Approaching the crystallographic phase problem with Artificial Neural Networks

Dionysia Moschou[1], Nicholas M. Glykos[1]

[1] Democritus University of Thrace, Dept. of Molecular Biology and Genetics, Structural and computational biology lab, Alexandroupolis, Greece

The objective of this work is for the network to learn the necessary algorithmic operations in order *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.

Introduction

Introduction to crystallography and the phase problem

In x-ray crystallography, the molecular structure of a crystal is determined by calculating the electron density function according to the Fourier factors, which operation needs the values of all the structure factors composed of two elements: their magnitude ($|F_{hkl}|$) and phase (a_{hkl}). In the experiments, however, only the intesities of the reflections are measured, and information on the relative phases is lost. This loss of information constitutes the *Phase Problem*.

Algorithmic methods are depeloped to identify these phases. *Direct methods* are based on the positivity and atomicity of electron density and leads through statistical methods to phase relationships.

$$\mathbf{F}_{hkl} = \sum_{h'k'l'} \mathbf{F}_{h'k'l'}\mathbf{F}_{h-h',k-k',l-l'}$$

Figure 1: Sayre's equation[1]

In two dimensions, it is relatively easy to solve the phase problem directly, but not so in three dimensions. The key step was taken by David Sayre who introduced a mathematical relationship that allows to calculate probable values for the phases of some diffracted beams(Fig. 1).

Artificial Neural Networks

ANNs are processing devices that are loosely modeled after the neuronal structure of the mamalian cerebral cortex but on much smaller scales. Neural neworks are typically organized in layers made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output.

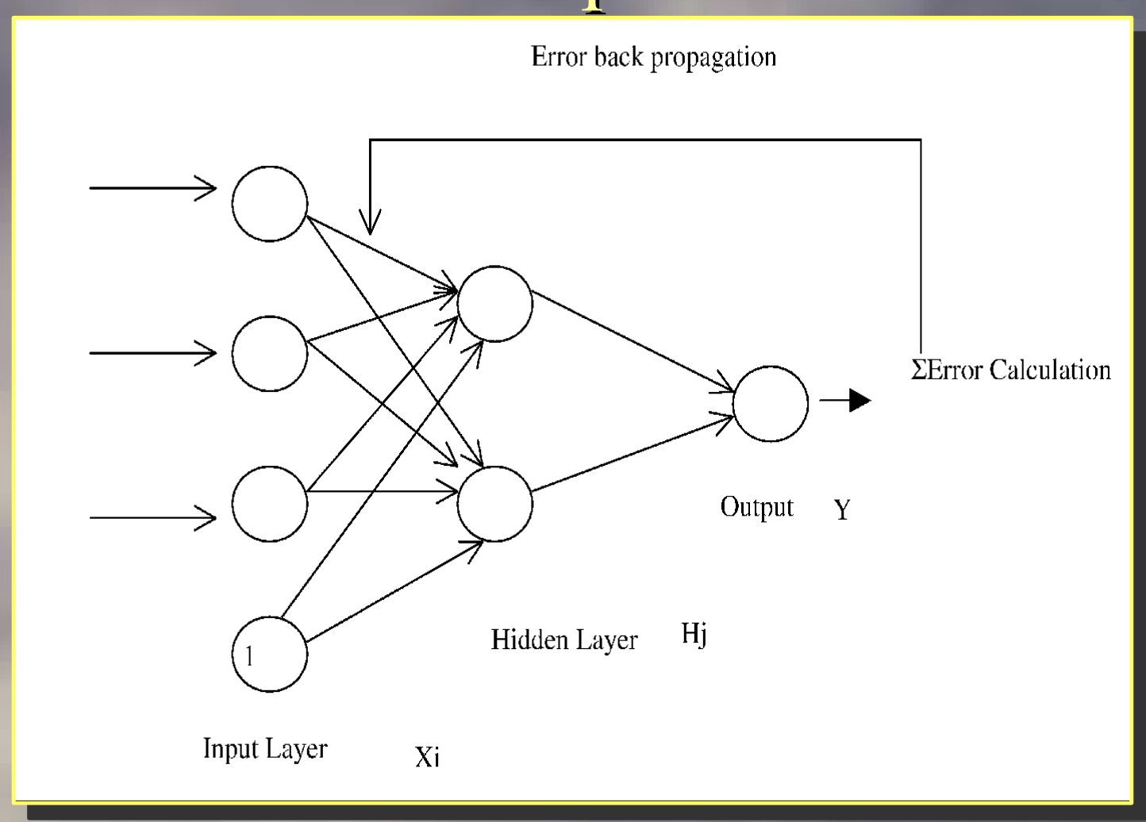


Figure 2: Typical backpropagational neural network

'Learning' is a supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments by calculating the *Mean Square Error*(Fig. 2)[2].

Once a neural network is 'trained' to a satisfactory level it may be used as an analytical tool on other data.

Computational tools

FANN Library
(Fast Artificial Neural Network Library):
"a free open source neural network library, which implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks."[3]

C language:
a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs.

Pepinsky's machine:
a program aiming to help with the teaching of crystallographic Fourier transforms. The program is able to calculate and interactively display the electron density maps corresponding to all phase combinations of a user-defined subset of structure factors[4].

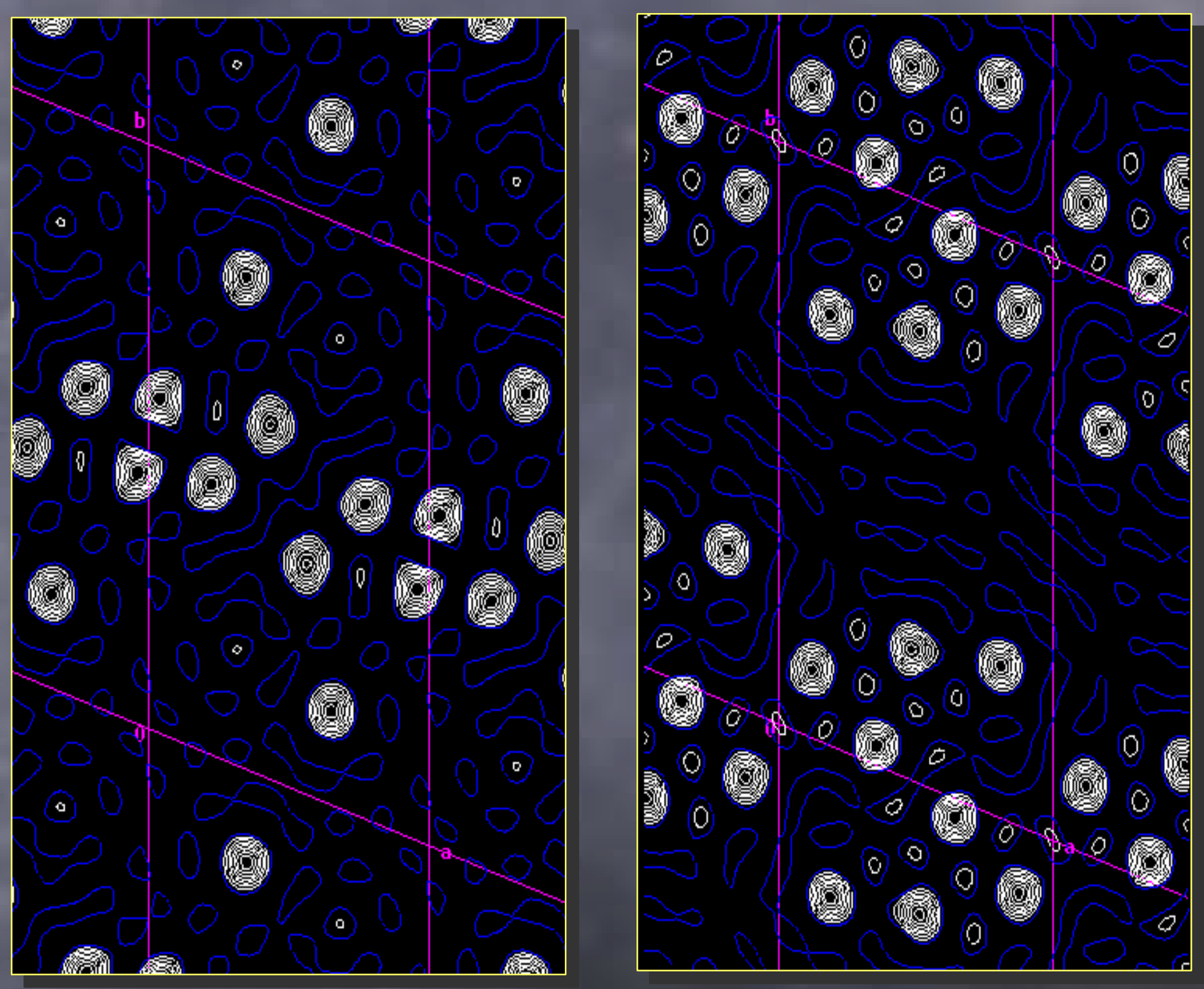
Method

Structure factors were calculated and data files were constructed as training sets containing all the structure's amplitudes' e-values as 'input' and a single 'output' of one phase. For a structure of 4 atoms in a cell of 7.0x12.0 Å, the pepinsky's machine receives 126 diffractions (Fig. 3, 4). For this case, the output is the phase of the reflection with structure factors h=-2, k=0.

At starting point, the Activation Functions for the hidden layer is defined as FANN_ELLIOT_SYMMETRIC and for the output layer as FANN_LINEAR_PIECE_SYMMETRIC (asserted by experiment).

- The first trial was to decide about *the number of the hidden layers* and *their number of neurons* with a training data of 80,000 sets (100%).
- Following this, a standard condition of 2 hidden layers and 60 neurons each was used to train the neural network by *different number of training sets* (100%) of the training data.

The network uses the training data to learn the relationships between the 'input' and the 'output'. At each 'epoch' the network used 90% of the training data to be trained, and 10% for Cross Validation so it evaluates the procedure. This training was set to stop at 1,000 epochs unless the Mean Square Error (MSE) of the training reaches a certain value (0.0001). At the end of each training trial, the network used the 10% of the training data to a feed-forward program so as to test its ability to estimate the correct output (phase) of each set.



Figures 3 and 4: Electron density maps for structures of 4 atoms in 7.0x12.0 cell as shown by Pepinsky's machine

Results and Conclusion

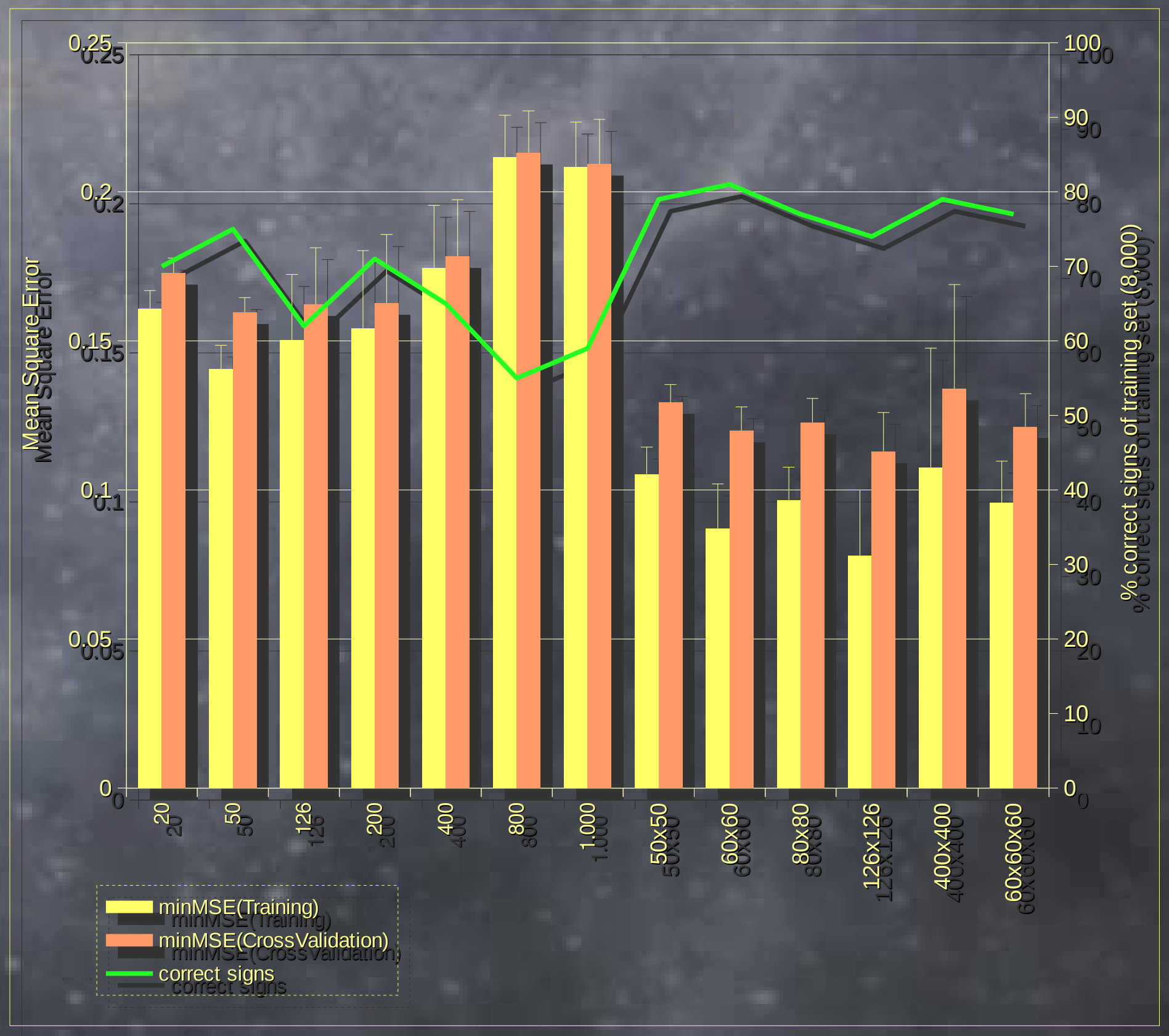


Figure 5: Trials by number of hidden layers and neurons

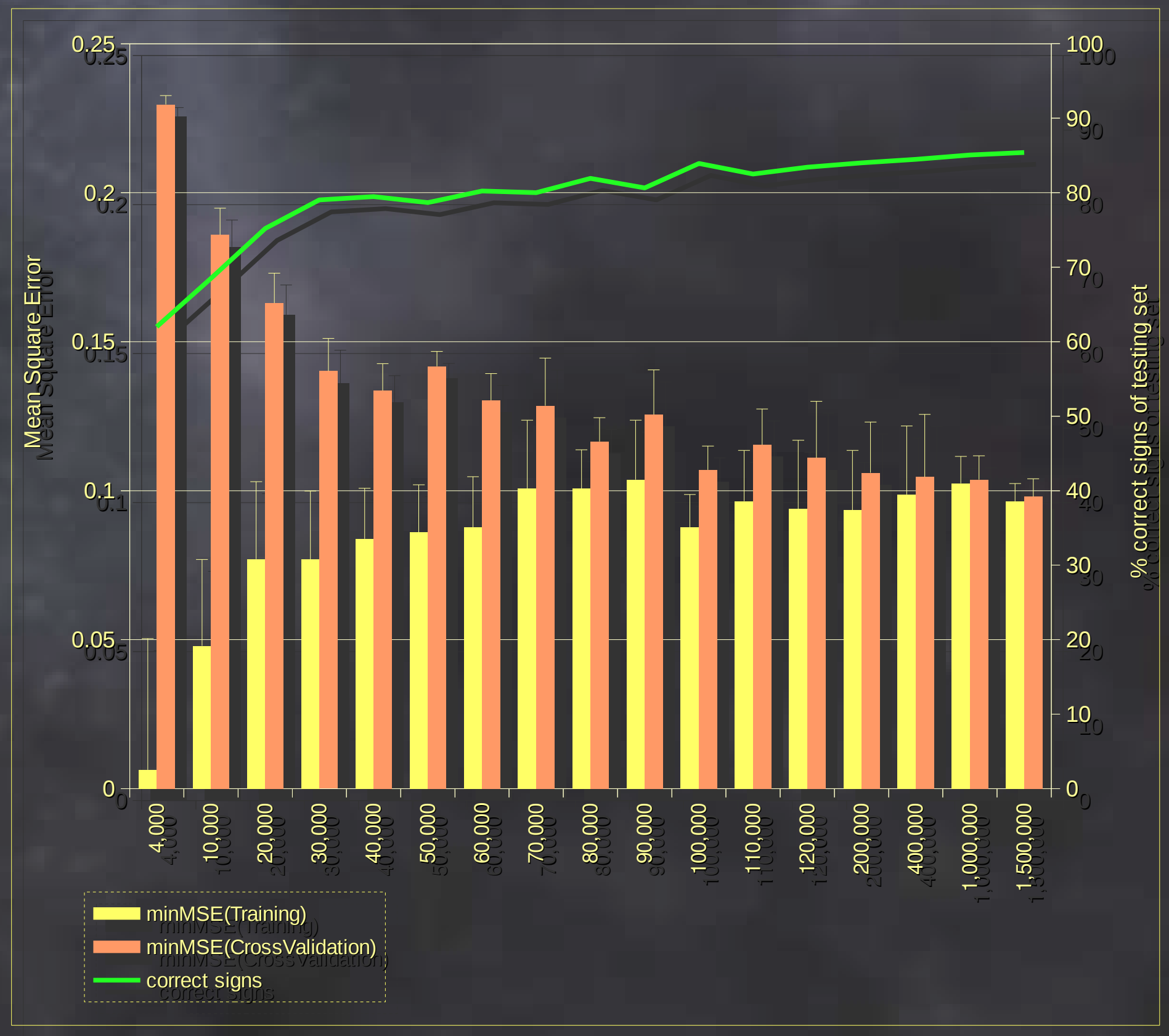


Figure 6: Trials by number of training sets (100%)

Figure 5 shows that the best training conditions are at the right part of the graph. Two or more hidden layers are capable of greater accuracy on the operations. There is lower Mean Square Error for the training and the Cross Validation, and a higher precision on the prediction of the correct outputs.

Figure 6 concludes the number of training sets is related to the training: the network needs more sets to learn the operations better. When the number of training sets increases, the Mean Square Error of the Cross Validation is minimized and the prediction of the correct output rises consecutively and reaches 85.3956% for 10% of 1,500,000 training sets.

Given these points, we can definitely train an ANN for the purpose, though more trials have to be performed.

References

[1] Glykos, N.M. (2015), ""A non-mathematical introduction to protein crystallography", e-book available at: http://hdl.handle.net/11419/3924,, p.212

[2] "A Basic Introduction to Neural Networks", University of Wisconsin-Madison

[3] Steffen Nissen (2003), "Implementation of a Artificial Neural Network Library (FANN), Department of Computer Science, University of Copenhagen (DIKU)

[4] Glykos, N.M. (1999), "Pepinsky's Machine: an interactive, graphics-based Fourier synthesis program with applications in teaching and research.", J. Appl. Crystallogr., 32, 821-823.

QR: POSTER (PDF)