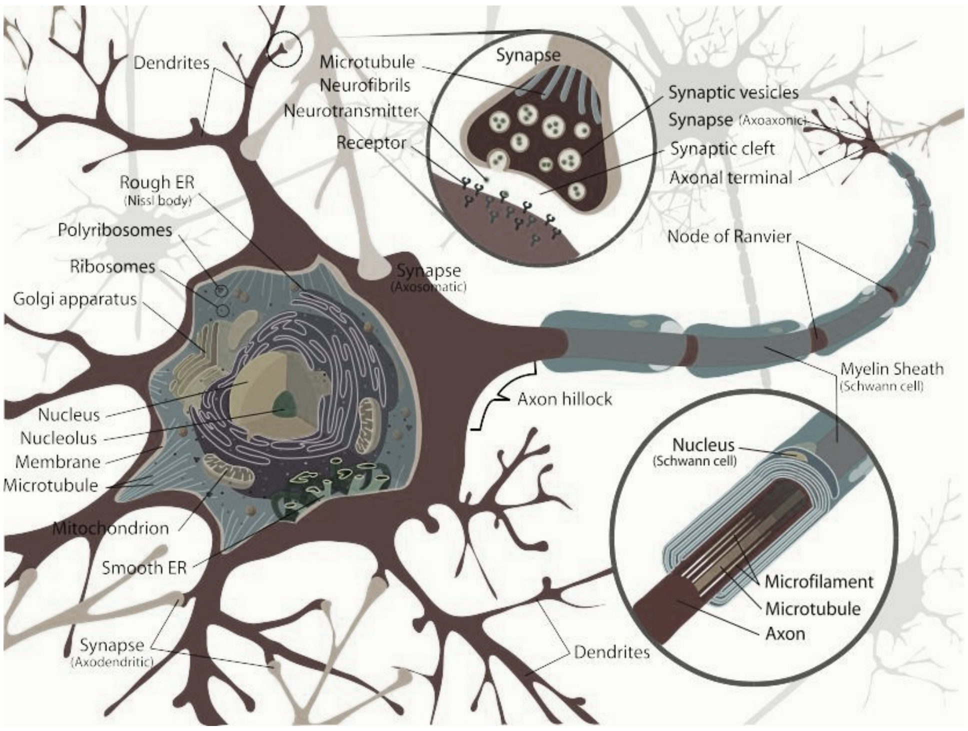


Figure 1: Complete neuron cell diagram. *(Work on public domain, 2007)*

Specialized types of neurons include sensory neurons, motor neurons and interneurons. A typical neuron consists of a cell body or soma, many dendrites, and an axon *(Fig. 1)* but their structure may vary depending on its specialized type. The nucleus is set in the soma, where most protein synthesis takes place. Its axon contains a great density of voltage-dependent sodium channels (*Rutecki, 1992*) and is often covered with myelin, an electrically insulating layer formed by an outgrowth of glial cells.

A neuron is an electrically excitable cell that processes and transmits information through electrical or chemical signals. The signals, or impulses, occur via specialized structures between neurons where the axon terminal of one cell contacts another neuron's dendrite, soma or axon *(Jain et al., 1996)*. These connections are called synapses. The human brain contains hundreds of billion neurons (*Herculano-Houel, 2009*). Neurons connect to each other so that they form neural networks.

The nervous system comprises the brain and spinal cord (the central nervous system; CNS) and sensory and motor nerve fibres that enter and leave or are wholly outside the CNS (the peripheral nervous system; PNS) *(Fig. 2)*.

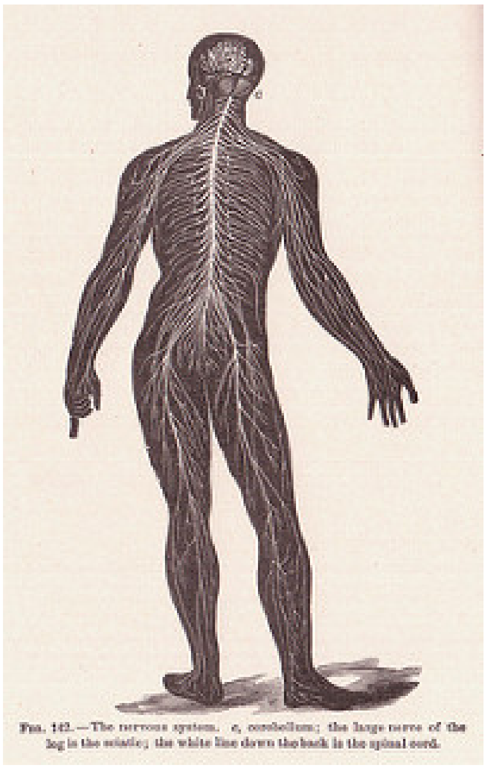

c
FIG. 142.—The nervous system. c, cerebellum; the large nerve of the leg is the sciatic; the white line down the back is the spinal cord.

Figure 2: Illustration of the nervous system. *c, cerebellum; the large nerve of the leg is the sciatic; the white line down the back is the spinal cord. From the book The Human Body and Health Revised by Alvin Davison, published in 1908 by Alvin Davison. Work on public domain.*

1.1.2 Synapses

The synapse is the functional unit of the brain *(Jain et al., 1996)*. Synapses are junctions between a pre-synaptic (transmitter) and a post-synaptic (receiver) neuron. There are two kinds of synapses depending on the form of the transmitted signal: electrical and chemical. There are fine differences between the two. Electrical signal may be inversely transmitted through a gap junction without delay, while the most common chemical signal uses a chemical mediator to be released and subsequently binded by its receptor.

Synaptic signals may be excitatory or inhibitory. Whether the excitation received is large enough, the neuron generates a brief pulse called an action potential which originates in the soma or the dendrites and travels through the axon to the axon terminal (*Kandel et al., 2013*). Electrical synapses allow neurons to communicate directly by electrically conductive junctions between two cells. The percentage of electrical activity eventually transmitted at the post-synaptic end is the synaptic weight *(Theodosi-Kokkinou, 2013)*. In a chemical synapse, when the action potential reaches the axon terminal, it triggers calcium uptake by opening its voltage-gated calcium channels. Calcium then causes the neurotransmitter to release into the synaptic cleft. The neurotransmitters activate receptors on the post-synaptic neuron. The target cell responds to the electrical or chemical signal depending on its specialized type (*Kandel et al., 2013*).

1.1.3 The principles of learning

Learning is the process in which knowledge is acquired or modified as a result of experience *(Kihlstrom et al., 2007)*. Eric Kandel mentioned in 2000 the idea first suggested by Santiago Ramon y Cajal *(1894)* that learning results from changes in the strength of the synapse. Plastic changes occur due to the synapses' ability to strengthen or weaken in response to their activity. This often results from the alteration of the number of neurotransmitter receptors (*Gerrow K. & Triller A., 2010*). Hebbian theory (*Hebb Donald, 1949*) suggests a basic mechanism for synaptic plasticity, where an increase in post-synaptic efficacy arises by repeated and persistent stimulation from the pre-synaptic cell. This persistent excitation results in some growth process or metabolic change in one or both cells. The main idea is that when a driving neuron participates in firing a subsequent neuron, the connection strength between the neurons will increase, whereas it will decrease in the absence of correlated firing *(Kowaliw et al., 2014)*. Synaptic plasticity is not only a fascinating area in neuroscience but also constitutes an important medical issue. Many neurodegenerative disorders such as Alzheimer's disease or dementia have synaptic effects and a great number of psychiatric medications exert their action on the synaptic receptors.

The molecular mechanisms for synaptic plasticity are linked to long-term potentiation (LTP), the long-lasting changes in the efficacy of a synaptic connection *(Kandel & Schwartz, 1982)*. The mechanisms include the activation of protein kinases. The activated forms phosphorylate post-synaptics receptors, improving cation conduction and thereby potentiating the synapse. Moreover, genes like *activin* β_A are upregulated *(Kandel & Schwartz, 1982)*. The protein derived from this gene alters the cytoskeletal structure by elongating the dendritic spine, which results in an increasing chance of synaptic contacts and the maintenance of LTP *(Shoji-Kasai et al., 2007)*.

1.1.4 Establishing long-term memory

There are two main types of long-term human memory: implicit (procedural) and explicit (declarative) memory *(Cohen & Squire, 1980)*. Implicit memory is acquired non-consciously, while explicit memory represents the actual knowledge of people, places and objects and can be used consciously *(Ullman, 2004)*. Explicit memory are therefore conscious memories that can easily be recalled. Explicit memory can be further identified as episodic or semantic memory. Episodic memory consists of information about personal experience, while semantic memory refers to facts and ideas in general. Semantic memory is acquired through associations and is stored in discrete regions of the cortex *(McLeod, 2012).*

In order for the explicit memory to be processed, information is associated with existing knowledge. The procedure takes place in the left prefrontal cortex. The information is further modified by protein synthesis in the neuron. The memory is therefore stored and may be recalled. Short-term memory uses the information after decoding long-term memory. When explicit memory needs to be retrieved, a kind of short-term memory also known as working memory is able to process the stored information *(McLeod, 2012).* The hippocampus is a temporary way station for long-term memory. The processing of information initially takes place in the unimodal or multimodal association areas of the cerebral cortex *(Kandel et al., 2013).*

Moreover, implicit memory is retrieved non-consciously and is involved in training reflexive motor or perceptual skills. Both sides of temporal lobes are associated with implicit memory. Implicit memory represents behaviors which are acquired by establishing associations between the behaviors and repeated environmental stimuli *(Kandel et al., 2013)*. One example of creating implicit memory is explained by an experiment conducted by Edward Thordike *(1898)* in which a cat attempted to randomly open a door. After many attempts, the cat had learned how to open the door *(trial and error)*. This procedure is also an example of learning.

1.2 Artificial neural networks

1.2.1 Biologically inspired computational intelligence

Having previously briefly described the principles of biological neural networks, an artificial model of them would be introduced; artificial neural networks. Advances in computational intelligence have motivated scientists in developing intelligent systems inspired by biological neural networks. ANNs are generally characterized as computational models with properties such as adaptability and learning ability, generalization and ability of parallel processing *(Kröse & Smagt, 1996)*. Artificial neural networks, from now on referred simply as neural networks, are statistical learning models originally designed as simplified models of the brain function. Interest in biological processes have provided scientists with better understanding of biological properties and the knowledge gained was used to improve computational systems *(Navlakha & Bar-Joseph, 2011).* For that reason, it is accepted that improvements in the area of artificial intelligence simultaneously triggers improvement on the knowledge of biological processes *(Navlakha & Bar-Joseph, 2011)*.

Humans throughout the ages have always been concerned about the brain functions. Not only neuroscientists but also computer scientists were interested in features of human cognition and wanted to embed the knowledge in their computational models. Computers usually work following a fixed program containing an algorithm. However, this in not the only way for a computational machine to work. Certain problems cannot be solved using a simple algorithm but depend on many subtle factors. Brains can calculate a variety of factors, but traditional computers cannot.

Evolution have given human brain properties that lack from the standard Von Neumann architectures. Brain's adaptivity allows it to learn by means of training samples after a period. Therefore, biological neural networks are capable to generalize and associate data upon training. Brain is therefore able to solve problems of the same class. The feature of pattern recognition relies on this principle. The distinction between two different objects is possible even if the exact objects are not previously presented. The ability of generalization in turn results in a degree of fault tolerance *(Navlakha & Bar-Joseph, 2011)*. The property refers to the ability of a system to produce reasonable responses when it is provided with noisy input data, or when the system has internal errors. External factors such as alcohol or drugs may cause destruction of neurons, yet cognitive abilities are not significantly affected *(Kriesel, 2007).* Furthermore, in spite of the high speed of computer's processing, humans can effortessly solve complex perceptual problems faster *(Jain et al., 1996).* This is due to structural differences when highly interconnected neurons consisting the network working in parallel enables the brain to achieve high performances. In contrast, a conventional computer works sequentially. While in parallel computing multiple processes execute at the same time, sequential processes are run one after another in a succession fashion *(Mivule, 2011)*. For that reason, the latter form of programming reaches the same result slower than in parallel computation.

Artificial neural networks are able to mimic the abilities of biological neural networks and adapt the mentioned properties. Because of this, neural networks are broadly used by scientists so that they approach problems by teaching computers to solve them. However, training a neural network consumes considerable amount of time and money. Not only the implementation of the network is complex, but also great processing and storage resources are required. Whether a realistic biological neural network was designed using artificial neural networks, the amount of inter-connections between neurons would demand great amounts of computer memory, hard disk space and processing power. Despite the limitations of the neural networks, advances in computer technology have allowed their usage to solve complex industrial and scientific problems in fields such as engineering, medicine and finance.

Besides the mentioned similarities, both biological and artificial neural networks consist of simple building units, the neurons, which serve as computational devices, though artificial neurons are much simpler that the biological ones. However, the most important feature that biological and artificial neural networks share is their learning ability *(Eluyode et al., 2013)*. Learning influences the strength of the connections between the neurons due to the synaptic plasticity *(Kowaliw et al., 2014)*. These synaptic connections basically determine the function of the whole network; upon training procedure, the appropriate connections between neurons should have been determined (*Hagan et al.,2014).* The goal of a learning system is to acquire a function that maps input events to output responses *(Matheus & Hohensee, 1987).*

1.2.2 Architecture of a neural network

An artificial neural network is an information-processing system that has certain performance characteristics in common with biological neural networks. A neural network consists of numerous simple procession elements called neurons, units or nodes *(Gereshenson, 2003)*. Each neuron is connected with other neurons by means of directed communication links. Imitating synapses' function, the links are associated with numerical values, the weights. Weights represent the strength of the synapses and may be altered due to synaptic plasticity. The information being used by the network to solve a problem is stored as the values of the weights.

Neurons accept and transfer signals. Those signals are the input and the output values respectively. Neurons have an internal state called the activation. The activation is a function of the inputs it has received, it is therefore called activation function. Whether the activation is identical to the output, the neuron would be called linear. However, this is rarely the case. Several activation functions have been developed that produce the output, which depends only on the activation. The activation in turn depends on the values of the inputs and their respective weights *(Gereshenson, 2003)*. The output of the neuron is transferred to one or more neurons, but only one at a time. In some cases, the neuron has also an external input called bias. Bias is an input value usually set to 1 that does not represent a variable of the problem, nor a signal from other neurons, but is also connected with the neuron through a weighted link. The interactions of neurons through the connections lead to a global behavior of the network *(Gereshenson, 2003)*.

A single input neuron with bias is represented in the *figure 3*. The input p is multiplied with the weight w and is sent to a summer Σ. All values are scalars. The external input is weighted by a bias b and is also sent to the sumer. The summer, or net output n goes into a function f which produces the output a of the neuron. Although the function is set by the designer of the neural network, scalars w and b are both adjustable parameters of the neuron and are adjusted upon training to help the output meet a specific goal.

In favor of biological synaptic signals, positive values of the weight *w* are considered as an excitation of the neuron, while negative values as inhibition *(Gereshenson, 2003)*. The process of adjusting the weights is called learning or training.

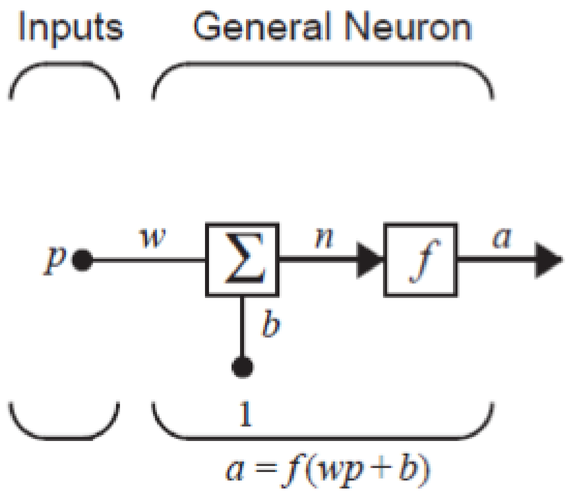


Figure 3: A simple one-input neuron representation with bias. *Equation at the bottom of the figure is the matrix form of the following equation 1 (Hagan et al.,2014).*

In most problems, neural networks usually implement more than one values of inputs. In this case, the output *a* of a neuron which has *R* inputs $p_1, p_2 ... p_R$ with weights corresponding to these input $w_1, w_2 ... w_R$ and a bias *b* will be:

$$a = f\ (b + \textstyle\sum_{j=1}^{R} w_j p_j) \qquad \textit{Equation 1}$$

Due to the large amount of processing data, a single neuron does not provide the necessary flexibility in a neural network to adjust its weights and approach acceptable solutions. A larger amount of neurons is usually needed to operate in parallel in what is called a layer of neurons. Both the output and the biases include *S* values as the number of neurons *(Fig. 4)*. However, the number of neurons in a layer is not always equal to the number of the inputs. Also, the activation function in a layer may alter among neurons. This property is similar to combining more of one neural networks in parallel.

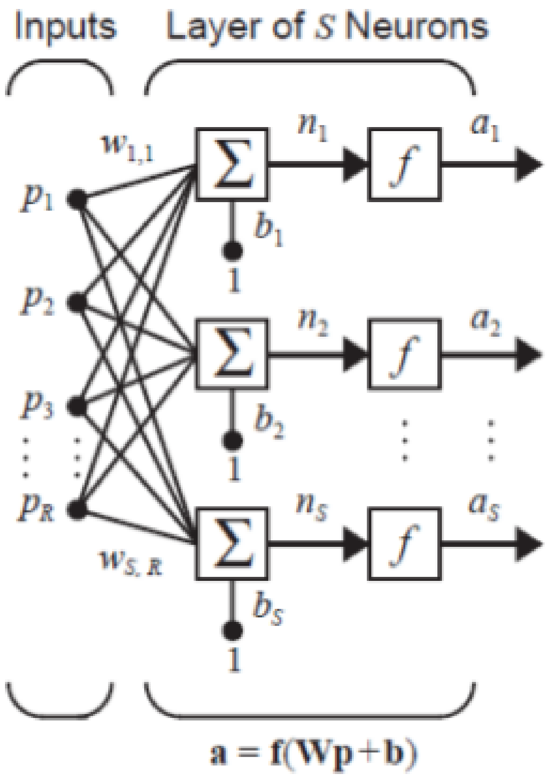


Figure 4: A layer of S neurons with R inputs. *S biases are used and S outputs are produced (Hagan et al.,2014).*

Layers of functions can also be combined to create various architectures of neural networks. Multilayer neural networks are layers of neurons combined in series. In this case, the output of the first layer serves as input to the second layer, and the output of the second layer serves as input to the third layer. The last layer is usually called the output layer. The first and second layers are called hidden layers, because there is no access to them trough inputs or outputs *(Fig. 5)*.

The connections in this neural network are characterized as directed links, in the sense that signals are always propagated from the first layer to the second and from the second to the third, and not in the opposite direction *(Matheus & Hohensee, 1987).* These networks are called feedforward networks. Bidirectional neural networks may also be implemented, when the information flows in either sense.

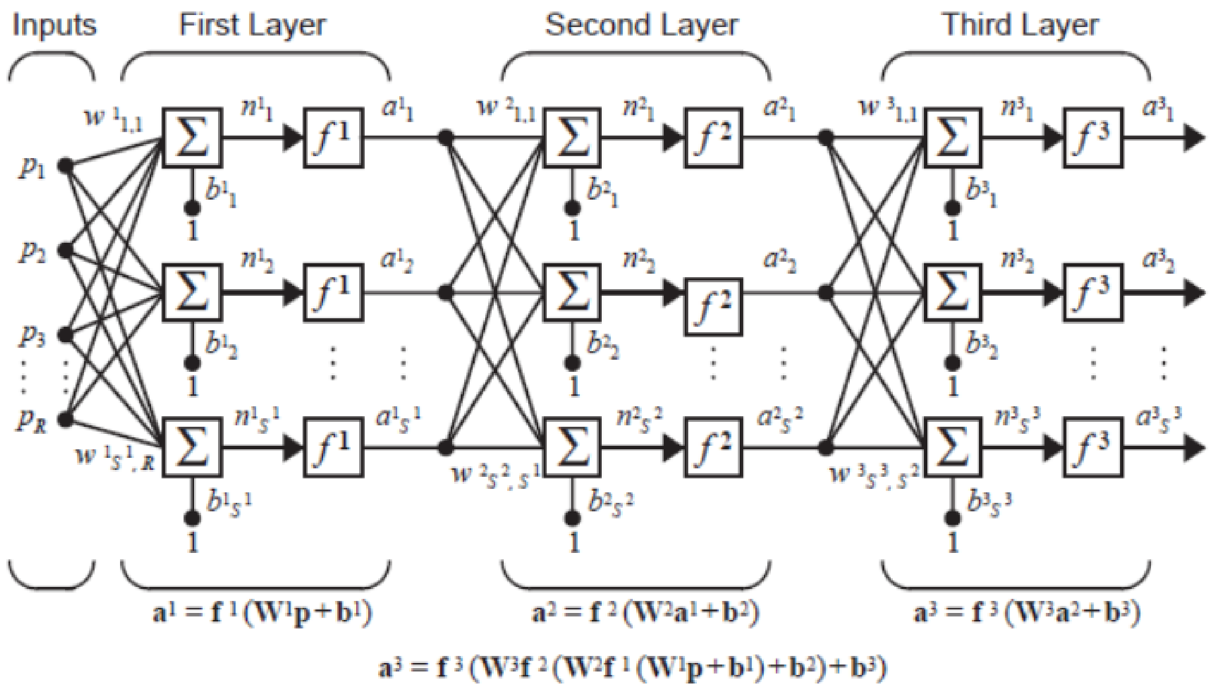


Figure 5: A multilayered neural network. *Three layers constitute the network. Each layer's output a is theinput of the next layer. Third layer, or output layer, produces the final a^3S output (Hagan et al.,2014).*

1.2.3 *Training an artificial neural network*

As mentioned before, the values of the weights possess the information used by the network to solve a class of problems. The values are altered and determined upon training, meaning that the aim of the training procedure is the adaptation of a neural network's weights to meet the needs of a particular problem. There are several ways of training an artificial neural network. The procedures are unsupervised training, reinforcement learning and supervised training.

In the first case, the neural network is being trained by means of only the input data. In unsupervised training, the network is trying to identify similar patterns and to classify them into similar categories. No output data is presented to the network *(Matheus & Hohensee, 1987)*. In comparison, in reinforcement learning the network receives the real value of output after network completion of a sequence and then decides whether the produced result is accurate *(Kowaliw et al., 2014)*.

Supervised learning, however, is the procedure where both input and output data is presented to the network before training. Training data is presented as training patterns of the input values and the corresponding, desired, correct output values. The network reads the data and then the output can be directly compared with the correct solution while the network's weights are being adjusted according to the difference *(Kowaliw et al., 2014)*. The purpose of the training is to alter the weights in a way that the network will not only remember the correct values for the inputs it has been trained with, but also to provide plausible answers to unknown, similar input data. This process is called generalization *(Matheus & Hohensee, 1987)*.

Backpropagation algorithms have been developed so that a feedback of error is propagated backwards in layered feedforward networks. The algorithm constitutes parallel distributed processing and is used in supervised learning. The algorithm evaluates the network's performance by calculating the difference between the produced and the desired, correct results via an error function. The function produces an error value which is used to guide the modification of weights appropriately *(Matheus & Hohensee, 1987)*. The idea of the backpropagation algorithm is to reduce this error, until the network learns the training data. The training process begins with random weights, and the goal is for them to be adjusted so that the error will be minimum *(Gereshenson, 2003)*. Weight initialization, though, has been possible via initialization algorithms that adjust the weights in an initial value previously determined to substantially reduce the initial error *(Yam & Chow, 2000)*.

Part B: Crystallography

1.3 Principles of crystallography

X-ray crystallography is a tool with which scientists can identify the atomic and molecular structure of a crystal. Those kinds of structures are not to be seen with the usual methods due to their small components of matter (atoms and molecules). X-ray crystallography uses X-rays (x-radiation), a form of electromagnetic radiation *(Als-Nielsen & McMorrow, 2011)*. Microscopes also use electromagnetic radiation to picture small objects, though their resolution is low. In order for two defined details of an object to be seen with a microscope, they should be separated by at least half the wavelength of the radiation used to view them *(Porter, 1906)*. Objects under study are atoms separated by distances of the order of 1 Å (0.1 nm). For atoms to be seen, an appropriate radiation should be used. X-rays are this kind of radiation, with wavelengths ranging from 0.1-100 Å *(Als-Nielsen & McMorrow, 2011)*. For the reasons mentioned, X-rays are a much more suitable type of radiation to view atoms rather than visible light.

Microscopes are designed so that a beam of radiation falls into the object under study and this radiation is then scattered by the object *(Drenth, 2007)*. The scattered radiation is then recombined by an appropriate lens system and results in an image of the scattering matter appropriately magnified. An optical microscope using visible light, recruits a series of lenses which diverge and focus the light passing through them, while an electron microscope uses magnetic lenses in an analogous matter. On a smaller scale, X-rays whose wavelengths are much shorter are scattered by the electrons in atoms. Scattered X-rays cannot be focused normally by any currently known experimental technique, hence direct visualization of an image cannot be conducted.

For that reason, the phenomenon of diffraction is an alternative but indirect way to view the molecules. Diffraction involves solids in a crystalline arrangement *(Müller, 2008)*. This crystalline arrangement is essential because the diffraction pattern of a single molecule is too weak to be observed. When molecules are arranged in a crystal, this pattern is easily observed at specific points due to the repeating units (molecules). The points have a direct relationship with the shape and form of the crystal. A diffraction pattern has potentially the same information as a direct image, though it is not easily observed. The result of the analysis of this pattern is a complete three-dimensional elucidation of the arrangement of atoms in the crystal under study. The information is obtained as two elements; the first one is the atomic positional coordinates while the second is the atomic displacement parameters. The first indicates the position of each atom in the repeated units of the crystals. From these a scientist is able to precisely calculate inter-atomic distances and angles of the atomic components. The latter element indicates the extent of atomic motion or disorder in the molecule *(Chatzidimitriou, 2015)*.

Diffraction effects are observed only if the obstacle used is not greater than the wavelength of the waves that impinge on it. Each wave has a periodic displacement, which is called the amplitude of the wave, and a distance between crest, which constitutes its wavelength *(Serway & Jewett, 2013)*. When an object scatters a beam of radiation, the disturbance of the beam may be assessed by the knowledge of the relative phases (phase difference) of all the scattered waves *(Als-Nielsen & McMorrow, 2011)*. This is due to the fact that the relative phases have a profound effect on the intensities of the scattered beams because of the principle of superposition, meaning that a total wave disturbance is the sum of the individual disturbances due to each source *(Sherwood & Cooper, 2010)*. The beams scattered by an object or crystal upon an X-ray experiment are captured on a detector, or photographic films. The relative phases and intensities of the scattered beams are determined by nature and depended on the atomic arrangement in the crystal. X-ray detection devices, however, can only sense the intensity of the waves. The values of the corresponding phases cannot be determined experimentally *(Hauptman, 1983)*.

In X-ray diffraction scattered beams by electrons in the crystal are essential in order to see small objects such as atoms. The diffraction pattern can be directly obtained on the detection device by measuring the intensities of the scattered beams. However, the relative phases cannot be determined. The lack of the phase information is called the phase problem. Only upon knowledge of the lost phases, crystallographers would be able to recombine all the information (intensities and phases) with proper mathematical computation techniques (Fourier synthesis) and obtain an electron density map which in turn gives information about the structure of the molecules in the crystal under study.

1.3.1 Crystals

Crystals are defined as solids composed of a regularly repeated arrangement of atoms *(Chatzidimitriou, 2015)*. Solids which possess well-defined, three-dimensional ordering of molecules which are in proximity to one another and have relatively strong interactions between them belong to a state known as crystalline state *(Sherwood & Cooper, 2010)* and results in its macroscopic geometrical shape. Crystals are bound by plane faces as a consequence of the regular stacking of molecules in layers. Every face in a crystal represents a plane parallel to a molecular layer. In order that a molecular structure of a material is studied with X-ray diffraction, crystals are to be used.

A regularly repeated structure such the internal of a crystal is called the motif. The motif is a structural unit which may be a single molecule, a group of several molecules or a group of ions. A conceptual array of points which defines the geometrical relation between the motifs is called the lattice *(Sherwood & Cooper, 2010)*. A crystal structure is therefore expressed as the convolution of the lattice and the motif. Two-dimensional lattices are called plane lattices while three-dimensional lattices are called space lattices. Both kinds of lattices are defined in terms of crystallographic unit vectors and crystallographic unit cells. Unit cells constitute the basic building units of a crystal and their repeated pattern represents the regularity and symmetry of the lattice.

An example of a unit cell in three dimensions is depicted in figure 6 which is described by the unit vectors a, b, c which define the crystallographic axes respectively. Any point in a three-dimensional lattice may be described using the crystallographic unit vectors as *(Eq. 2)*:

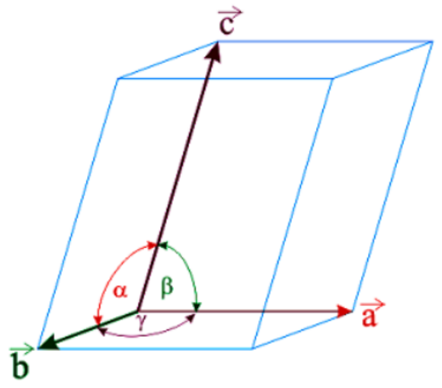


Figure 6: 3-D crystallographic unit cell with axes a,b,c** (Chatzidimitriou, 2015)**.

$$r = p\vec{a} + q\vec{b} + r\vec{c}$$ *Equation 2*

Crystal structures may be characterized by a set of symmetry elements which is an important property of well-ordered, geometrical objects. The symmetry of the lattices is such that in two dimensions only five (5) lattices exist *(Fig. 7)*, while in three dimensions fourteen (14) lattices are defined *(Ladd & Palmer, 2013)*.

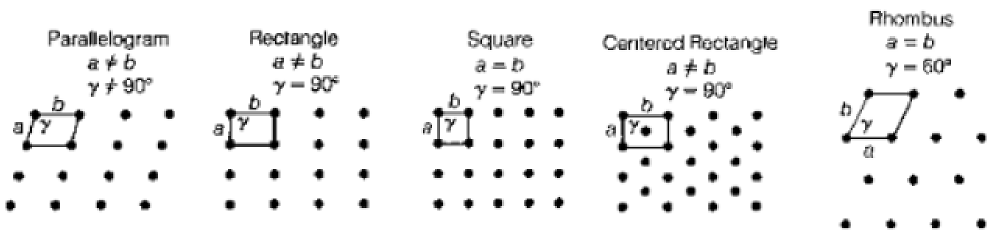


Figure 7: The five (5) 2-D lattices described by crystallographic unit vectors a,b** (Chatzidimitriou, 2015)**.

An example of the oblique p2 space group is illustrated above as appeared in *International Tables for X-ray crystallography*. The twofold axis is at the origin of the cell and it will reproduce one of the structural units. Symmetry elements are represented at the right-hand diagram; the twofold axis manifests itself in two dimensions as a center of symmetry. Three additional centers of symmetry are generated at the points (x, y) = (½,0), (0,½) and (½,½). The four centers of symmetry are all different in that the structural arrangement is different as seen from each of them *(Fig. 8)*.

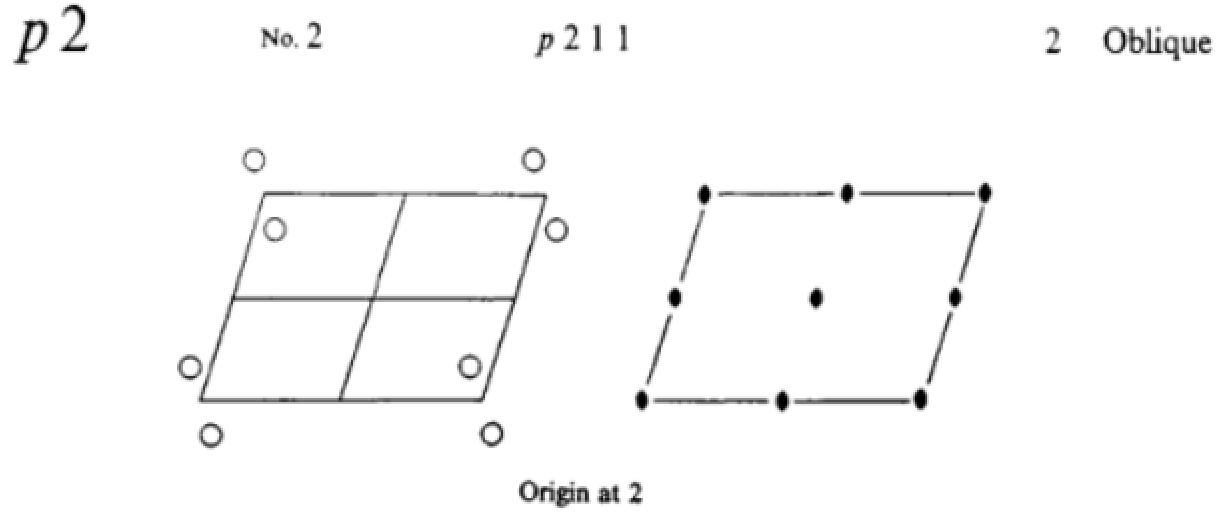


Figure 8: The 2-D space group p2.

1.3.2 X-rays as electromagnetic waves

Both X-rays and visible light are electromagnetic waves where an electric field E and a magnetic field B oscillate in a wave-like form in two mutually perpendicular planes *(Serway & Jewett, 2013)*. The difference between light and X-rays is the range of their wavelengths, with X-rays having much shorter ones.

A simple wave may be expressed as a sine wave by the equation:

$$y(t) = A\sin(2\pi f t + \varphi) = A\sin(\omega t + \varphi) \qquad \textit{Equation 3}$$

where *A* is the amplitude, meaning the peak deviation of the function from zero, *f* is the frequency, *ω* equals *2*π*f* and represents the angular frequency and *φ* is the phase in radians. The phase is an expression of relative displacement between the corresponding wave and an origin.

1.3.3 Defining an electron density map

When an electromagnetic wave passes through matter, the electric field E causes charged particles such as protons and electrons to oscillate. The oscillation of a particle creates wave-like disturbances in the electric field. In consequence, the particles become secondary sources of electromagnetic radiation. This effect is known as scattering. The combination of scattering of X-rays in crystals gives rise to the macroscopic event of diffraction *(Sherwood & Cooper, 2010).*

X-ray scattering is mainly a result of electrons *(Sherwood & Cooper, 2010)*. The high frequencies of the X-rays ranging from $3x10^{16}$ Hz to $3x10^{19}$ Hz and the weight of protons being much more than of the electrons are the reason that electrons scatter more efficiently X-rays to the point where the total diffraction pattern is considered a product of electron scattering only. The diffracted waves form a pattern which is determined by the nature and structure of the obstacle. This pattern of radiation scattered by the object is called its diffraction pattern.

The diffraction pattern contains information of the structure of the diffracting obstacle *(Chatzidimitriou, 2015)*. If an obstacle can be described by an amplitude function *f(r)*, where *r* is a vector in two or three-dimensional space and *f(r)* is zero everywhere outside the boundaries of the obstacle, then the diffraction pattern amplitude *F(k)* is given by the Fourier transform of *f(r)*:

$$F(\vec{k}) = \int_{all\ \vec{r}} f(\vec{r})e^{i\vec{k}\vec{r}}\, d\vec{r} \qquad \textit{Equation 4}$$

This equation gives us the diffraction pattern if the obstacle (*f(r)*) is known. Conversely, if the diffraction pattern is known, then using the Fourier inversion theorem, the amplitude function *f(r)* can be inferred in terms of *F(k)* as follows:

$$f(\vec{r}) = \int_{all\ \vec{k}} F(\vec{k})e^{-i\vec{k}\vec{r}}\, d\vec{k} \qquad \textit{Equation 5}$$

The definition of the electron density map of the crystal is the purpose of a crystallographic experiment. The electron density map of a crystal is a three-dimensional description of the electron density in a crystal structure *(Usón & Sheldrick, 1999)*. This map describes the contents of the unit cells averaged over the whole crystal and not the contents of a single unit cell *(Drenth, 2007)*.

The diffraction of X-rays by the contents of the unit cell is determined by the electrons. Since the unit cell is associated with the lattice and the diffraction pattern is sampled at specific regions (the reciprocal lattice points *hkl*) the amplitude of the diffraction pattern is associated with particular lattice points. These specific amplitudes of the diffraction pattern at the reciprocal lattice points are called structure factors and are given by:

$$\vec{F}_{hkl} = V \iiint_0^1 \rho(x,y,z)\, e^{2\pi i\,(hx+ky+lz)}\ dx\, dy\, dz \qquad \textit{Equation 6}$$

where *V* represents the volume of the unit cell and *ρ(x,y,z)* is the electron density map as a function of position *x, y, z*. The set of structure factors is in fact the Fourier transform of the electron density map *ρ(x,y,z) (Cochran, 1952)*. Whether the structure factors are determined as a result of an X-ray crystallographic experiment, the electron density map may be conducted through an inverse Fourier transform. The electron density map is therefore given by:

$$\rho(x,y,z) = \frac{1}{V} \sum_h \sum_k \sum_l \vec{F}_{hkl}\, e^{-2\pi\iota\,(hx+ky+lz)} \qquad \textit{Equation 7}$$

The calculation of the electron density map using structure factors is called Fourier analysis and it is the ultimate purpose of a crystallographer who performs a crystal structure analysis.

1.3.4 The phase problem

The aim of a crystallographer who performs an X-ray diffraction experiment, is to determine the molecular structure of a crystal by calculating the electron density function according to the Fourier synthesis. In order to calculate this equation the values of the structure factors are all needed; their magnitude and phase:

$$F_{hkl} = |F_{hkl}|\, e^{\iota\, \alpha_{hkl}} \qquad \textit{Equation 8}$$

where a_{hkl} represents the phase of the structure factor. Therefore, the calculation of the electron density map requires the knowledge of all structure factors in both magnitude and phase. In X-ray diffraction experiments however, only the intensities of the reflections are measured, and information of the relative phases is lost because the intensity of a structure factor is equal to $|F_{hkl}|^2$. The phases cannot be obtained directly from physical measurements *(Hauptman, 1983)*. Thus half the information is lost during this procedure. The lack of the phase information constitutes the phase problem *(Taylor, 2003)*.

The relationship between structure factor magnitudes and electron density may be defined from the molecular structure or electron density. Therefore, the values of the phases can be obtained by a prior knowledge of the electron density or structure *(Taylor, 2003)*. Various methods have been developed to deduce the relative phase angles of the structure factors. The interpretation of inter-atomic vectors is a method in which an initial trial structure is obtained through Patterson and heavy atom methods. From these structures calculated phase angles can be derived *(Ladd & Palmer, 1980)*. An alternative method is the use of isostructural crystals. Two isomorphous crystals are used consisting the same atoms expect for a replacement of one or more atoms in one structure with different types of atoms than the other, such as heavy atoms, or the presence of one or more additional atoms in one of them. Phases can be therefore estimated by comparing such crystals and this method is the method of choice for macromolecular phase determination *(Chatzidimitriou, 2015)*.

A third method of phase determination are direct methods. These methods are based on the positivity and atomicity of electron density and leads through statistical methods to phase relationships between normalized structure factors *(Hauptman, 1983)*. Direct methods are mathematical techniques that attempt to solve the phase problem from the observed amplitudes through purely algorithmic relationships, with no recourse to any structural information *(Hauptman, 1997)*. This is the feature that these techniques have in common with the neural networks, since the neural network will not have access to structural chemical information upon training but only previously presented to observed amplitudes, just as the case of direct methods.

Direct methods have already the ability to solve small-molecule structures of up to about 100 atoms *(Usón & Sheldrick, 1999)*. The methods are based on two basic criteria. The first is that given a set of observed amplitudes $|F_{hkl}|$, the corresponding phases must be such as to produce non-negative electron density values everywhere. The second criterion is that, if the atoms is a structure are identical and in close distance but do not overlap in space, a relationship among diffracted amplitudes and therefore among phases exists *(Hauptman, 1997)*. The second criterion may be extended in a probabilistic sense to the case of unequal atoms *(Giacovazzo, 2013)*.

Direct methods involve the comparison of the structure factors magnitude and in oder to generalize the mathematics, the normalized structure factor E_{hkl} (E-value) can be derived corresponding to each amplitude *(Giacovazzo, 2006)*. The advantage of *E-values* is that it has been shown that its use is equivalent to structure points but without suffering from thermal motion *(Drenth, 2007)*. In centrosymmetric crystals, all structure factors F_{hkl} are real, as a result of the Fourier transform properties *(Sherwood & Cooper, 2010)*. This means that the structure factors have 0 or π phases and therefore can be assigned with positive or negative signs respectively. In consequence, the corresponding E_{hkl} is assigned with the same sign.

Another powerful approach to the problem of phase determination is provided by the probability relationships. An algorithmic equation shows the relationship between the structure factor and the sign in order that phase information is extracted. These sign relationships are based on the Sayre's equation:

$$F_{hkl} = \varphi_{hkl} \sum_{h'} \sum_{k'} \sum_{l'} F_{h'k'l'} \; F_{h-h',k-k',l-l'}$$ *Equation 9*

The implication of Sayre's equation is that any structure factor F_{hkl} is determined by the products of all the pairs of structure factors whose indices add to give (*hkl*) *(Sayre, 1952)*. Sayre pointed out that for the case where F_{hkl} is large, the series must strongly tend towards one direction (positive or negative in centrosymmetric cases) and that this direction is generally determined by the agreement in sign among products between large structure factors. Sayre's equation can be therefore extended in the case of non centrosymmetric crystals. Certain linear combinations of the phases are defined as following:

$$\varphi_{-h-k-l} + \varphi_{h'k'l'} + \varphi_{h-h',k-k',l-l'} \approx 0$$ *Equation 10*

where φ_{hkl} is the phase of the *hkl* reflection. This is known as the sum of angles formula and reflections *(-h-k-l), (h'k'l'), (h-h' k-k' l-l')* define a triplet of reflections whose angles are related *(Ladd & Palmer, 1980)*. The relative phase combinations are of great importance in direct methods and they are called structure invariants, because they are uniquely determined by the crystal structure but are independent of the choice of origin *(Hauptman, 1983)*.

Part C: Objective

Crystallography as a tool provides great knowledge to molecular biologists regarding features of proteins, DNA and RNA, and the relationships among such molecules by allowing the structural depiction of them and analyzing their functions. Despite the development on X-ray crystallography, crystallographers face a fundamental obstacle; the phase problem. The phase problem is called upon the lack of the phases' information after an X-ray crystallographic experiment. Various techniques of phase determination have been developed to overcome the problem. Direct methods is an example technique that uses solely the observed intensities of the diffraction pattern of a crystal without prior knowledge of the molecule under study or other structural and chemical information.

Artificial neural networks are algorithmic models with the ability to learn by means of the observed data. This feature is similar to the algorithmic operations represented by the direct methods. Therefore, neural networks might be able to learn the appropriate relationships between the observed data, thus the amplitudes, and their corresponding phases suggested by direct methods. A trained neural network with the ability to approximate estimate the phases on a set of amplitudes regarding a diffraction pattern would be a great tool and subsequently enhance the work of crystallographers.

The aim of this research is to examine whether neural networks could be sufficiently trained to reach the specific target. To this end, hypothetical 2-dimensional, centrosymmetric structures were used to simplify the problem. Neural networks were implemented and trained upon structural data of the simplified structures. Then, the networks were evaluated for their performance, thus their ability to produce the desired, correct phases corresponding to a set of amplitudes. Whether the network has the ability to be trained for the simplified structures, it could be possible for additional research to be done for more complicated structures.

Chapter 2

Network's implementations

The nature and arrangement of the connections between neurons determined both what types of functions a network is able to compute and what it is able to learn with respect to a given learning rule *(Matheus & Hohensee, 1987)*. For the purpose of this research, neural networks were implemented using Fast Artificial Neural Network (FANN) Library. FANN is an open source library which implements multilayer neural networks. It supports cross-platform execution and it is easy to use with example networks supplemented in the installation package. The parameters are easy to optimize through simple functions *(Nissen, 2005)*. While FANN works in many programming languages, C was preferred and used for both creating, training and testing the networks. Experiments took place in Linux based Ubuntu operating system. FANN's latest version 2.20 was used for all trials.

The main benefits of using FANN Library was the speed and the simplicity of its networks. A newly created network can be trained on a training data set consisting of the input and the output data. When the data set is introduced to the network, the weights of the network are adjusted to the information presented in order that it gives the same outputs as seen in the training data. The aim of the training process is to minimize the mean square error (MSE) of the training data; this would happen by adjusting the weights appropriately and would be evaluated by investigating whether the trained network produces the desired, correct output. Genetic algorithms have been introduced to assess the problem. Backpropagation algorithms examine the current MSE on each training epoch and therefore adjust the weights to minimize it.

However, training may result to an over-fitted network. The term suggests a network that has memorized the relationship between the input and the output giving precise results for the training data, but cannot predict the correct output on other data resulting in poor generalization performance. In order to prevent over-fitting, the network would be evaluated every epoch by using a percentage of the training data with which the network was assessed for its accuracy *(Mitchel, 2005)*. At the end of the training process, the network was tested on another group of data sets, the test data. Testing the network examines its accuracy to predict the correct signs of the output upon the completion of the training.

2.1 Structures under study

For the purpose of this research, simple structures were studied. More specifically, 2-Dimensional centrosymmetric structures in oblique p2 crystals were employed to assess the network's ability to process and learn the appropriate phase relationships. For the simplified problem, the phases were not examined altogether for the employed structure factors but the network assessed one phase at a time; for a set of amplitudes, the phase in question was chosen to be a structure invariant. As mentioned before *(paragraph 1.3.4)*, structure invariants are characterized by linear phase relationships suggested by Sayre *(1952)* when he employed large structure factors in his equation. By assessing one phase at a time and selecting the phase that corresponds to a structure invariant, the complexity of the problem was reduced.

A program in C has been created to produce the structures of interest. The program sets random x, y coordinates for the position of the atoms. The arrangement of the defined number of atoms in a defined unit cell produce the structure factors, according to the given maximum resolution at 1.0 Å which determines how many the reflections are. A minimum fractional separation between the atoms is also taken into account and set at 0.15. The factors produced are based on an equation *(Eq. 11)* that gives the structure factors of the unit cell with an origin of coordinates at the point 00, the center of symmetry in the particular unit cell.

$$F_{hk_{0,0}} = \sum_{j=1}^{N} f_j \cos 2\pi(hx_j + ky_j) \qquad \textit{Equation 11}$$

Afterwards, the amplitudes (F_{hk} for two dimensions) are turned into *E-values*. E-values E_{hk} represent the amplitudes but without the effects of the temperature factor. The temperature factor represent the uncertainty for each atom. The reason for the selected input data is to simplify the problem.

Two different crystals were used for this research's experiments. Their dimensions were 2.0x3.0 Å and 7.0x12.0 Å. Structure factors were studied for 2 or 4 atoms in the respective unit cells. Random structures were created representing the mentioned conditions. The program had an output of *hk* reflections defined both the amplitude and the phase. For the centrosymmetric structures under study, the phase has the form of a positive or a negative sign.

Fourier synthesis was performed on structure factors produced by the program using the 'Pepinsky's machine' program. The electron density maps of some experimental structures are presented *(Fig. 9)*.

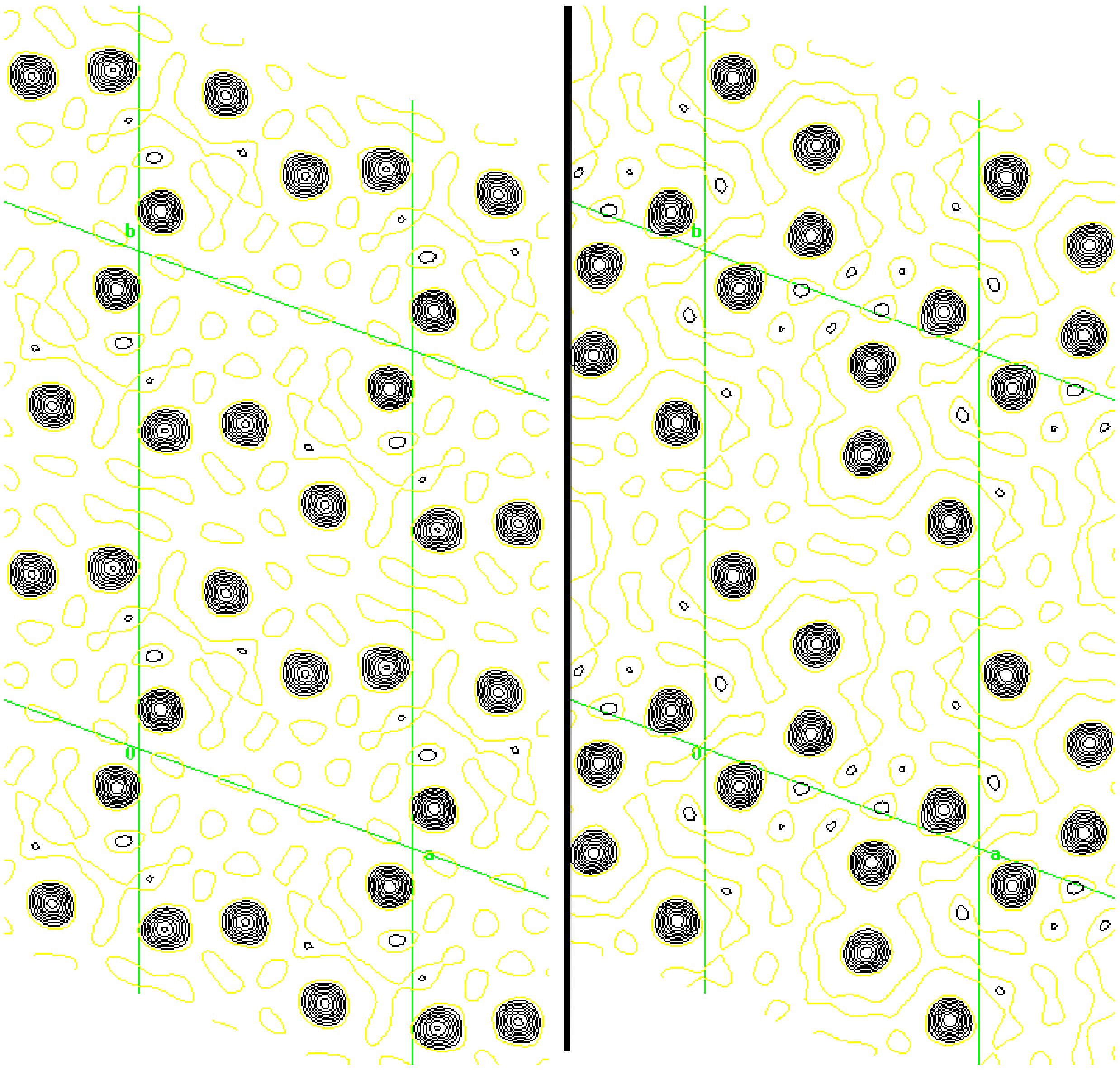


Figure 9: The electron density maps of two random structures created to be used for training neural networks. *There are presented structures of 4 atoms in a 7.0x12.0 Å unit cell.*

Data sets were created using the program in C. Each data file contained multiple training sets or patterns, a couple of input data and the corresponding sign of the structure factor being negative when phase=π or positive when phase=0. The number of the training sets used, the number of the input values and the number of output values were written at the first line of the data set. Following that, each training set has been displayed into two lines; first was the input and then the desired output. The number of the input data was determined by the number of the reflections. While the input data was numerous, the output data was a single digit with a positive or a negative sign.

In this simple case, the data structure factors produced by the program in C were 7 when 2 atoms are employed and 126 when the atoms are 4. The difference on the amount of the structure factors are not only because the number of the atoms alters, but also for the size of the cell is importantly smaller in the first case.

The output of each data set was assigned to be the sign of a single amplitude given as structure factors. The amplitude of interest was chosen to be a structure invariant. The desired outputs had a bipolar form (1 or -1). When the structure factor is positive, the desired output value is +1 and when it is negative, it is -1. The network may produce outputs with low accuracy, as long as these outputs are positive or negative.

A data set was fed into the network during its training. The network used the information stored in the data file in order to associate the input values, meaning the E-values and therefore the structure factors, with the correct, desired output, thus the sign of a single structure invariant reflection. Upon completion of the training process, the network should be able to estimate the correct sign corresponding to a newly introduced input set.

2.2 Creating the network

2.2.1 Training algorithms

The overall objective of a neural network is to be able to be familiar with problems by means of only the training and, upon sufficient training, to be able to solve unknown problems of the same class. For this purpose, an artificial neural network could adapt to these problems and therefore learn by imitating the learning process of a biological neural network. In this case supervised learning will be used, because the neural networks are going to be trained with examples with known, inputs and outputs. The training process aims to alter the strength of the connections, develop new connections or delete existing connections. A connection may be deleted by assigning a zero value to its weight, while an inactive connection can be activated by changing its zero value to something else. Higher values of weight means a stronger connection.

To this end, a training algorithm is needed which is going to adjust the weights and biases to suitable values. An issue to be determined is when the weights are adjusted during the training process. Incremental training updates the weights and biases after each data pattern is presented. In batch training, the weights and biases are updated only after all the data sets are presented. A total error of the patterns is calculated by means of an error function operations and the weights are adjusted properly. Both approaches are referred as backpropagation learning. While the first approach is expected to be faster, the latter seems to achieve more precise results when the problem is more complex *(Nissen, 2003)*.

Advanced batch algorithms, RPROP (resilient backpropagation) and QUICKPROP, have been developed in order to adjust weights faster and more efficiently. RPROP algorithm is a direct adaptive method of faster backpropagation learning. The main difference between RPROP and QUICKPROP is that the latter uses advanced parameters such as learning rate which cannot be manually adjusted in the adaptive method of RPROP. Both algorithms are expected to achieve good results for a variety of problems.

Both mentioned algorithms are based on a mathematical concept called the gradient, a general form of the derivative when several variables are used. Gradient descent algorithms include some measure of error the value of which will change as the value of one weight changes when other weights or biases' values are not altered. A gradient is made up of several partial derivatives. The sign of a partial derivative for a weight indicates how the weight should be altered. For instance, when the sign of the partial derivative of a weight is positive the weight should go negative and vice versa. RPROP algorithm adjusts the weights during training by means of only the sign of the gradient, meaning the partial derivative over all patterns.

When the input values are presented, they are propagated forward through the network to compute the output value. This output is produced based on the current network's weights and is then compared with the output values suggested by the training set. An error signal is estimated and is passing through the network as a feedback so that the error signal alters the weights appropriately. The process propagates the errors back one layer and is repeated for every layer. The whole procedure constitutes an epoch. Afterwards, the weights are adjusted is a direction that minimizes the error between the outputs of the network and the desired values. Several epochs are needed in order to produce a minimum of the error. The training process is finished when an acceptable error is produced. For this thesis' experiments, a minimum error at 0.001 is required while 1000 epochs are designed to occur unless otherwise mentioned.

A main disadvantage of gradient descent algorithms is that the error often reached a local minimum value rather than a global minimum on the error surface *(Matheus & Hohensee, 1987)*. The error surface has multiple local minimums and the gradient operations may pause in a local minimum value without ever reaching a global minimum. Another issue could be when a flat error surface is presented, a great amount of training epochs are required in order to achieve the desired minimum value. However, a local minimum is acceptable as long as it has a satisfactory value.

In order to solve this issue, the values of the weights may be adjusted before training using an initialization algorithm. Initialization of the weights can promote the network to reach the desired error faster and more precisely. A common random weight initialization is the Nguyen-Widrow's initialization algorithm. The initialization algorithm generates initial weights and bias values so that the active regions of the neurons are distributed approximately evenly over the input space. In order for the algorithm to perform, it examines the range of the input data and therefore adjusts the weights of the neurons so that each neuron will learn more readily. When initialization is attempted during this thesis' experiments, it occurs once before the training process.

2.2.2 Function approximation

Neural networks are capable of function approximation meaning that they learn a function by means of only the observed data. In order for the network to learn a problem, it must be defined as a function with a set of input and output supported by examples of how this function should work. The input and the output values consisting a data set represent the variables of the function is question. The network learns the function that associates the input with the output values by slowly adjusting the weights of its neurons in every layer to produce the same output as in the example data set.

A way to enhance network's training speed and precision is to set the range of the input and the output values *(Nissen, 2003)*. To this end, activation functions may be defined. Since the default values used in neural networks are [0,1] and the desired output values are [-1,1], activation functions should be used that set the range of the output value at [-1,1].

For that reason, symmetric functions were selected which have output that ranges from -1 to 1. Four functions were used; sigmoid symmetric *(Eq. 12)*, gaussian symmetric *(Eq. 13)*, elliot symmetric *(Eq. 14)* and linear piece symmetric *(Eq. 15)* activation functions.

$$f(x)=\frac{2}{1+e^{-2sx}}-1 \qquad \textit{Equation 12}$$

$$f(x)=e^{-(xs)^2}-1 \qquad \textit{Equation 13}$$

$$f(x)=\frac{xs}{1+|xs|} \qquad \textit{Equation 14}$$

$$f(x)=xs \qquad \textit{Equation 15}$$

where *x* is the input to the function, *f(x)* is the output value and *s* the steepness. The steepness of an activation functions characterizes how fast the function will go from the minimum to the maximum. While the parameter has a range from 0 to 1, meaning the extreme values when 1 is the highest speed, the default activation steepness is 0.5 and was not altered before the experiments.

Among all activation functions tested, only FANN Linear piece symmetric has a range including the limits -1 and 1, when all others are ranged between (-1,1). All four functions were compared to the default activation function FANN Sigmoid stepwise having an output range [0,1]. The functions were evaluated for their efficacy to help the network adapt to problem when used in hidden or/and output layer.

2.2.3 Network's size and connectivity

A very first step on designing the network is to define its size. Whereas the number of neurons in the input and the output layer is determined by definition of the problem, the number of neurons in the hidden layer as well the amount of the hidden layers used may vary. The procedure in order to decide about those two key features is largely a matter of experience. With many problems, accuracy can be sufficiently obtained with one or two hidden layers and a few hidden neurons in both layers. In case neurons are less than the number compared to the complexity of the problem, the network will not be able to detect the signals in the data set. The complexity of the problem relates to the number of the weights and biases. On the other hand, if the network consists of too many hidden neurons or layers, it is likely to memorize the data by resulting to over-fitting.

The number of the hidden neurons required cannot be evaluated without looking through the complexity of the problem. To this end, a number of neural networks with different sizes were tested in order to identify the approximate suitable number of the two characteristics. The number of neurons in the hidden layers were ranging from a few neurons to some hundred neurons. One to seven hidden layers were also tested with a small number of neurons in each layer.

One way to lower the complexity of a network while sustaining its size is to adjust the parameter of the connection rate *(Nissen, 2003)*. The default value of the parameter is 1 representing a fully connected, dense network. When this parameter is 0.5 for example, the network will contain the same amount of neurons but only half as many connections. Networks with less connectivity than of a dense one are called sparse *(Kowaliw et al., 2014)*. Trials were conducted with sparse networks compared to dense networks when connection rate is 0.8, 0.5 and 0.25.

2.3 Training the network

As mentioned before, designing a multilayer network has the ultimate aim to become familiar with problems by assessing the functional relationships between the input and the output training data and therefore solve similar problems. This process is called generalization. In agreement with Ockham's razor, the key strategy for obtaining good generalization is to find the simplest model that explains the data *(Zimmermann et al., 2002)*.

There are many methods to sufficiently achieve the network's training so that it provides plausible answers; growing, pruning, global searches, regularization and early stopping. Growing methods start with zero neurons and gradually add some in the network until it achieves an adequate performance. On the contrary, pruning methods start with a great number of neurons in the network and stepwise remove neurons or weights until the performance is significantly poor *(Islam & Murase, 2001)*. Global searches assess all possible network architectures to determine the simplest model that explains the data. The rest of the methods, regularization and early stopping, sustain the network small and simple by constraining the magnitude of the weights but not the amount of them.

In the first three approaches, growing, pruning, and global searches, over-fitting may occur for the networks are larger. In order to avoid this, early stopping method was used. Early stopping is the simplest method with an overall idea that as training progresses the network uses more and more of its weights, until all weights are fully used when training reaches a minimum of the error surface. If training process is terminated earlier, fewer parameters would be used preventing over-fitting.

In order to determine when the training procedure should be stopped, a method called cross validation takes place. Cross validation uses 10% of the data set (cross validation set) in order to evaluate the degradation of the error value. The training process should stop when the error value does not decrease. In this case, the error value was calculated using the mean squared error function (*MSE*):

$$mse = \frac{1}{n}\sum_{i=1}^{n}(t_i - o_i)^2 \qquad \textit{Equation 16}$$

where t_i is the value of the ith output of the neural network and o_i is the desired output. The number of outputs is n. The same error function is used to evaluate the performance on the training data.

Another measure of performance during training is the number of failed bits. The bit fail limit is a parameter that represents the maximum allowed difference between the desired and the produced output value during training and cross validation. Each output that diverges more than this limit is counted as an error bit. In symmetric activation functions like the ones used in the experiments, the difference is divided by two. The number of failed bits represent the inability of the network to predict the desired output values. For that reason, it will be used for evaluation is some cases but not many of them as there are more preferable methods to do so.

There are four (4) conditions for a training process to stop. Early stopping suggests the first one when the error returned from the validation process do not decrease. Another condition is when the network reaches a desirable performance. Training would also stop when the error value received on the training data is really small (0.001 in this case). Lastly, the training process is set to take place for a maximum number of epochs which is reached when the other conditions are not satisfied.

Upon completion of the training procedure the network, meaning the values of the adjusted weights, will be printed in a file in the current directory.

2.4 Testing the network

In order to finally evaluate the performance of a trained network, the network is executed using a test set. The test set represents 10% of the data set and the network runs forward using the input values of the test set. The network produces output values performing the functions learned during training. The produced output values are then compared to the desired ones in order to assess its efficacy to correctly predict them. The desired output values, therefore the correct phases of the structure factors that consist the data sets, are 1 or -1. However, the network would considerably perform well when the signs of the phases are correctly predicted. A network that produces output values equal or larger that 0 for positive phases and values smaller than 0 for negative phases is considered successful. The percentage of the correctly produced outputs indicates the correctly predicted phases and is consequently calculated and presented in the results section.

Chapter 3

Results

The objective of the current research thesis was to build a neural network in C and train it to learn the necessary algorithmic operations in order to estimate the crystallographic phase by means of only the directly observed experimental data. For that reason, the data given as 'input' represent each time the reflections of a hypothetical 2-Dimensional structure in centrosymmetric p2 crystals. All data set contain the E-values of the amplitudes of those reflections. The network receives the E-values and the respected phase of a structural invariant reflection. After multiple training cycles often set 1000, each neural network was assessed for its satisfactory outcome by its Mean Square Error (MSE) or further tested using 10% of the presented data set of inputs.

Experiments were conducted for structures of 2 and 4 atoms. Structures of one atom were not suitable to be examined due to their little variety on structural reflections.

3.1 Experiments with structure factors of 2 atoms as input data

Data sets represent the E-values of the reflections of hypothetical 2-Dimensional structure in centrosymmetric p2 crystals. Cell is 2.0x3.0 Å and a minimum fractional separation between the atoms is set at 0.15. The maximum resolution is 1.0 Å.

Multiple training cycles were attempted in order to examine parameters such as the appropriate training algorithm used from the network, the activation function, the number of hidden layers and neurons that will be able to process the operations and the number of input data sets that will be enough in order that the network associates the input with the respective output, but not too many to manage an over-fitting network.

The data sets contain 7 reflections as input and one output. The output is 1 or -1 representing a positive or negative phase respectively for the reflection 02. The reflection under study was chosen as a structure invariant detected for it is a large structure factor in multiple structures.

3.1.1 Activation function

70 data sets were fed into a neural network when the training algorithm is set as default (RPROP) and using only one hidden layer with 7 neurons in it. Symmetric activation functions were chosen to be tested for their span [-1,1], for the expected output is -1, 1. Training was attempted for different activation functions in hidden or output layer. When one of each is changed, the other one remains default (FANN Sigmoid stepwise).

Experiments with different activation functions in hidden layer led to the conclusion that FANN Elliot symmetric function in hidden layer gives a lower Mean Square Error (MSE) than Gaussian symmetric, Linear piece symmetric and Sigmoid symmetric *(Fig 10)*. The mentioned functions cannot be determined for their results because of the activation function set in output layer. For that reason, the experiment was re-conducted for different activation function in output layer, while hidden layer remains default (Sigmoid stepwise) or set as FANN Elliot symmetric.

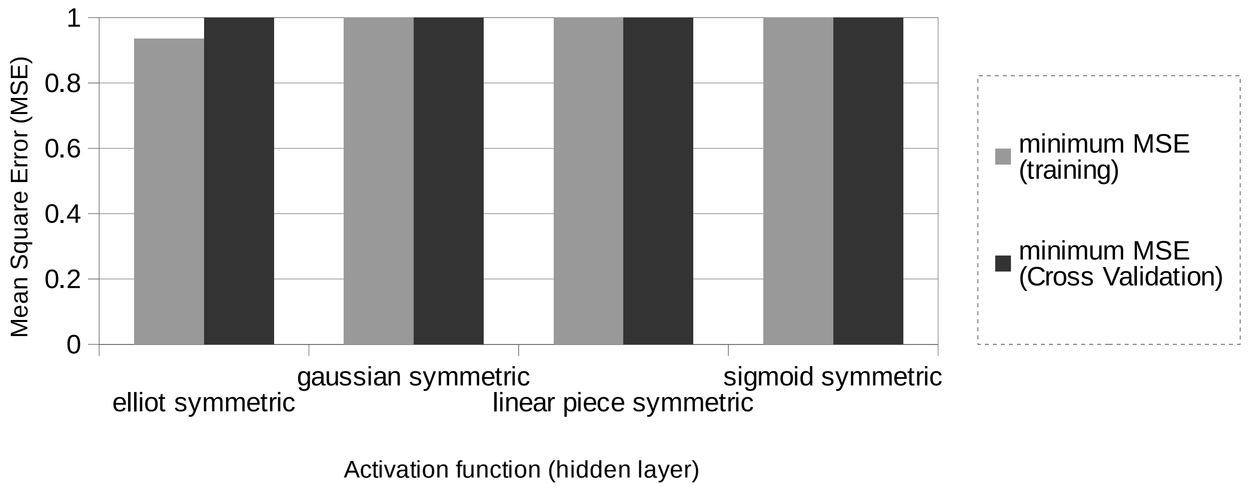


Figure 10: FANN activation functions for hidden layer do not have a great affect on the outcome when activation function for output layer is default. *However, FANN Elliot symmetric give a lower MSE for the training. Bar graphs represent the MSE.*

Different activation functions on output layer have multiple effects on the MSE of training and cross validation. While MSE of training is low, MSE of cross validation is variable *(Fig. 11)*. In order to determine for the most suitable activation function, they were associated with FANN Elliot symmetric on hidden layer.

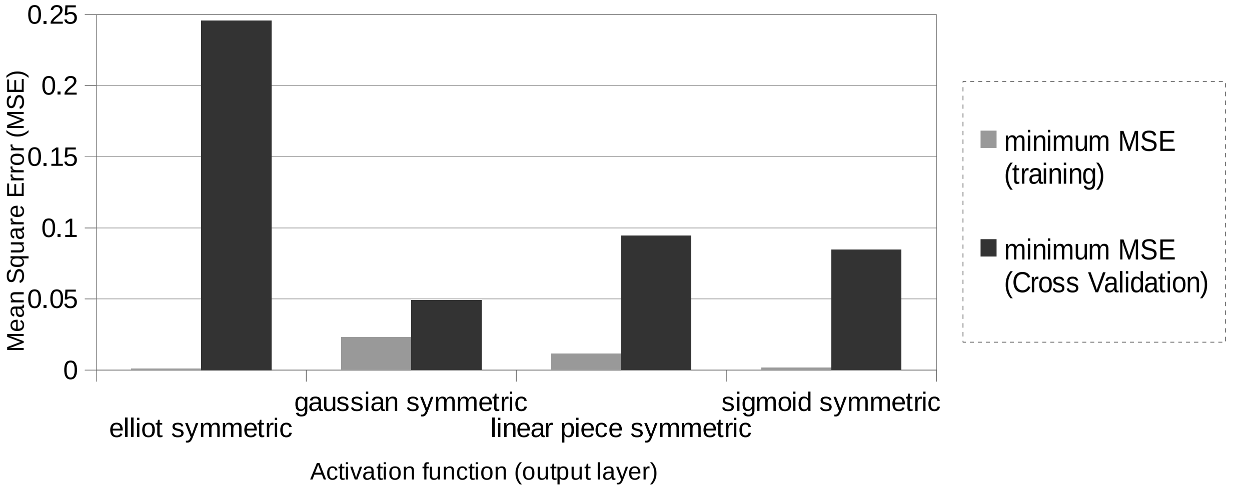


Figure 11: The effectiveness of activation functions is variable when hidden layer is default. *Different functions on output layer will have multiple values of MSE for Cross Validation, while MSE for training is low. Bar graphs represent the MSE.*

When FANN Elliot symmetric is set as the activation function on hidden layer, the effectiveness of the activation functions in question varies. While all of them have low MSE on training, the MSE on cross validation is variable *(Fig. 12)*. In order to determine about the most suitable activation function for output layer the number of the fail bits was examined, indicating the number of output neurons which differ more than the bit fail limit.

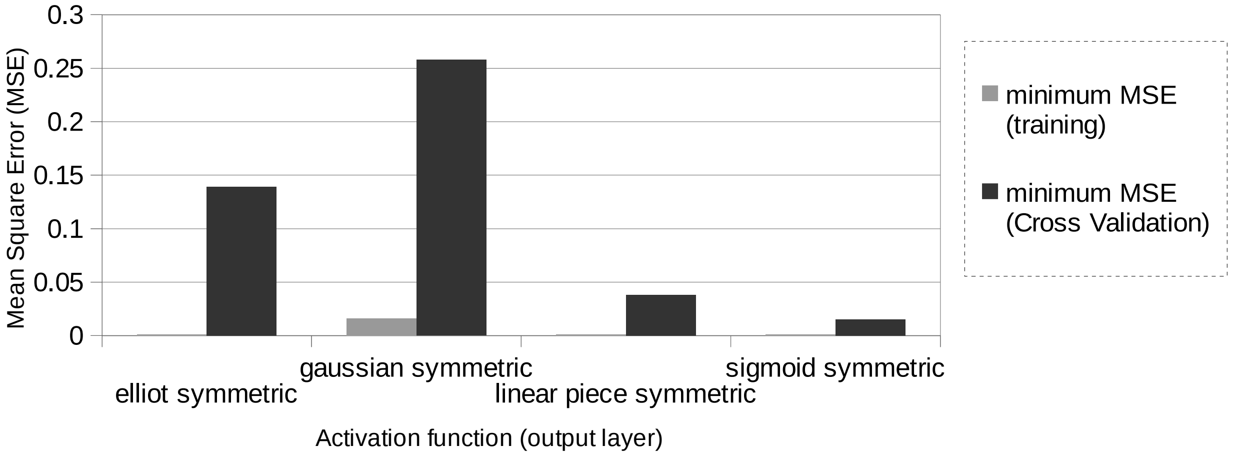


Figure 12: Different activation functions on output layer have different effect on MSE when associated with FANN Elliot symmetric in hidden layer*. While FANN Gaussian symmetric seem to be the least suitable function, all other three have low MSE on training. Bar graphs represent the MSE.*

Both FANN Linear piece and Sigmoid symmetric seem to have a low MSE after training and cross validation, while the fail bit is low, meaning that the training was successful *(Fig. 13)*. FANN Linear piece symmetric is probably more suitable as an activation function on output layer when FANN Elliot symmetric is on hidden layer.

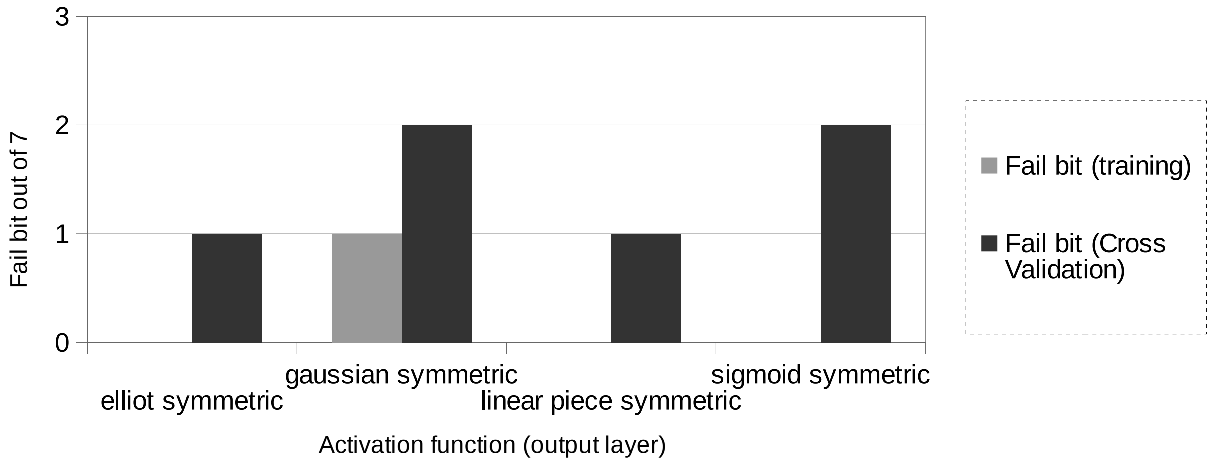


Figure 13: Fail bit represents the training success of the neural network. *Bar graphs represent the fail bit.*

3.1.2 Training Algorithm

In order to determine the most suitable training algorithm for the neural network, the four (4) algorithms used in FANN were tested using 70 data sets for training, FANN Elliot and Linear piece symmetric activation functions for hidden and output layer respectively, and one hidden layer with 7 neurons. The algorithms tested were batch, incremental, QUICKPROP and the default RPROP. While MSE of cross validation does not differ among each trial, the default training algorithm RPROP seem to be the most suitable one where MSE of training is 0.001 and the prediction of correct signs is the highest among the rest. FANN QUICKPROP has the lowest MSE on cross validation but lacks on prediction, while batch algorithm has higher MSE on training and cross validation. Incremental is not a suitable training algorithm due to its high MSE on training and its low prediction of correct signs *(Fig. 14)*.

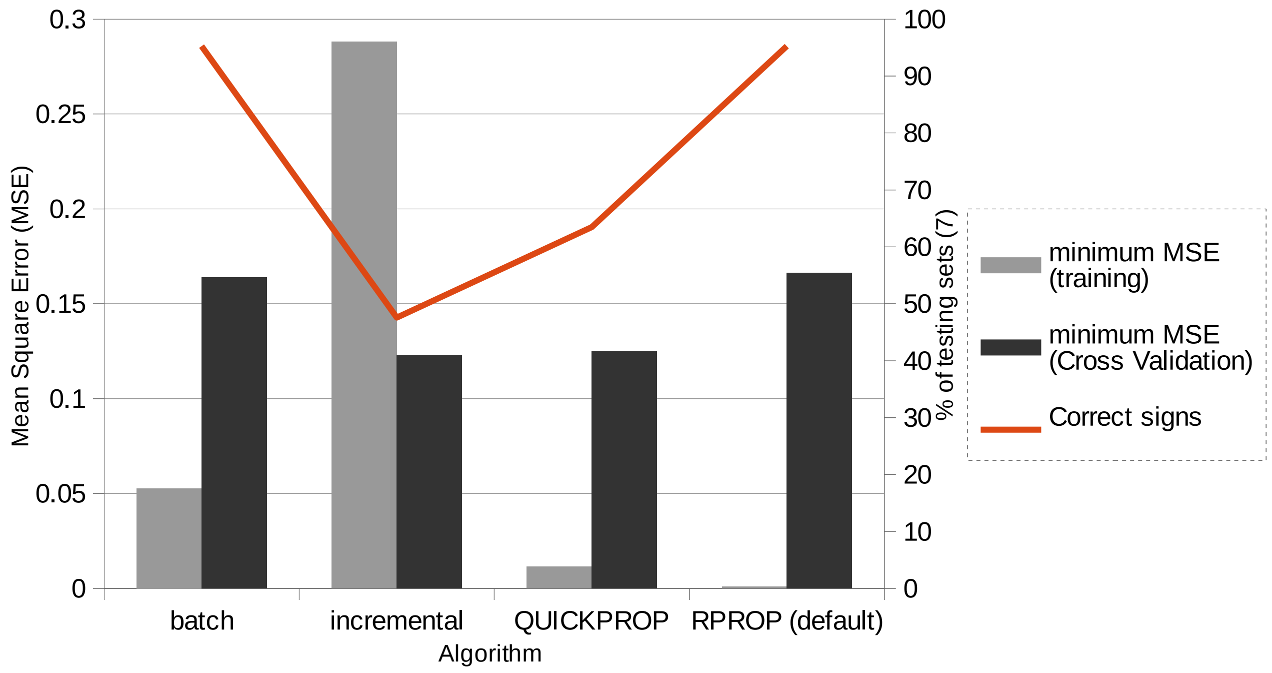


Figure 14: Default training algorithm seems to be the most suitable among the rest. While MSE is low, it manages to predict 85.71% correct signs. *Bar graphs represent the value of MSE. Line represents the percentage of correctly predicted phases.*

3.1.3 Number of training sets

While the training algorithm is determined, the activation function is yet to be decided. In order to examine the appropriate activation function of output layer, multiple numbers of training sets are tested to also assess whether the size of the network can affect the training ability of the neural network. Having set the activation function of the hidden layer FANN Elliot symmetric, one hidden layer of 7 neurons is used to examine 10, 40, 70, 140, 200, 500, 700, 1000, 1400 and 70000 training sets.

Firstly, FANN Sigmoid symmetric was assessed as the activation function of output layer and compared to FANN Linear piece symmetric. Both activation functions were examined for the ability of the neural network to predict correct signs when tested. So far, FANN Linear piece symmetric seems to be a most suitable activation function for output layer when FANN Elliot symmetric is set on hidden layer *(Fig. 15)*.

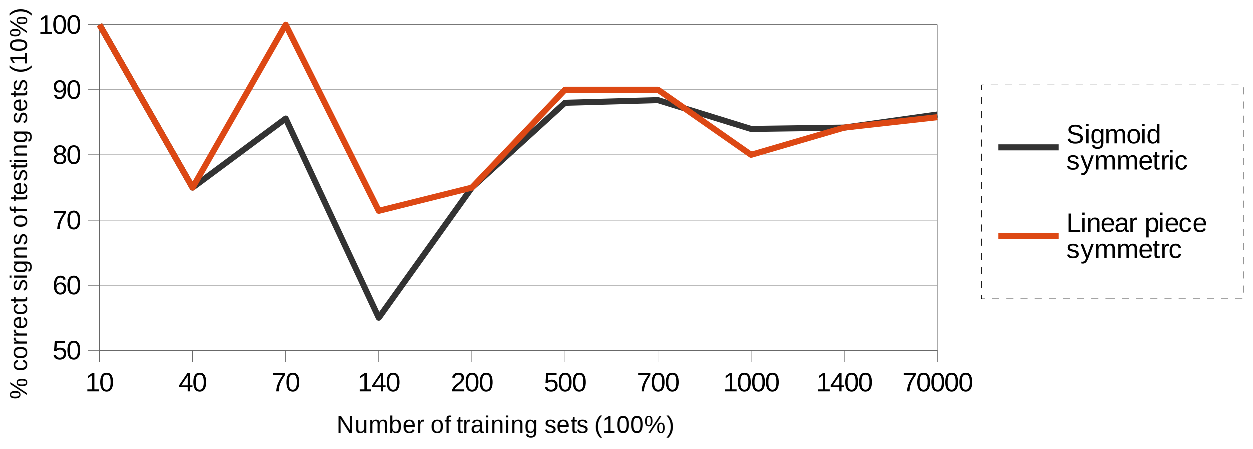


Figure 15: Activation function FANN Linear piece symmetric on output layer seem to be more efficient than FANN Sigmoid symmetric when Elliot symmetric is set for hidden layer. *Lines represent the percentage of correctly predicted phases.*

The number of training sets seem to have an effect on the training of the network. Although a few training sets can train a network to predict 100% correct signs, the small amount (10%) of the testing set on cross validation should be taken into account. For instance, when a neural network is being trained over 10 training sets, the test is taken place on just 1 set, indicating that it could be a matter of chance. On the contrary, a great number of training sets may have a negative effect by over-fitting the network; the network may have memorized the results rather than learn how to predict them.

The number of training sets may be of great importance to the training of the neural network but a suitable number of sets cannot be determined. The graph representing the training and cross validation of multiple neural networks using FANN Linear piece symmetric as the activation function of output layer do not predict a pattern of association between the number of training sets and the successful training of the neural network *(Fig. 16)*.

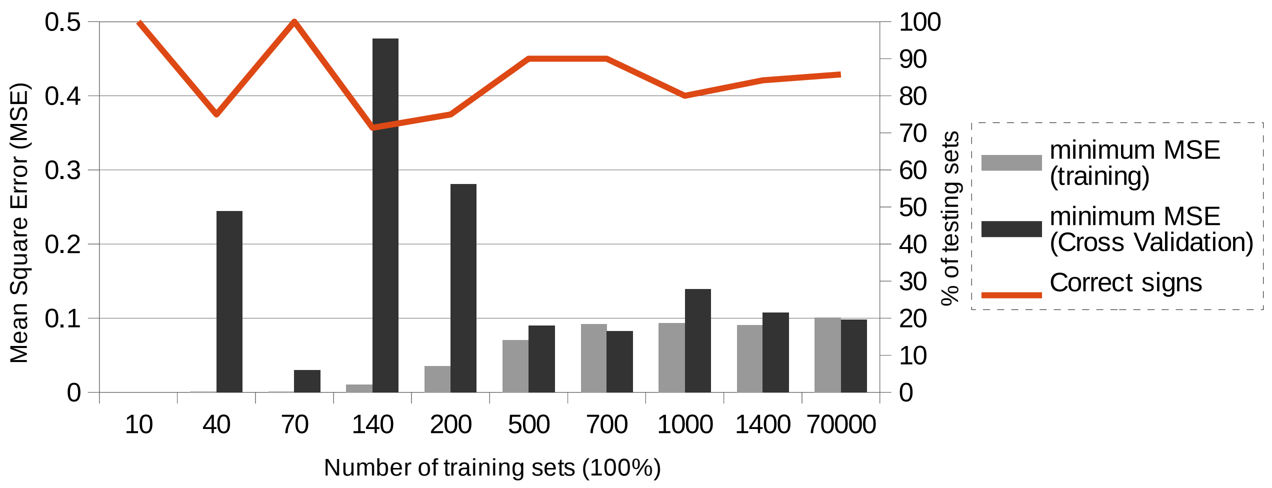


Figure 16: The effectiveness of the number of training sets on the learning rate of the network is variable. *Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases.*

3.1.4 Number of neurons and hidden layers

Having established the parameters of training algorithm and the activation function of both hidden and output layer, the size of the network is to be determined. Different conditions of one or two hidden layers were examined, while altering the number of neurons on each layer. For that reason, training was attempted on neural networks using 70 data sets for one or two hidden layers of 3, 7, 14, 50, 70 and 100 neurons each.

As presented by the graph in *figure 17*, a few neurons or hidden layers do not seem to allow the network process the appropriate operations. In order to determine about the size of the network, the trials were re-conducted for several numbers of training sets using two (2) hidden layers with different number of neurons in each of them. The numbers of sets tested were 200, 700 and 70000 for 3, 7, 14, 50, 70 and 100 neurons in both hidden layers.

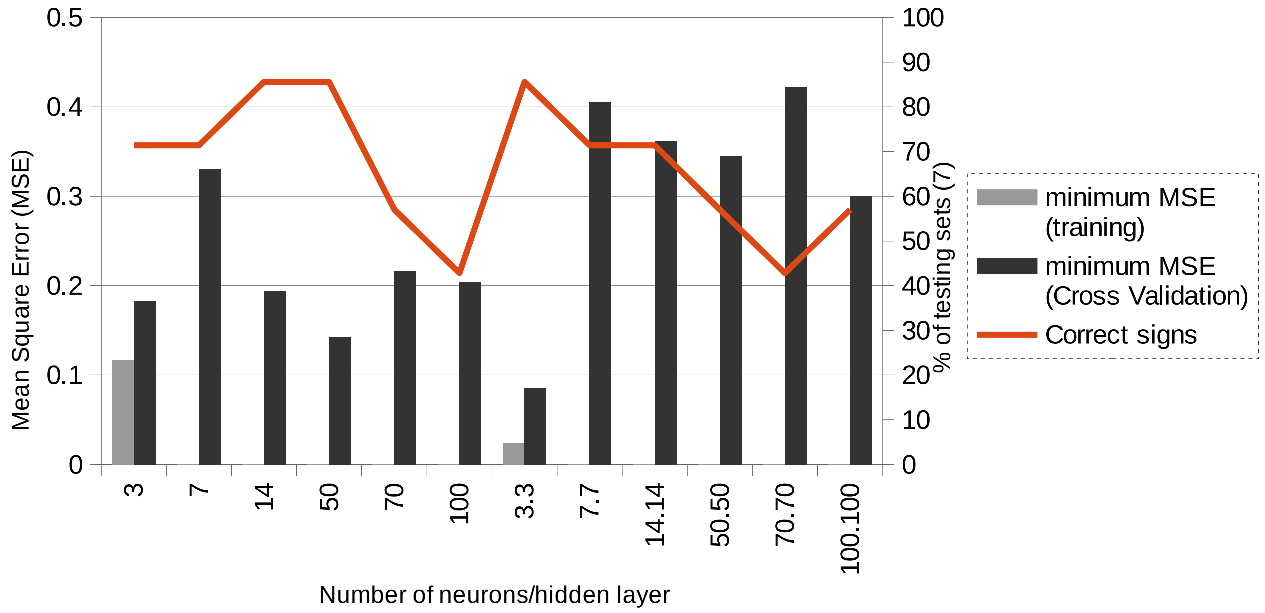


Figure 17: The effectiveness of the number of neurons and hidden layers is variable. *Bargraph represent the MSE. Line represents the percentage of correctly predicted phases.*

Results suggest that both the number of neurons in the hidden layer and the number of data sets correlate with the network's ability to learn how to predict the correct output. The more the number of sets fed into the network, the more complex and numerous the neurons should be in order for the network to be trained *(Fig. 18)*.

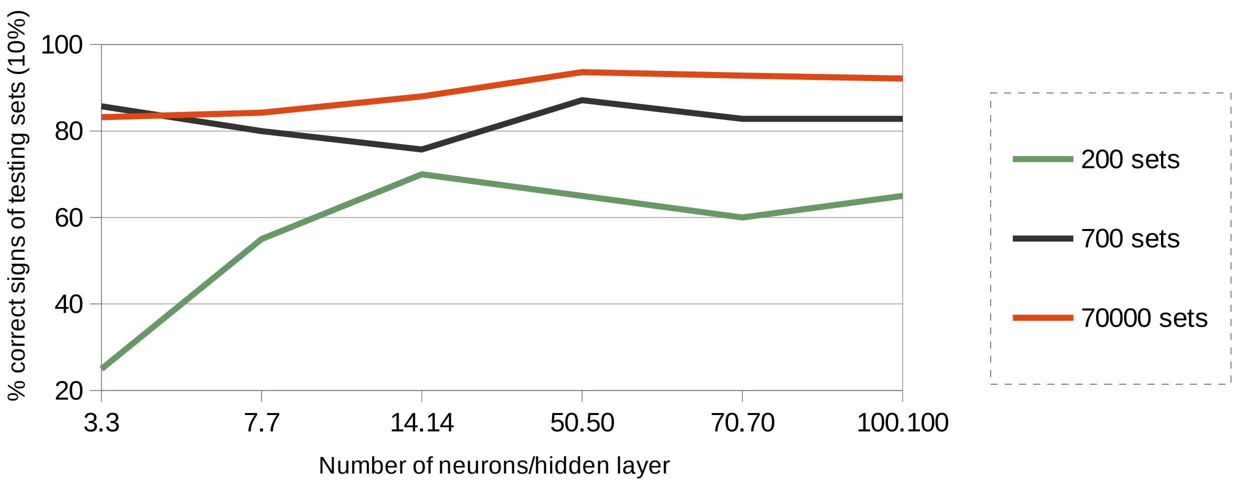


Figure 18: The number of training sets upon which the network is trained is related to the success of the training. *Lines represent the percentage of the correctly predicted phases.*

3.2 Experiments with structure factors of 4 atoms as input data

Having previously established parameters such as the Training algorithm and the Activation function of the hidden and the output layer, a neural network will be trained upon data on bigger structures. Thus, data sets were produced using structure factors of 4 atoms in a 7.0x12.0 Å unit cell. The main target of the experiments was to establish the appropriate number of hidden layers, the neurons in each hidden layer and the number of training sets in order that the network learns the necessary algorithmic operations to predict the correct output, thus the correct phase. Alternative parameters were also examined; networks were tested about their ability to learn while being sparse or dense, when the input data is shuffled during training, when the weights of the input data are initialized using the Widrow and Nguyen's algorithm and lastly it was assessed whether the number of the cycles of the training procedure has an effect on the outcome.

Data sets represent the E-values of the reflections of hypothetical 2-Dimensional structure in centrosymmetric p2 crystals. A minimum fractional separation between the atoms is set at 0.15 while the maximum resolution is 1.0 Å. The data sets contain 126 reflections as input and one output. The output is 1 or -1 representing a positive or negative phase respectively for the reflection $\bar{2}0$. As in previous experiments, the reflection under study was chosen as a structure invariant detected for it is a large structure factor in multiple structures.

For better understanding the effect of the amount of hidden layers and the neurons in each layer, experiments were conducted with multiple trials using input data of 80000 training sets. A simple network was implemented with a default training algorithm and activation functions FANN Elliot symmetric on hidden layer and FANN Linear piece symmetric on output layer. Ten (10) trials of training were conducted on each condition. Results showed that one hidden layer does not allow the network to predict the correct phases on the level that two or more hidden layers do. Moreover, the MSE on both training and cross validation is higher, indicating that more than one hidden layers are expected to produce a neural network much more capable of learning *(Fig. 19)*.

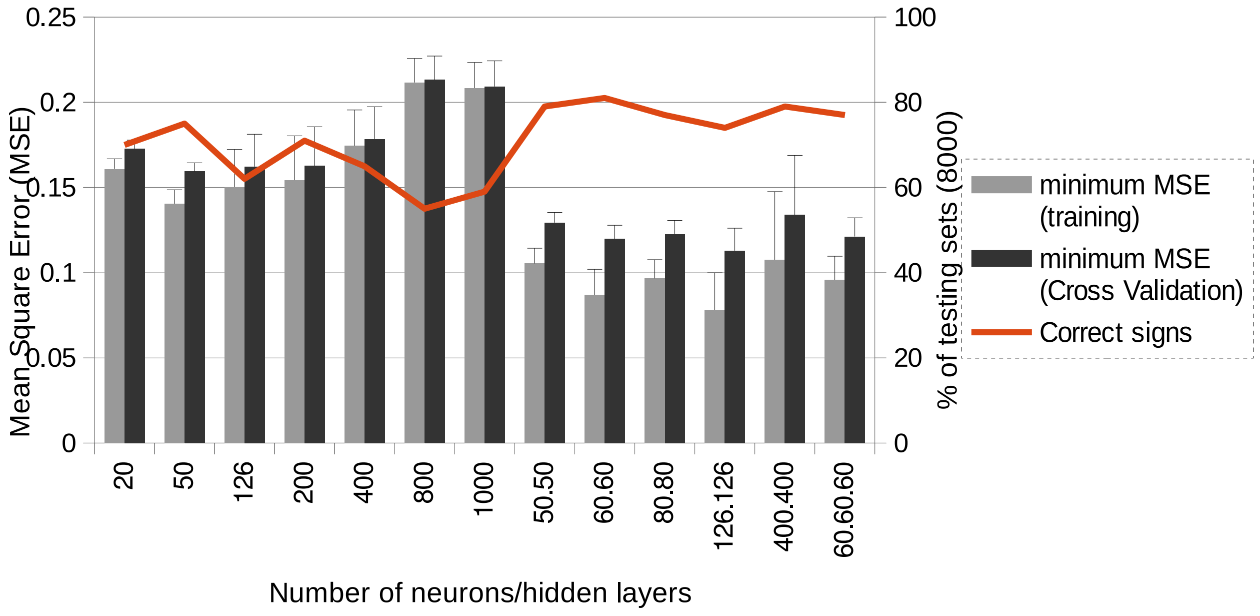


Figure 19: A more complex hidden layer is more capable of depicting the appropriate operations and allow the network to learn. *Two or more hidden layers consist a network that can more accurately predict the correct output. Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases. Error bars represent the standard deviation. Number of replicates per condition=10.*

More experiments were conducted in order to compare the results of the network trained on 80000 training sets with a network trained on 20000 sets. Both networks are able of learning and manage to predict 74-81% and 71-74% correct phases respectively. Less MSE and better prediction of phases is observed when the network is trained with more training sets. However, the range of the percentage of correct prediction is limited in both conditions, suggesting that when two hidden layers are constructed, the number of neurons on each layer has not a great effect on the outcome *(Fig. 20-21)*.

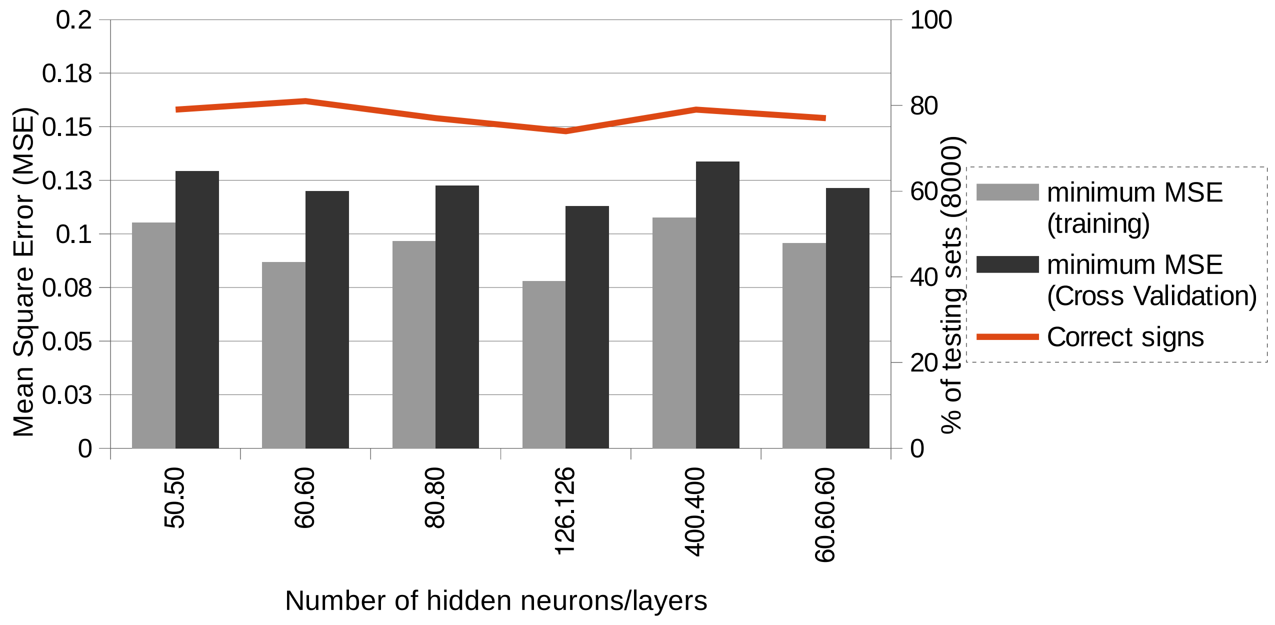


Figure 20: Simple train of neural network upon input data of 80000 training sets. *The percentage of correct prediction of the phases is almost steady. Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases.*

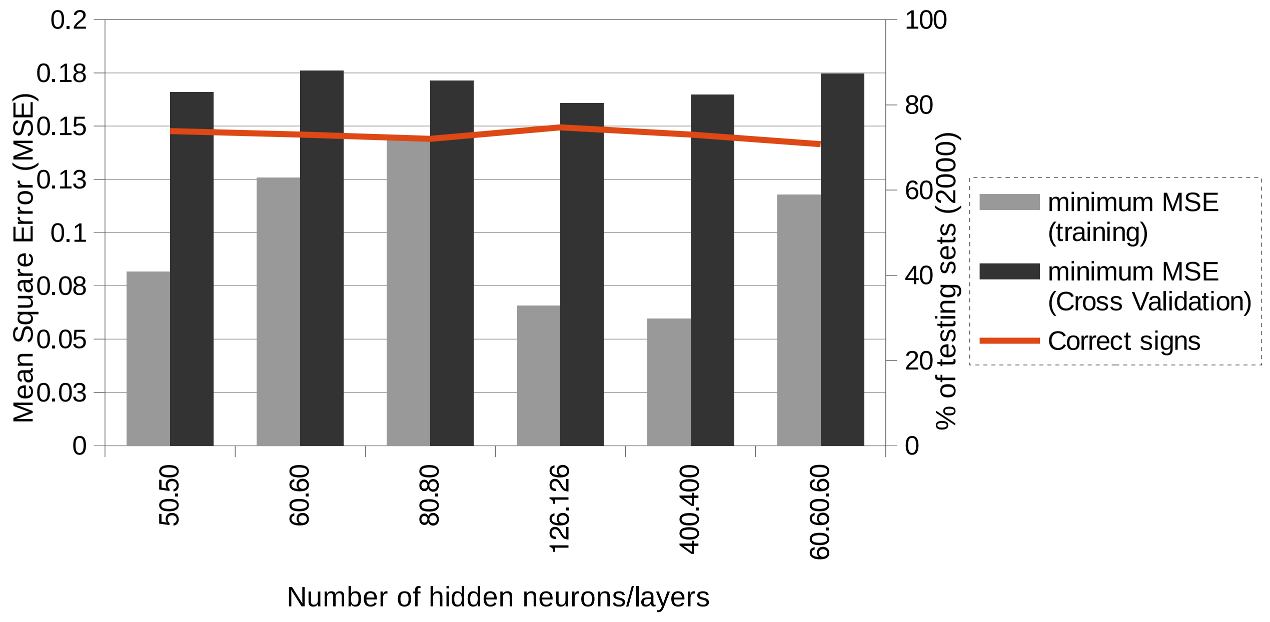


Figure 21: Simple train of neural network upon input data of 20000 training sets. *The percentage of correct prediction of the phases varies on a limited range. Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases.*

3.2.1 Alternative ways to read the training data

Neural networks trained upon input data of 20000 training sets were examined about whether a change on the input format has an effect on learning ability of the network. At first, the input data set was presented to the network as constructed and then shuffled on every cycle of the training procedure. Therefore, the same input training data set was used from the network to be trained but in a different order. Results were compared to the simple training. While the MSE on both training and cross validation does not seem to differ from the simple training, the network's ability to learn was precisely lower than which is presented in figure 21, as shown by the percentage of the prediction of the correct phases after the training *(Fig. 22)*. For that reason, shuffled training sets are not suitable for the network in order that it learns to predict the expected outcome.

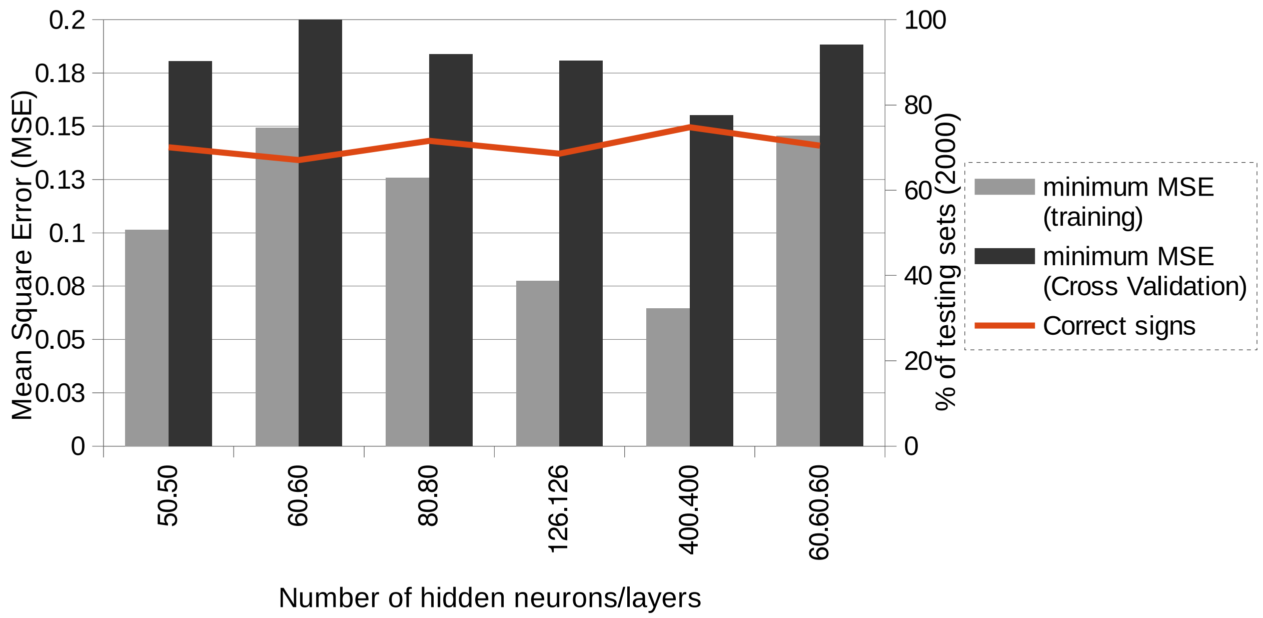


Figure 22: Shuffled training sets do not boost the ability of the network to learn. *Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases.*

3.2.2 Alternative ways to adjust weights

Another process of reading the input data implicates the weights on the input neurons. The weights represent the synapses and are altered on every cycle of training in order to reflect the input-output association. An initialization of those weights were attempted using the algorithm developed by Derrick Nguyen and Bernard Widrow *(1990)*. The algorithm was introduced as a tool that improves the learning speed. After several trials of training the network with initialized weights the MSE of both training and cross validation is slightly less but the percentage of prediction of correct phases is almost the same as when the network is trained without initialization of the weights.

Although not many differences are observed upon initialization, the network seems to perform better than in the original conditions which are shown in figure 21 *(Fig. 23)*. For that reason, the following experiments will be conducted using initialized weights.

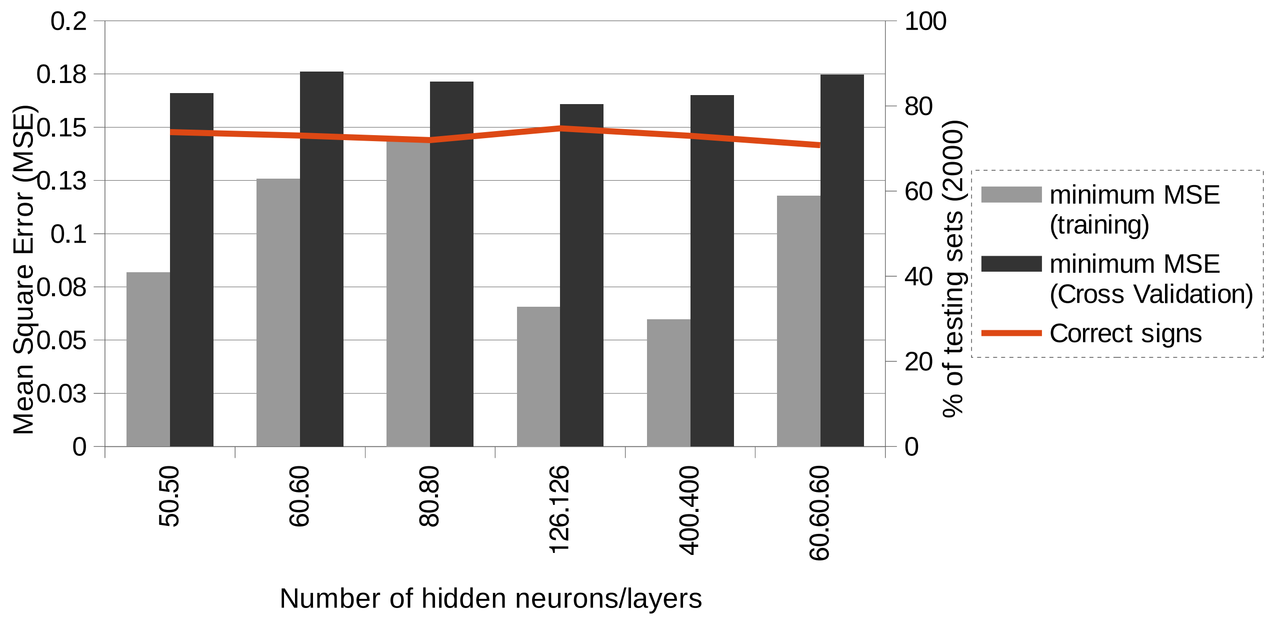


Figure 23: Neural networks seem to perform better in training when using initialized weights according to Nguyen's and Widrow's algorithm. *Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases.*

3.2.3 Sparse networks

Sparse networks are defined as less linked than dense or full high connection networks. The difference between the two is due to the connection rate, a factor among [0,1]. When the factor is 0.5, only half of the possible connections between neurons are taking place in the network, while the factor 1 is equivalent to a simple dense network where all neurons are connected. The purpose of sparse networks is to prevent over-fitting.

Three (3) conditions of sparse networks were examined; when the connection rate were 0.8, 0.5 and 0.25. All three networks were to be compared to the dense network in figures 24-26 using a training input data set of 20000 sets of structure factors.

Results presented below suggest that although the connection rate may have a variable effectiveness on the MSE during training, it does not affect the outcome of the procedure and do not foster the network to predict the phases more accurately. The prediction of the phases is roughly at the same level on every condition *(Fig. 24-25-26)*. Therefore, sparse networks are not considered of benefit to their learning ability.

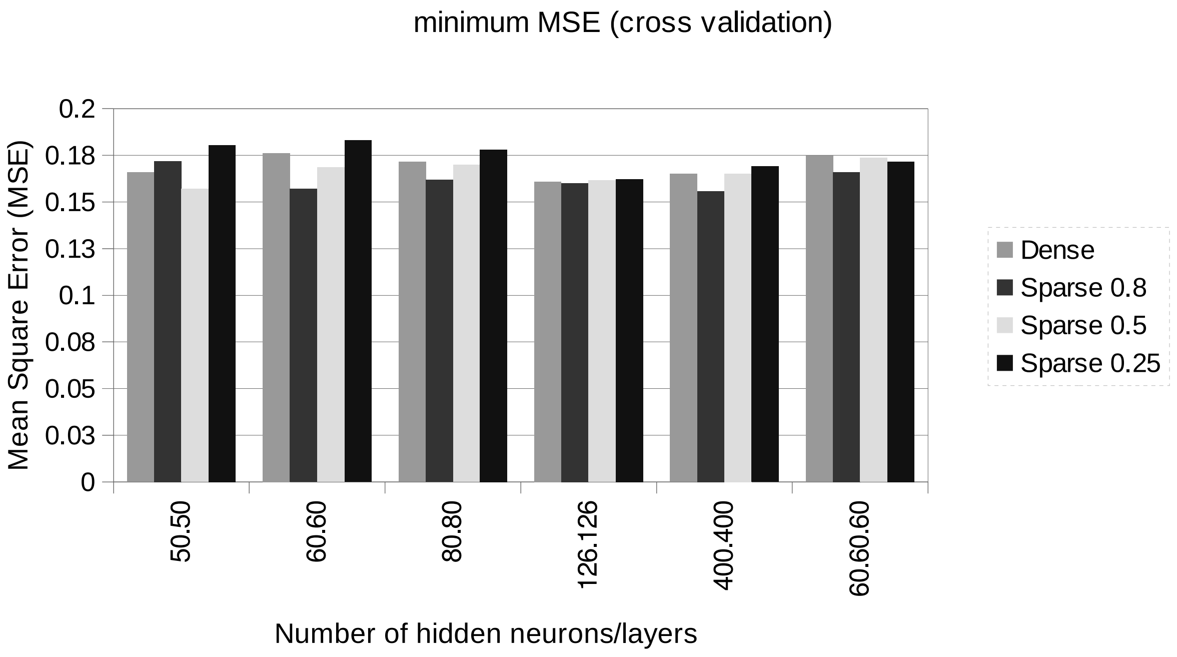


Figure 24: The connection rate has no great effect on cross validation. *Bar graphs represent the MSE.*

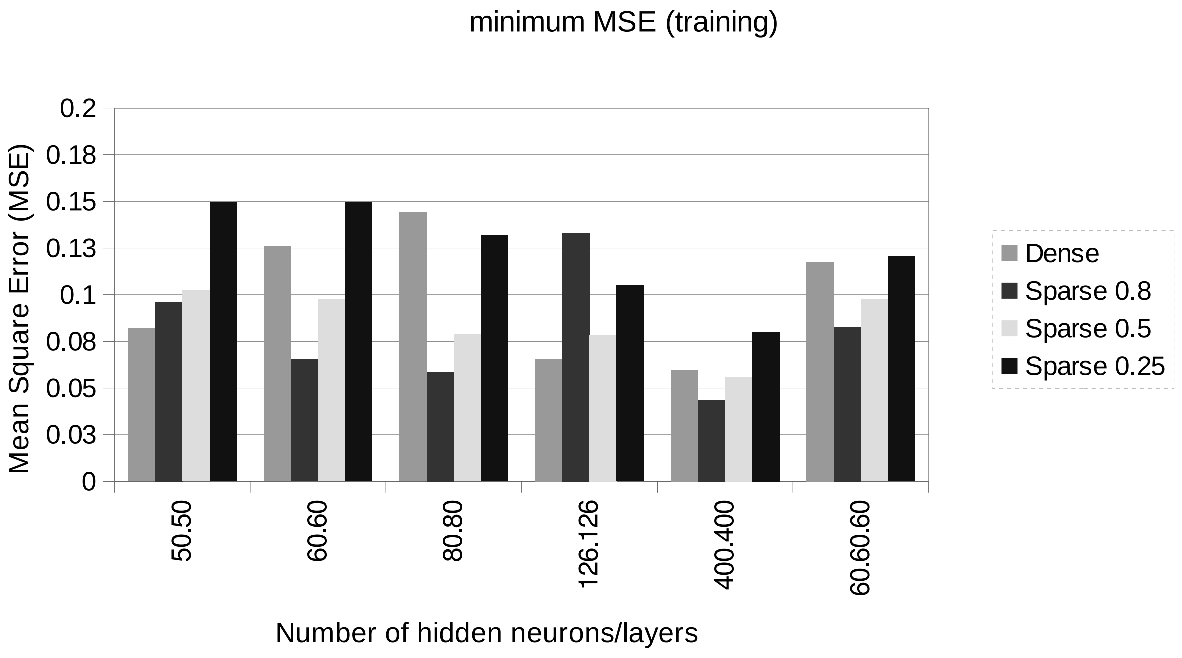


Figure 25: The effectiveness of the connection rate is variable. *Sparse networks with low connection rate do not perform well during training, although sparse networks with 0.8 connection rate compete on the performance of the dense network. Bar graphs represent the MSE.*

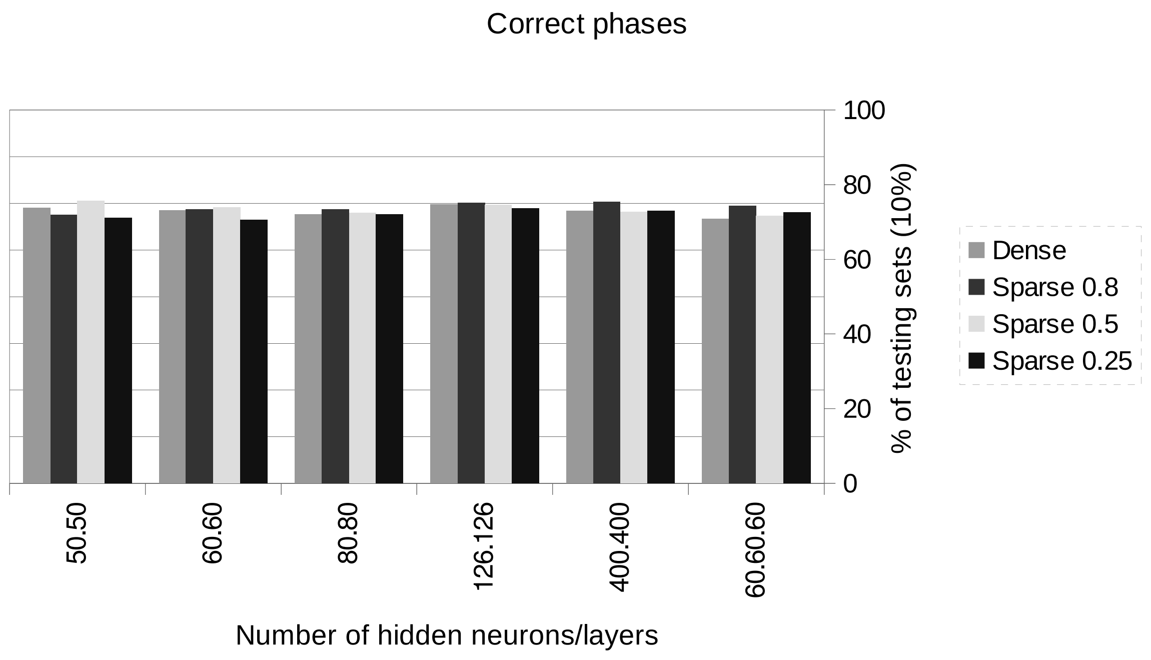


Figure 26: Connection rate has no great effect on the network's ability to predict the correct phase after training. *Bar graphs represent the percentage of correctly predicted phases.*

3.2.4 Attempts on different numbers of neurons and hidden layers

So far, neural networks are structured dense using the default training algorithm, while FANN Elliot symmetric and Linear piece symmetric are considered the appropriate activation functions on hidden and output layer respectively. The network is trained using an input data set of 126 reflections and its weights were initialized using the Nguyen's and Widrow's algorithm. Trials on different numbers of training sets were performed altering every time the number of neurons in the hidden layer in order to establish a relationship between them and the network's learning ability.

Training was attempted using 4000, 40000 and 80000 training sets. While the percentage of phases predicted correctly varies among the conditions, it ranges about 3.2%. On a closer examination, when the network is trained on 4000 training sets, it may predict 60.7-64.5% correct phases. On the other hand, when the network is trained on 80000 training sets, it is able to predict 82-84% correct phases. The results suggest that the number of the training sets fed into the network are of a greater importance than the number of the neurons, although the number of layers may also have a considerable consequence.

In order to more thoroughly assess the influence of the hidden layers on the network's learning ability, more trials were performed with complex hidden layers. Using an input data set of 80000 sets, networks were structured carrying three (3) or four (4) hidden layers of different number of neurons in each of them. The results are presented in the following figure. Both the MSE on training and cross validation seem to not differ among the conditions. More importantly, the percentage of the correct phases predicted do not also alter to a considerable extent. The values fluctuate between 78.9-82.2% suggesting that three or four hidden layers do not have a great effect on network's learning ability *(Fig. 27)*.

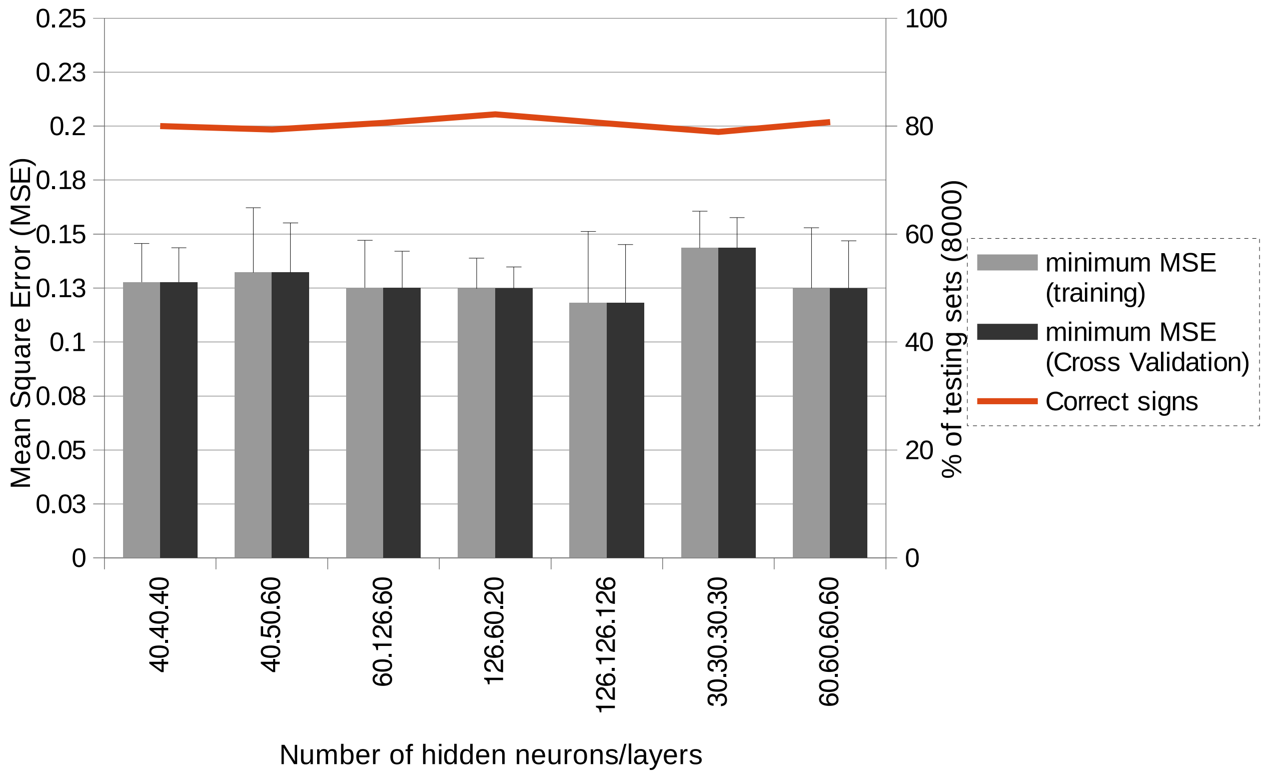
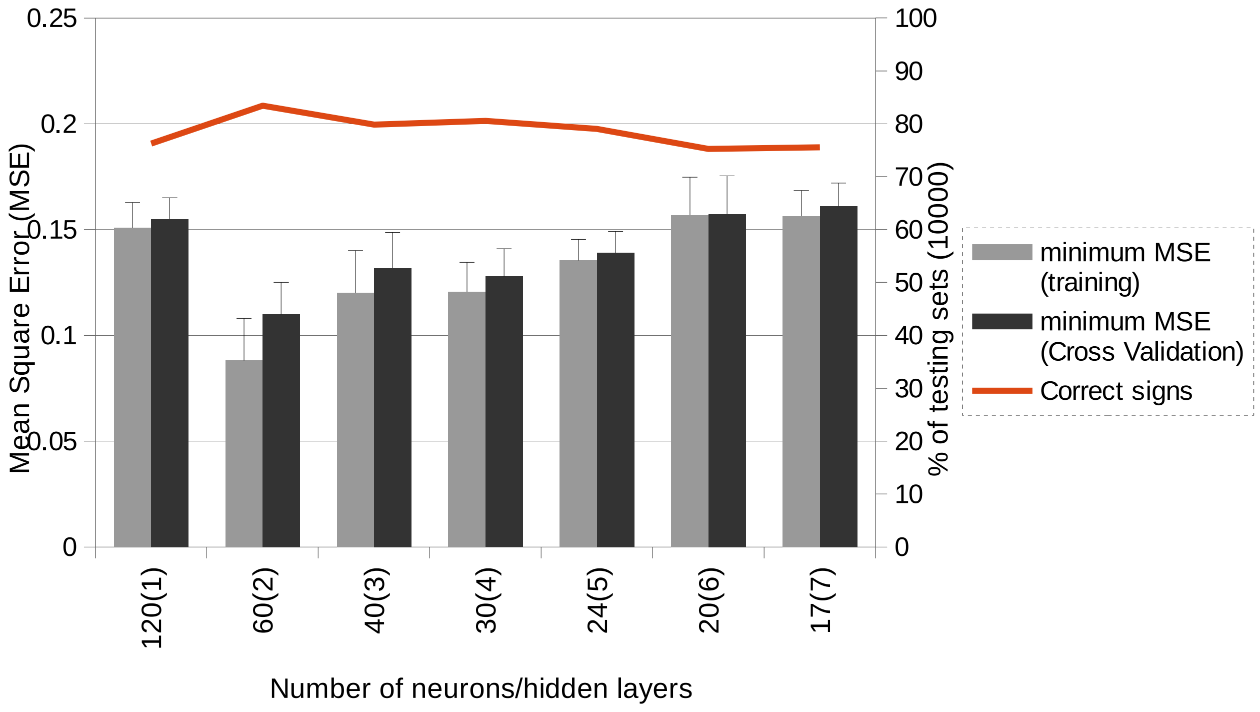


Figure 27: Different numbers of neurons in two (2) or three (3) hidden layers do not alter the performance of the network. *Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases. Error bars represent the standard deviation. Number of replicates per condition=10.*

Up until now there is not an established relationship between the number of hidden layers and the learning ability of the network. To this effect, deeper networks were constructed with one (1) to six (6) and seven (7) hidden layers in order to conclude in a suitable size of the network. A series of trials was performed when 120 neurons in total were divided into 1 to 7 hidden layers. Afterwards, another series of

(A)



(B)

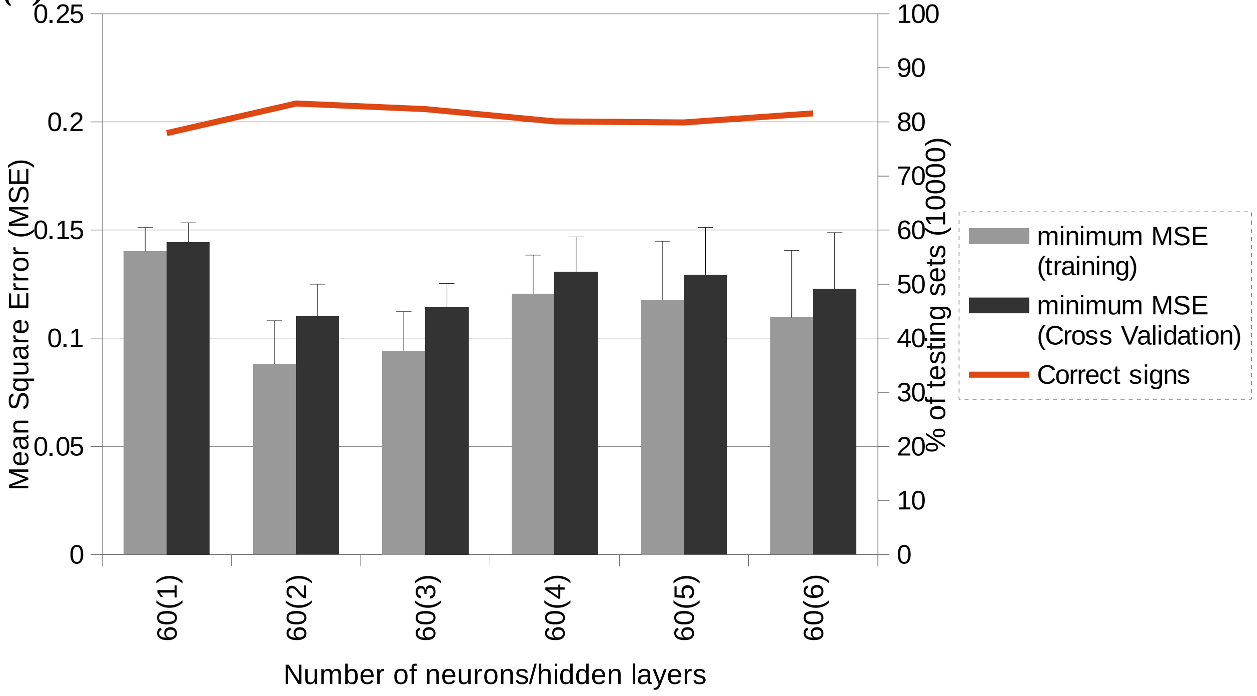


Figure 28: Two (2) hidden layers seem to consist a network with greater performance than of one or more than two hidden layers. *Seven (7) hidden layers with 120 total number of neurons in A. Six (6) hidden layers with 60 neurons in each layer in B. Bar graphs represent the MSE. Line represents the percentage of correctly predicted phases. Error bars represent the standard deviation. Number of replicates per condition=10.*

trials attempted training with 1 to 6 hidden layers having 60 neurons in each layer. All kinds of networks used 100000 training sets as input data set and their weights were initialized using the Nguyen's and Widrow's algorithm. Results confirmed that a single layer does not promote learning. Moreover, both figures suggest that a network with more than two (2) hidden layers gives feedback of a greater MSE on both training and cross validation, while it predicts less correct phases *(Fig. 28)*.

As a conclusion, it is safe to assume that two layers are best for the network to learn the appropriate relationship between the input data set and the output. While not many differences are observed among different number of neurons in the hidden layers, the experiments would be resumed using two hidden layers with 60 neurons in each layer.

3.2.5 Attempts on different numbers of training sets

Already, a relationship between the number of the training sets in the input data and the ability of the network to predict the correct phases after training is suggested. In order to further examine this relationship, training trials were performed. This time the network was the same on all trials; the training algorithm was the FANN library's default RPROP, the activation functions for hidden and output layers were FANN Elliot symmetric and Linear piece symmetric respectively. The dense network carried two hidden layers with 60 neurons in each layer. The weights were initialized before training using the Nguyen's and Widrow's algorithm. Multiple conditions were attempted for the network to be trained. During each attempt, the input training data set was altered by its number of training sets. Each condition was performed ten (10) times.

Different numbers of training sets were fed into identical networks. Those numbers ranged between 4000 and 1500000. Results suggested a great impact on the network. Although the MSE of training is slightly higher when the training sets are more, the MSE of cross validation is lower. Most especially, a relationship between the number of training sets and the prediction of correct phases after training is proven; the more the training sets introduced to the network, the higher percentage of phases is accurately predicted upon training. However, the probability of bias is also increased for most of the possible structures have already been introduced to the network during training process *(Fig. 29)*.

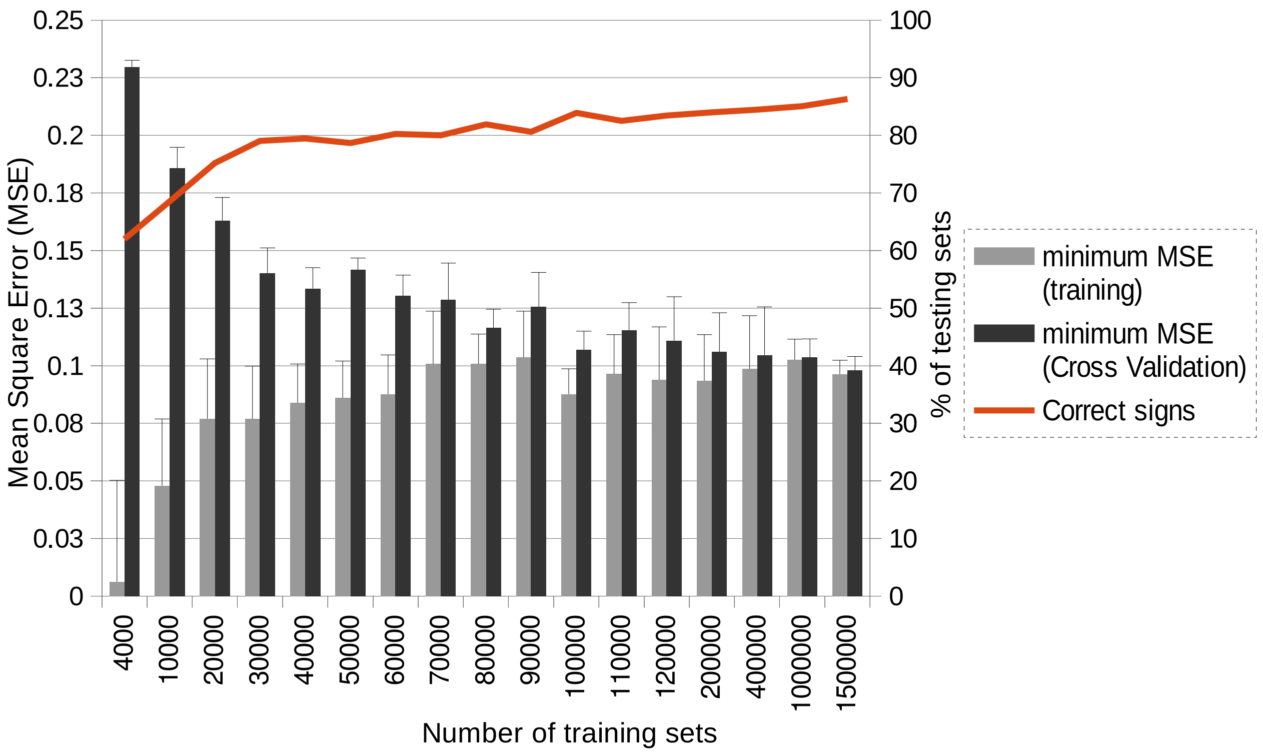


***Figure 29: The number of training sets directly relates to the network's ability to predict the correct phases upon training.** Bar graphs represent the MSE. Line represent the percentage of the correctly predicted signs. Error bars represent the standard deviation. Number of replicates per condition=10.*

On closer inspection, while a network trained on 4000 training sets may predict 62% phases correctly, the percentage rises and finally reaches 86.33% when the network is trained on 1500000 training sets. The cross validation process was monitored for this network and presented in figure 30. The graph represents the percentage of correctly predicted phases during the training process. Data was selected every epoch. The more the epochs of training, the more accurate the network is at predicting the phases *(Fig. 30)*.

Despite that the relationship between the training sets and the observed percentage is established, an equation cannot be determined for the association to be expressed mathematically. Additional trials should be conducted in order to define this relationship.

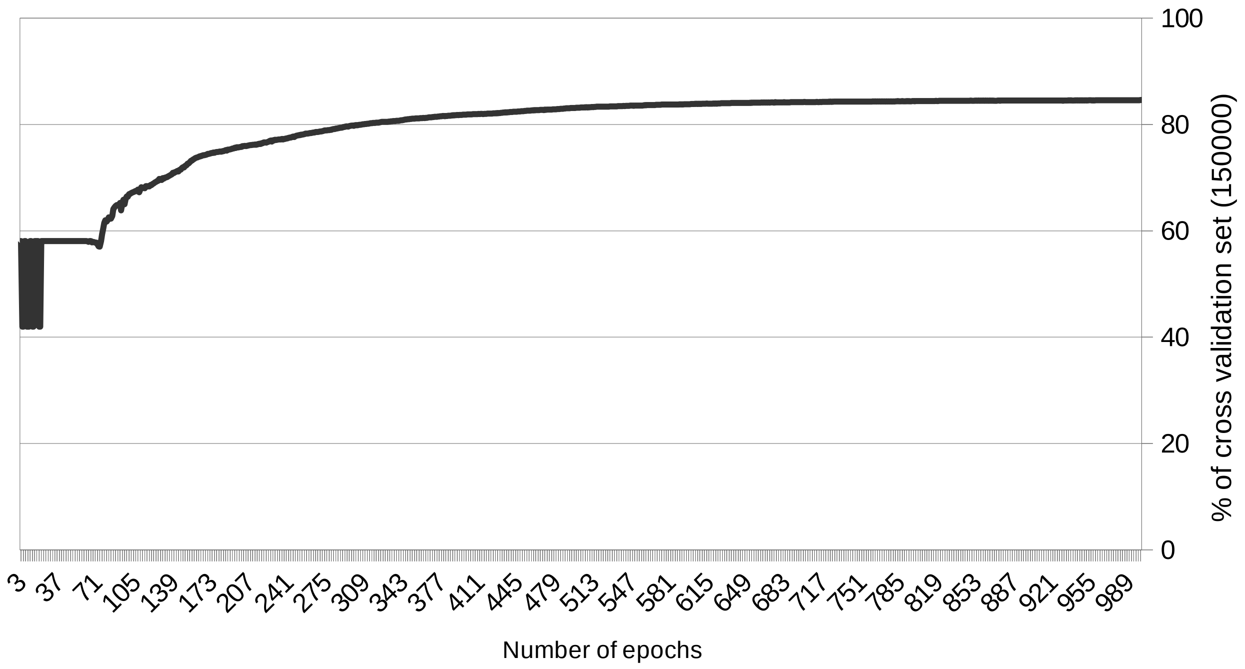


Figure 30: The percentage of correctly predicted phases is constantly rising during training.

3.2.6 Finding the maximum number of epochs needed

While the suitable parameters have yet been established, another constant to be determined is the maximum number of cycles set for the training. The cycles, also called epochs, are the number of times that the network is being trained by reading the input data set and altering the neurons' weights. Wrongfully, it could be suggested that the more the cycles the better the results. However, this is not the truth. A great number of training epochs could lead to over-fitting. For every epoch the network re-reads the training set, the result could be an outcome of memorization rather than learning.

A neural network was constructed with default training algorithm and activation functions FANN Elliot symmetric and Linear piece symmetric for hidden and output layers respectively. The dense network was carrying two hidden layers with 60 neurons in each layer. The input data set had 100000 training sets and the weights were initialized using the Nguyen's and Widrow's algorithm.

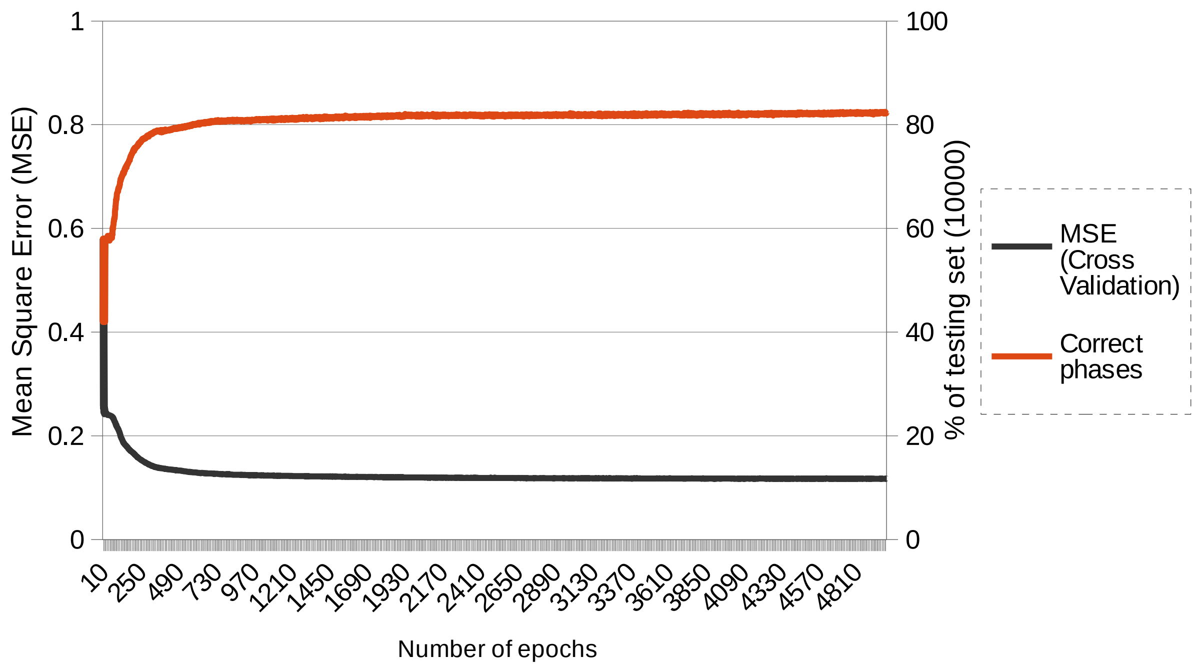


Figure 31: After about 3200 epochs the training process reaches its minimum MSE on cross validation. *Meanwhile, the percentage of correctly predicted phases is constantly increasing, though it may occur as a result of over-fitting.*

The network was trained for 5000 epochs. Feedback was received every epoch about the percentage of correct prediction of the phases and the MSE of the cross validation data. The observed success of the network on learning and predicting the correct output is depicted by a positive trend line. The trend line is constantly rising up until 4992 training epochs. The miminum MSE is reached at 4866 epochs, suggesting that over-fitting has not occurred until then, since over-fitted networks are characterized by a non-decreasing MSE. However, the alterations occurred on the MSE after the epoch numbered 3204 are limited to the fourth decimal digit meaning no significant progress on the network's learning process *(Fig. 31)*.

3.3 Testing the network on alternative structures

Network trained on structure factors of 4 atoms was evaluated on its efficacy to predict the correct phase on newly introduced data. The trials had three aims: to assess the network's ability to predict the output on fresh data with structure factors of 4 atoms and alternatively its accuracy on more and less than 4 atoms. Input data sets were tested with structure factors of 3, 4 and 5 atoms all of which structures contain the reflection $\bar{2}0$ for which the network was trained.

All three data sets represent the E-values of the reflections of hypothetical 2-Dimensional structures in centrosymmetric p2 crystals. Cell was 7.0x12.0 Å and a minimum fractional separation between the atoms is set at 0.15. The maximum resolution was 1.0 Å. The data sets contain 126 reflections as input and one output. The output is 1 or -1 representing a positive or negative phase respectively for the reflection $\bar{2}0$. Changes occurred only on the number of the atoms which consist each structure. Data sets with structure factors of 3, 4 and 5 atoms had 3000, 4000 and 5000 training patterns respectively.

The network of interest was previously trained of 1500000 training sets of 126 reflections as input and one output for the reflection $\bar{2}0$. The network had two (2) hidden layers with 60 neurons in each layer. The training algorithm was default while the weights had been initialized using the Nguyen-Widrow's algorithm. The activation functions on hidden and output layer were FANN Elliot symmetric and Linear piece symmetric respectively. The network was trained for 1000 epochs using the 90% of the data set and has been evaluated on the 10% of the same set. The trained network correctly predicted 86.33% of the phases on the test set (10% of the data set).

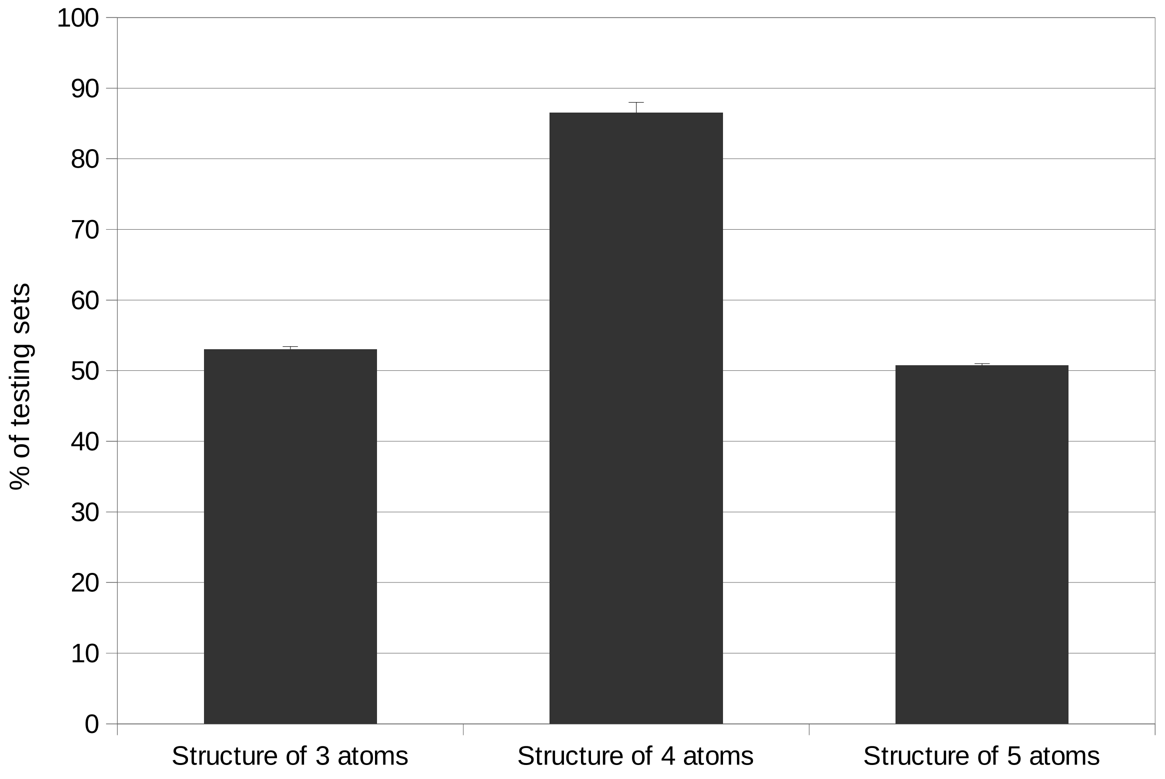


Figure 32: A neural network trained upon structure factors of 4 atoms can successfully predict the phases of new structure factors of 4 atoms, but not of less or more atoms. *Bar graphs represent the percentage of correctly predicted phases. Error bars represent the standard deviation. Number of replicates per condition=2.*

The network was tested about its accuracy to predict the correct phases on newly presented data sets. After two trials of test, it successfully predicted up to 86.5% correct phases of structure factors of 4 atoms. However, the network was not able to predict the output of training sets with structure factors of 3 or 5 atoms. The results were almost 50% correct prediction (53.03% for structure factors of 3 atoms and 50.73% for structure factors of 5 atoms), indicating that the network was not able to correctly predict the output of the test *(Fig. 32)*.

Chapter 4

Discussion

The current thesis dealt with a fundamental obstacle in crystallographic techniques, the phase problem, and assesses the ways in which an artificial neural network (ANN) may be trained in order to solve this. The objective of the research presented in this thesis was to examine whether an ANN can be sufficiently trained to achieve this goal, and if so, to investigate the appropriate implementations of it. The idea is that artificial neural networks, as biological ones do, are capable of training by means of observed, experimental data, a property similar to what it is suggested by direct methods. Direct methods have been developed in order to establish an association between the amplitudes and the phases of the structure factors of a defined crystal. This relationship refers to an algorithmic equation that leads to the correct phases when only the amplitudes are known. For that reason, an artificial neural network may adopt the ways of the direct methods and therefore answer a fundamental question: which are the phases of the structure factors of which only the amplitudes are observed?

To further investigate this suggestion, simple hypothetical 2-dimensional, centrosymmetric structures were employed for the networks to be trained. Structure factors of each crystal were transformed into E-values which, together with the phase of a structure invariant, constituted a training set. The phase was 1 (0 rad) or -1 (π rad) due to the crystal's symmetry. Multiple training patterns formed the training data. ANNs of FANN library implemented in several ways were trained by the method of supervised learning when introduced to the training data for multiple epochs. Every epoch a feedback was received regarding the error value, indicating the difference between the desired and the produced output value. The error value forced the weights of the network to alter by means of minimizing the error. The training procedure needed about a couple of minutes or days to be completed, depending on the size of the network or the information used. Upon completion of the training process, the network was evaluated for its performance. A successful network could predict the signs of the output values corresponding to similar structure factors.

Experiments began with simple multi-layered networks. Structure factors of crystals of 2 atoms were employed and only 7 reflection, meaning only 7 input values, were produced. Despite the difficulties for identifying the suitable size of the network and the training data, the activation function on both hidden and output layer, and the training algorithm used were defined. These two parameters were decided and not altered throughout the following experiments.

Since structures of 2 atoms are too small, the structure factors produced by pepinsky's machine having the defined properties are limited in number. Training data of too many training sets may contain replicates of the same structure factors. Memorization could easily occur if all possible training sets are introduced into the network during training. Thus, the networks would lack the ability of generalization. In order to avoid that but also retain the problem simplified, structures of 3 atoms were skipped and therefore structures of 4 atoms were used for training the networks.

An attempt to define the appropriate size of the network resulted in the suggestion that networks with two hidden layers were more capable of coping with the complexity of the problem and therefore perform better. However, a comparison between four-layered networks trained on 20000 and 80000 training sets showed that differences on the number of the neurons are not as important as the number of the hidden layers which is more likely to have an impact on the training efficacy. Yet, it is noteworthy that the number of the training sets in the training data relates to the network's performance. For the relation to be determined, the number of training sets required for sufficient training was examined. Results indicate that the number of the training sets was directly related to the performance of the network. The best performance noticed was from the network trained with the most training sets, while its generalization ability was retained. This could be easily examined by assessing the slope of the error value during cross validation. Results indicated that no network in this study suffered from over-fitting.

Alternative implementations to enhance training were performed. Most of them showed no significant improvement, such as shuffling the sets which consist the training data every epoch or adjusting the connectivity of the network in different rates. Success was provided from a network when the algorithm suggested by Derrick Nguyen and Bernard Widrow was used. As explained before, this initialization method is proved to promote the error of the training minimize faster *(Nguyen & Widrow, 1990)*. In agreement with that belief, evaluation of the trained networks resulted in better performance. For that reason, this initialization method was used for the experiments.

Overall, the experiments of this thesis had two aims; firstly, to determine whether a neural network could be trained sufficiently by means of observed intensities and therefore predict correct phases of non-previously introduced structure factors. If the case is such, the network inspired by direct methods could solve the crystallographic phase problem and greatly enhance the crystallographer's work. A second aim was to decide about the desired implementations that promote a network's learning in order that it achieves the defined goal. While the latter aim was examined during all experiments, the first one was accomplished at last when the evaluation of the network occurred. The network that performed best of all was the one trained on 1500000 training sets. This network was tested for its performance and it produced 86.3% correct outputs for new structure factors. This percentage, though it is not perfectly reaching 100%, is desired. Sayre himself introduced his methods by stating that the technique would give larger number of correct phases than any other technique, but still not all of them *(Sayre, 1952)*. For that reason, the network is considered successful since it can produce the desired values in a great extent.

However, when a great number of sets are introduced to the network, the probability of bias is also increased since all possible structure factors have probably been used. In order to confirm the success of the network, the training sets should be examined in order to avoid repeatability of structures. An additional disadvantage is that it cannot predict the phases of structures of other numbers of atoms. Experiments showed that when the network, which is trained on structure factors of 4 atoms, is tested on structure factors of crystals of less or more than 4 atoms, the correctly predicted phases are only the half, meaning it is mostly a matter of chance rather than relationships are learned. This fact means that if neural networks are to be used as a tool for crystallographers, they should be firstly trained on structures of all the possible numbers of atoms and then the networks would be combined.

Another thing that may be considered an impediment is the fact that only one phase of a structure invariant was employed. This could be solved thanks to the ability of the networks to be combined, as mentioned before. This means that multiple networks could be trained on the same reflections of the same structure factors as input values but different phase as the output value. In the last experiments that employed structures of 4 atoms, only the phase of the reflection 20 of the crystal was used as an output. Similarly, the phase of the reflection $\bar{2}0$ could also be assessed and also other reflections. In this case, 126 networks should be combined, since there are 126 reflections. Though the training of such an amount of networks seems to be an enduring procedure, this simplified problem could set up the fundamental implementations of the network in order that it achieves the specific goal and, as a consequence, it could be easier to train one network using training data with a number of training sets and all their corresponding phases.

In summary, the networks seem to be able to learn the relationships between the normalized values of the amplitudes and the phases of structure factors. Although this is the case, several trials should be conducted to determine additional parameters such as the learning rate that could be adjusted when using the QUICKPROP training algorithm and whether the normalized values representing the amplitudes of the structure factors in the training data should be unitary or in any other form. Also, a relationship between the number of the training sets and the performance of the network could be determined. Other suggestions of further research on the matter is the use of different types of learning, such as unsupervised training, and the use of other activation functions while setting the output values at a different range.

Conclusively, the phase problem still remains a challenge for researchers. Although the great advances in this field, existing techniques have limitations that severely imped the work of crystallographers and thus the development in biology, a field that the knowledge structures of proteins, DNA and RNA molecules are of great importance. Therefore, an efficient phase determination technique would enhance the progress not only on the field of crystallography, but also on other sciences such as molecular biology.

Chapter X
Literature

1. Als-Nielsen Jens, McMorrow Des (2011) *Elements of Modern X-ray Physics* 2Nd ed., John Wiley & Sons, Ltd, United Kingdom, doi: 10.1002/9781119998365

2. Bradl Monika and Lassmann Hans (2009) *Oligodendrocytes: biology and pathology.* Acta Neuropathol. 119(1):37-53. doi:10.1007/s00401-009-0601-5

3. Cajal S.R. (1894) *La fine structure des centres nerveux*. Proc. R. Soc. Lond. 55: 444–468

4. Giacovazzo Carmelo (2006) *International Tables for Crystallography*, Vol. B, Chapter 2.2, pp. 210-234, Kluwer Academic Publishers, ISBN 0-7923-6592-5

5. Giacovazzo Carmelo (2013) *Phasing in crystallography: a modern perspective*, Rend. Fis. Acc. Lincei, DOI 10.1007/s12210-012-0209-x

6. Chatzidimitriou Dimitrios (2015) *Approaching the phase problem of X-Ray crystallography with feed forward neural networks*, [Master thesis], Democritus University of Thrace, School of Health Sciences, Department of Molecular Biology and Genetics

7. Cochran W. (1952) *A Relation between the Signs of Structure Factors*, Acta Crystallographhica Vol. 5, pp. 65

8. Cohen, N. J., & Squire, L. R. (1980) *Preserved learning and retention of pattern analyzing skill in amnesia: Dissociation of knowing how and knowing that*. Science, 210, 207-210.

9. Drenth Jan (2007) *Principles of Protein X-Ray Crystallography*, 3rd ed., Springer-Verlag New York, DOI 10.1007/0-387-33746-6

10. Eluyode O.S. et al. (2013) *Comparative study of biological and artificial neural networks.* Euro. J. Appl. Eng. Sci. Res., 2(1):36-46

11. Gereshenson Carlos (2003) *Artificial Neural Networks for Beginners*, [e-tutorial] arXiv:cs/0308031v1

12. Gerrow K., Triller A. (2010) *Synaptic stability and plasticity in a floating world,* Current Opinion in Neurobiology Vol. 20 Ch. 5, pp. 631-639

13. Hagan T. Martin, Demuth B. Howard et al. (2014) *Neural Network Design* 2nd ed., Martin Hagan, ISBN 0971732116, 9780971732117

14. Hebb, D.O. (1949) *The Organization of Behavior,* New York, Wiley & Sons, ISBN 978-

0805843002

15. Hauptman Herbert (1983) *The phase problem of x-ray crystallography*, Proc. Indian Acad. Sci. (Chem. Sci.), Vol. 92, Numbers 4 & 5, pp. 291-321
16. Hauptman Herbert (1997) *Phasing methods for protein crystallography*, Structural Biology, Vol. 7, pp. 672-680
17. Herculano-Houel Suzana (2009) *The Human Brain in Numbers: A Linearly Scaled-up Primate Brain*, Front/ Hum. Neurosci. Vol. 3, p. 31, doi: 10.3389/neuro.09.031.2009
18. Islam Md. Monirul, Murase K. (2001) *A new algorithm to design compact two-hidden-layer artificial neural networks*, Neural Networks Vol. 14, pp. 1265-1278
19. Jain K. Anil, Jianchang Mao, K. Mohiuddin (1996). *Artificial Neural Networks: A tutorial*, IEEE Computer, pp. 31-44
20. Kandel R. Eric (2000) *The Molecular Biology of Memory Storage: A dialog between genes and synapses*, Nobel Lecture, pp. 392-439
21. Kandel R. Eric, Schwartz H. James (1982) *Molecular Biology of Learning: Modulation of Transmitter Release.* Science, Vol. 218, pp. 433-443
22. Kandel R. Eric, Schwartz H. James, Siegelbaum A. Steven, Hudspeth A.J, (Eds) (2013) *Principles of Neural Science* 5th edition, McGraw-Hill Companies, United States of America, ISBN 978-0-07-139011-8
23. Kihlstrom F. John, Dorfman Jennifer, Park Lillian (2007) *Implicit and Explicit Memory and Learning*, M. Velmans & S. Schneider (Eds.), The Blackwell Companion to Consciousness, Oxford, United Kingdom
24. Kimelberg K. Harold and Nedergaard Maiken (2010) *Functions of astrocytes and their potential as therapeutic targets.* Neurotherapeutics. 7(4): 338–353. doi:10.1016/j.nurt.2010.07.006
25. Kowaliw Taras, Bredeche Nicolas, Doursat René (Eds.) (2014) *Growing Adaptive Machines*, Studies in Computational Intelligence Vol. 557, Springer Berlin Heidelberg, doi:10.1007/978-3-642-55337-0
26. Kriesel David, (2007) *A Brief Introduction to Neural Networks*, available at http://www.dkriesel.com
27. Kröse Ben, and Patrick van der Smagt (1996) *An Introduction to Neural Networks.* University of Amsterdam.
28. Ladd Mark, Palmer Rex (1980) *Theory and Practice of Direct Methods in Ctystallography*, Springer US, DOI: 10.1007/ 978-1-4613-2979-4

29. Ladd Mark, Palmer Rex (2013) *Structure Determination by X-ray Crystallography* 5th ed., Springer US, DOI 10.1007/978-1-4614-3954-7

30. Lewis Andrew, Mostaghim Sanaz, and Randall Marcus (Eds.) (2009) *Biologically-Inspired Optimisation Methods*, Studies in Computational Intelligence Vol. 210, Springer-Verlag Berlin Heidelberg, doi:10.1007/978-3-642-01262-4

31. Lodish H., Berk A., Zipursky S. Lawrence, Matsudaira P., Baltimore D., and Darnell J. (2000) *Molecular Cell Biology, 4th edition.* W. H. Freeman. New York. ISBN-10:0-7167-3136-3

32. Matheus J. Christopher and Hohensee E. William (1987) *Learning in artificial neural systems*, Comput. Intell. Vol. 3, pp. 283-294

33. McLeod, S. A. (2012) *Working Memory,* retrieved from www.simplypsychology.org/working %20memory.html

34. Mitchel M. Tom (2005) *Overfitting, Cross-Validation,* [notes], Machine Learning 10-701

35. Mivule, Kate (2011) *Difference between Sequential and Parallel Programming,* retrieved from https://mivuletech.wordpress.com/2011/01/12/difference-between-sequential-and-parallel-programming/

36. Müller Ulrich (2008) *Symmetry Relations between Crystal Structures*, Gargnano, Italy

37. Navlakha Saket and Bar-Joseph Ziv (2011) *Algorithms in nature: the convergence of systems biology and computational thinking*, Molecular Systems Biology 7; Article Number 546; doi:10.1038/msb.2011.78

38. Nguyen Derrick and Widrow Bernard (1990) *Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights*. Proceedings of the International Joint Conference on Neural Networks, 3:21–26.

39. Nissen Steffen (2003) *Implementation of a Fast Artificial Neural Network Library (FANN),* [graduate project report], Department of Computer Science, University of Copenhagen (DIKU)

40. Nissen Steffen (2005) *Neural Networks made simple*, Software 2.0

41. Porter, A.B. On the diffraction theory of microscopic vision. Phil.Mag. 11, pages 154-166 (1906)

42. Rutecki P.A. (1992) *Neuronal excitability: voltage-dependent currents and synaptic transmission*, J. Clin. Neurophysiol., Vol. 9, Ch. 2, pp. 195-211

43. Sayre D. (1952) *The Squaring Method: a New Method for Phase Determination*, Acta Crystallohraphica, Vol. 5, pp. 60-65

44. Serway A. Raymond, Jewett W. John (2013) *Physics for Scientists and Engineers, 9th edition.* Brooks Cole. Boston, USA. ISBN: 1133947271

45. Sherwood D. and Cooper J. (2010) *Crystals, X-rays and Proteins. Comprehensive Protein Crystallography*, Oxford University Press, Oxford, United Kingdom, ISBN: 9780199559046

46. Shoji-Kasai, Yoko; Hiroshi Ageta; Yoshihisa Hasegawa; Kunihiro Tsuchida; Hiromu Sugino; Kaoru Inokuchi (2007) *Activin increases the number of synaptic contacts and the length of dendritic spine necks by modulating spinal actin dynamics*, Journal of Cell Science Vol. 120, pp. 3830-3837

47. Taylor Garry (2003) *The phase problem*, Acta Crystallographica, Section D, Vol. 59, pp. 1881-1890

48. Theodosi-Kokkinou Laoura (2013) *Τεχνητά Νευρωνικά Δίκτυα και εφαρμογές στα Συστήματα Αυτόματου Ελέγχου,* [thesis], Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, Πολυτεχνική Σχολή, Πανεπιστήμιο Πατρών

49. Thorndike, E. L. (1898) *Animal intelligence: An experimental study of the associative processes in animals*. Psychological Monographs: General and Applied, 2(4), i-109.

50. Tulving, Endel (2002) *Episodic Memory: From Mind to Brain.* Annual Review of Psychology. 53: 1–25. doi:10.1146/annurev.psych.53.100901.135114.

51. Ullman, M.T. (2004) *Contributions of memory circuits to language: the declarative/procedural model*. Cognition. 92: 231–70. doi:10.1016/j.cognition.2003.10.008.

52. Usón Isabel, Sheldrick M. George (1999) *Advances in direct methods for protein crystallography*, Structural Biology, Vol. 9, pp. 643-648

53. Yam Y.F. Jim, Chow W.S. Tommy (2000) *A weight initialization method for improving training speed in feedforward neural network,* Neurocomputing Vol. 30, p. 219-232

54. Zimmermann Hans-Jürgen et al. (2002) *Advances in Computational intelligence and learning : methods and applications*, Kluwer, B.V. Deventer. The Netherlands. DOI 10.1007/978-94-010-0324-7

Appendix I

Example of training data file for structure factors of 2 atoms as mentioned on *2.1*. The first line cites the number of the training patterns (10), the number of reflections as input (7) and the number of the phases as output (1).

```
10 7 1
+1.87608 +2.97672 +0.02324 +1.64449 +0.12569 +1.87608 +0.77821
+1
+0.83292 +1.09425 +1.09765 +1.76183 +0.34105 +0.83292 +1.54783
-1
+2.22952 +0.07125 +1.02207 +0.49430 +1.32892 +2.22952 +2.10555
+1
+1.29444 +3.42147 +1.96316 +2.16837 +0.38278 +1.29444 +1.67511
-1
+2.15153 +1.34987 +0.20732 +3.41033 +1.86049 +2.15153 +2.06882
+1
+0.51394 +3.51686 +2.12215 +0.97241 +0.41264 +0.51394 +0.68217
-1
+0.90814 +3.58643 +1.02128 +0.21268 +0.68247 +0.90814 +1.22959
+1
+1.31562 +2.83897 +2.65778 +1.33488 +1.25583 +1.31562 +2.51276
-1
+0.03749 +0.35743 +1.95507 +0.48679 +3.22831 +0.03749 +0.08466
-1
+2.06480 +1.94201 +0.22693 +0.61404 +1.72346 +2.06480 +0.06126
+1
```

Appendix II

Final program in C for creating, training and testing an ANN using the FANN library, the implications of which are analyzed on *2.2-2.3* paragraphs. It is consisted of 126 input neurons, 2 hidden layers of 60 hidden neurons in each layer, and 1 output neuron. The desired MSE is 0.001 and the maximum number of epochs for training is 1000. The limit of the failed bits is set 0.5. The activation function on hidden layer is FANN Elliot symmetric, while on output layer is FANN Linear piece symmetric. RPROP training algorithm is default. The network reads a “training.dat” training data file and uses 90% of the patterns for training and 10% for evaluation on cross validation. Weights are initialized using the Nguyen and Widrow’s algorithm before the training process. Every epoch of training, the network gives feedback on the MSE and the number of the failed bits on training and cross validation. An additional line forces the data patterns to shuffle:

```
fann_shuffle_train_data( train );
```

Whether the MSE reaches the limit (0.001), the training process stops. Upon completion of the training, the network is saved on the current directory and a final test is taken place for the network to be evaluated. The results of the testing process are also saved on the current directory.

```c
#include <unistd.h>
#include "floatfann.h"


int main()
{
        struct fann_train_data *all;
        struct fann_train_data *train;
        struct fann_train_data *test;
        fann_type     *out;

        char   filename[200];
        FILE   *fileout;

        int           i;
        float         error;
        unsigned int bitfail_train, bitfail_test;

        const unsigned int num_input  = 126;
        const unsigned int num_output = 1;
        const unsigned int num_layers = 4;
        const unsigned int num_neurons_hidden = 60;
        const float desired_error = (const float) 0.001;
        const unsigned int max_epochs = 1000;


        setlinebuf( stdout );

        struct   fann   *ann   =   fann_create_standard(num_layers,   num_input, num_neurons_hidden, num_neurons_hidden, num_output);

        fann_set_activation_function_hidden(ann, FANN_ELLIOT_SYMMETRIC);
        fann_set_activation_function_output(ann, FANN_LINEAR_PIECE_SYMMETRIC);

        fann_set_bit_fail_limit( ann, 0.50 );
/*
        fann_set_training_algorithm( ann, FANN_TRAIN_RPROP );
*/
        all   = fann_read_train_from_file("training.dat");
        train =  fann_subset_train_data( all, 0, (int)(0.90*fann_length_train_data(all)) );
        test  =  fann_subset_train_data( all, (int)(0.90*fann_length_train_data(all)), (fann_length_train_data(all)-(int)(0.90*fann_length_train_data(all))) );


        fann_init_weights( ann, train );

        printf("\nTraining  :  %d  sets,    cross-validation  :  %d\n", (int)(0.90*fann_length_train_data(all)), (fann_length_train_data(all)-(int)(0.90*fann_length_train_data(all))) );

        printf("\nCycle MSE(training) MSE(test)     Bitfail(train/test)\n");
        printf("----- ------------- ---------     -------------------\n");
        for(i = 1 ; i <= max_epochs ; i++)
        {
                error = fann_train_epoch(ann, train);
                bitfail_train = fann_get_bit_fail( ann );
                fann_reset_MSE(ann);
                fann_test_data(ann, test);
                bitfail_test = fann_get_bit_fail( ann );
                printf("%5d      %6.4f      %6.4f            %5d  /%5d\n", i, error, fann_get_MSE(ann), bitfail_train, bitfail_test );
/*
                fann_shuffle_train_data( train );
*/
                if ( error < desired_error )
                {
                        break;
                }
        }

        sprintf( filename, "%d.net", getpid() );
        fann_save(ann, filename );

        sprintf( filename, "%d.output", getpid() );
        fileout = fopen( filename, "w" );

        fprintf(fileout, "Statistics (MSEs) for this run : %6.4f %6.4f\n", error, fann_get_MSE(ann) );

        for ( i=0 ; i < fann_length_train_data(test) ; i++ )
                {
                        out = fann_run( ann, test->input[i] );
                        fprintf(fileout, "%+4.2f  %+4.2f\n", i, test->output[i][0], out[0] );
                }

        fclose( fileout );

          fann_destroy_train(train);

        fann_destroy(ann);

        return 0;
}
```

Appendix III

Program in C for testing a trained ANN as mentioned on *2.4* upon newly introduced data “training.dat”. A network is being created from the saved information about weights from file “out.net”. The network produces outputs for the input values of the training patterns and then it is evaluated by comparing the produced outputs with the correct, desired ones.

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include "fann.h"

int main()
{

        fann_type *out;
        unsigned int i;

        struct fann *ann;
        struct fann_train_data *test;
        char   filename[200];
        FILE   *fileout;

        printf("Creating network.\n");

        ann = fann_create_from_file("out.net");

        if(!ann)
        {
                printf("Error creating ann --- ABORTING.\n");
                return -1;
        }

        fann_print_connections(ann);
        fann_print_parameters(ann);

        printf("Testing network.\n");

        test = fann_read_train_from_file("training.dat");

        sprintf( filename, "%d.output", getpid() );
        fileout = fopen( filename, "w" );

        fprintf(fileout, "Test:");

        for ( i=0 ; i < fann_length_train_data(test); i++ )
                {
                        out = fann_run( ann, test->input[i] );
                        fprintf(fileout,  "%+4.2f  %+4.2f\n",  i,  test->output[i][0],
out[0] );
                }

        fclose( fileout );
        printf("Cleaning up.\n");
        fann_destroy_train(test);
        fann_destroy(ann);

        return 0;
}
```

Appendix IV

The output of the testing process mentioned on *2.4* of an ANN after completion of training. On the first line the MSE for the training and the cross validation are presented in sequence. Next lines refer to the desired, correct output of the test file (1st column) and the output produced by the trained network (2nd column).

```
Statistics (MSEs) for this run : 0.0644 0.2516
-1.00     -1.00
+1.00     -0.05
-1.00     -1.00
+1.00     +0.03
-1.00     +0.32
+1.00     +0.62
+1.00     +1.00
-1.00     -1.00
+1.00     +0.98
+1.00     -1.00
+1.00     +0.76
-1.00     +0.45
+1.00     +1.00
+1.00     -1.00
```

Appendix V

Example of a saved network upon completion of the training procedure, as mentioned on the end of *2.3*. The network is consisted of 7 input neurons, 7 hidden neurons in 1 hidden layer and 1 output neuron. It is trained upon 140 training patterns of structure factors of 2 atoms.

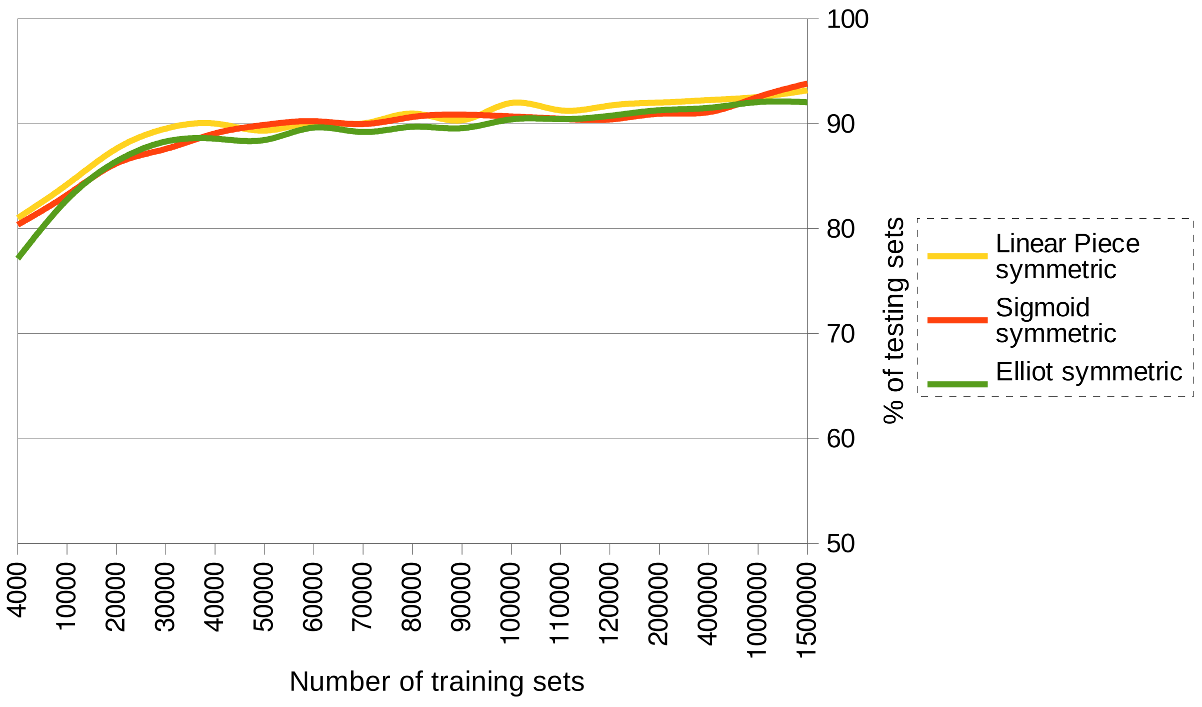
```
FANN_FLO_2.1
num_layers=3
learning_rate=0.700000
connection_rate=1.000000
network_type=0
learning_momentum=0.000000
training_algorithm=2
train_error_function=1
train_stop_function=0
cascade_output_change_fraction=0.010000
quickprop_decay=-0.000100
quickprop_mu=1.750000
rprop_increase_factor=1.200000
rprop_decrease_factor=0.500000
rprop_delta_min=0.000000
rprop_delta_max=50.000000
rprop_delta_zero=0.100000
cascade_output_stagnation_epochs=12
cascade_candidate_change_fraction=0.010000
cascade_candidate_stagnation_epochs=12
cascade_max_out_epochs=150
cascade_min_out_epochs=50
cascade_max_cand_epochs=150
cascade_min_cand_epochs=50
cascade_num_candidate_groups=2
bit_fail_limit=5.00000000000000000000e-01
cascade_candidate_limit=1.00000000000000000000e+03
cascade_weight_multiplier=4.00000005960464477539e-01
cascade_activation_functions_count=10
cascade_activation_functions=3 5 7 8 10 11 14 15 16 17
cascade_activation_steepnesses_count=4
cascade_activation_steepnesses=
2.50000000000000000000e-01   5.00000000000000000000e-01
     7.50000000000000000000e-01   1.00000000000000000000e+00
layer_sizes=8 8 2
scale_included=0

neurons
(num_inputs,   activation_function,     activation_steepness)   =
(0,            0,                       0.00000000000000000000e+00)
(0,            0,                       0.00000000000000000000e+00)
(0,            0,                       0.00000000000000000000e+00)
(0,            0,                       0.00000000000000000000e+00)
(0,            0,                       0.00000000000000000000e+00)
(0,            0,                       0.00000000000000000000e+00)
(0,            0,                       0.00000000000000000000e+00)
(0,            0,                       0.00000000000000000000e+00)
(8,            11,                      5.00000000000000000000e-01)
(8,            11,                      5.00000000000000000000e-01)
(8,            11,                      5.00000000000000000000e-01)
(8,            11,                      5.00000000000000000000e-01)
(8,            11,                      5.00000000000000000000e-01)
(8,            11,                      5.00000000000000000000e-01)
(8,            11,                      5.00000000000000000000e-01)
(0,            11,                      0.00000000000000000000e+00)
(8,            13,                      5.00000000000000000000e-01)
(0,            13,                      0.00000000000000000000e+00)
```

```
connections
(connected_to_neuron,        weight)   =
(0,                          -4.34681236743927001953e-01)
(1,                          -7.58793294429779052734e-01)
(2,                          -3.52416872978210449219e+00)
(3,                           7.82115507125854492188e+00)
(4,                           7.44669497013092041016e-01)
(5,                          -3.39526593685150146484e-01)
(6,                           5.38342332839965820312e+00)
(7,                           1.88518965244293212891e+00)
(0,                           7.48186528682708740234e-01)
(1,                           3.67557024955749511719e+00)
(2,                          -1.16717791557312011719e+00)
(3,                          -1.40391731262207031250e+01)
(4,                           9.08543050289154052734e-01)
(5,                           7.91132688522338867188e-01)
(6,                          -6.29477930068969726562e+00)
(7,                           2.97550964355468750000e+00)
(0,                           7.70006775856018066406e-01)
(1,                           4.93141508102416992188e+00)
(2,                           2.22041636705398559570e-01)
(3,                           1.49636435508728027344e+00)
(4,                           2.11342167854309082031e+00)
(5,                           6.86577200889587402344e-01)
(6,                           3.39053893089294433594e+00)
(7,                          -1.46971292495727539062e+01)
(0,                          -7.37441480159759521484e-01)
(1,                          -2.55124378204345703125e+00)
(2,                          -7.15351629257202148438e+00)
(3,                           2.14271378517150878906e+00)
(4,                          -5.63871431350708007812e+00)
(5,                          -6.45886719226837158203e-01)
(6,                           1.24140157699584960938e+01)
(7,                           7.85034799575805664062e+00)
(0,                          -4.38544750213623046875e-01)
(1,                          -3.36525499820709228516e-01)
(2,                           7.83576905727386474609e-01)
(3,                           7.27182924747467041016e-01)
(4,                          -2.22151905298233032227e-01)
(5,                          -2.78709053993225097656e-01)
(6,                          -6.56169831752777099609e-01)
(7,                           2.28409075736999511719e+00)
(0,                           3.09093832969665527344e+00)
(1,                          -2.04296326637268066406e+00)
(2,                           9.33002412319183349609e-01)
(3,                          -1.61425342559814453125e+01)
(4,                           1.05581057071685791016e+00)
(5,                           2.96433115005493164062e+00)
(6,                          -2.86347889900207519531e+00)
(7,                           2.96372795104980468750e+00)
(0,                          -2.89040422439575195312e+00)
(1,                           8.49083006381988525391e-01)
(2,                          -2.00622305274009704590e-01)
(3,                           1.89347229003906250000e+01)
(4,                          -2.94706225395202636719e+00)
(5,                          -3.06512904167175292969e+00)
(6,                           3.82614850997924804688e+00)
(7,                           1.05424873530864715576e-01)
(8,                           3.64621043205261230469e+00)
(9,                           5.87012195587158203125e+00)
(10,                          2.93393301963806152344e+00)
(11,                          1.96664261817932128906e+00)
(12,                          1.08182411193847656250e+01)
(13,                          8.22098064422607421875e+00)
(14,                          4.82929944992065429688e+00)
(15,                         -2.94149971008300781250e+00)
```

Appendix VI

Some additional attempts of further research as suggested in the *Discussion* chapter, on multiple numbers of training sets were conducted on neural networks with activation function on output layer being FANN Sigmoid symmetric and FANN Elliot symmetric, compared to the attempts previously presented on *3.2.5* with FANN Linear piece symmetric. Networks had 126 input neurons, 2 hidden layers with 60 neurons in each one and 1 neuron in the output layer. The training algorithm was default and the activation function on the hidden layer was FANN Elliot symmetric. 10% of the training sets were used for evaluation. No differences were noticed on the performance of the trained network, although the network with the FANN Linear piece symmetric activation function on the output layer produced slightly more correct outputs in total.


% of testing sets
Linear Piece symmetric
Sigmoid symmetric
Elliot symmetric
Number of training sets