MONASH University
Engineering

Assessment cover sheet

Unit and student details

Unit code	TRC 4901	**Unit title**	Artificial Intelligence		
If this is a group assignment, each student must include their name and ID number and sign the student statement.					
Student ID	29392004	**Surname**	Ranjan	**Given names**	Rivyesch
Student ID		**Surname**		**Given names**	
Student ID		**Surname**		**Given names**	
Student ID		**Surname**		**Given names**	

Assessment details

Title of assignment /lab	Lab 4 - Character recognition using a Backpropagation Neural Network	**Authorised group assignment**	Yes ☐ No ☑
Lecturer/tutor	Dr. Parasuraman	**Tutorial day and time**	Wednesday 8 am
Due date	19/05/2021	**Date submitted**	17/05/2021
Has any part of this assessment been previously submitted as part of another unit/course?			Yes ☐ No ☑

Submission date and extensions

All work must be submitted by the due date. If an extension of work is required, please complete and submit a Special Consideration application (in-semester assessment task) form to your examiner/lecturer/tutor.

Plagiarism and collusion

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations.

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student statement and signature

Student Statement

I have read the university's Student Academic Integrity Policy and Procedures.
I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://www.monash.edu/legal/legislation/current-statute-regulations-and-related-resolutions
I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
No part of this assignment has been previously submitted as part of another unit/course.
I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:

i. provide to another member of faculty and any external marker; and/or
ii. submit it to a text matching software; and/or
iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.

I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Student signature	Rivyesch Ranjan	**Date**	17/05/2021
Student signature		**Date**	
Student signature		**Date**	
Student signature		**Date**	

Please note that it is your responsibility to retain a copy of your assignment/laboratory work

Privacy Statement

The information on this form is collected for the primary purpose of assessing your assignment/laboratory work and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters, and statistical analyses. If you choose not to complete all the questions on this form Monash University it may not be possible for Monash University to assess your assignment/laboratory work. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. You have a right to access personal information that Monash University holds about you, subject to any exception in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

Table of Contents

1.0 Introduction 4
2.0 Methodology 4
2.1 Creating Input Characters 4
2.2 Preparing inputs for training 4
2.3 Designation of Targets 5
3.0 Training Neural Network 5
4.0 Results 6
4.1 Variation of Number of Neurons and Number of Layers 6
4.2 Variation of Training Algorithm 7
4.3 Variation of Transfer Function in Each Layer 7
5.0 Discussion 7
6.0 Conclusion 8
7.0 Appendix 9
7.1 Variation of Number of Neurons and Number of Hidden Layers 9
7.1.1 Case 1 9
7.1.2 Case 2 10
7.1.3 Case 3 11
7.1.4 Case 4 12
7.1.5 Case 5 13
7.1.6 Case 6 14
7.1.7 Case 7 15
7.1.8 Case 8 16
7.1.9 Case 9 17
7.1.10 Case 10 18
7.2 Variation of Training Algorithms 19
7.2.1 BFGS Quasi-Newton 19
7.2.2 Resilient Backpropagation 20
7.2.3 Scaled Conjugate Gradient 21
7.2.4 Fletcher-Powell Conjugate Gradient 22
7.2.5 Polak-Ribiere Conjugate Gradient 23
7.2.6 Variable Learning Rate Backpropagation 24
7.3 Variation of Transfer Function 25
7.3.1 Case 11 25
7.3.2 Case 12 26
7.3.4 Case 14 27

7.4 Matlab Code for Random Noise Creation 28

1.0 Introduction

A Back Propagation Neural Network (BPNN) is designed for character recognition purposes to authenticate the identity of a person as well as a document. In this lab the BPNN is trained to recognize 10 characters comprising of a mixture of upper-case letters and numbers. The network was trained using a combination of perfect quality inputs and inputs with random noise. Once the network is established, the effect of different training parameters such as number of layers, number of neurons in each layer and training algorithms on the performance of the BPNN are investigated and analysed.

2.0 Methodology

2.1 Creating Input Characters

Ten unique alphanumeric characters were chosen to be input into the neural network for training. These are 'H', 'I', 'V', 'Y', 'E', 'S', '2', '3', '4' and '0'. Each character was designed in a 10 x 8 array by populating it with Boolean values of 0s and 1s in MATLAB. These alphanumeric characters are then each duplicated four times. The duplicates are then subjected to random noise which essentially just inverts the Boolean values at random locations. The noise function created to induce this noise limits the number of pixels that can be inverted to a maximum of four. Figure 1 seen below shows all the samples used in the training process, with the left most column containing all of the alphanumeric characters without noise.

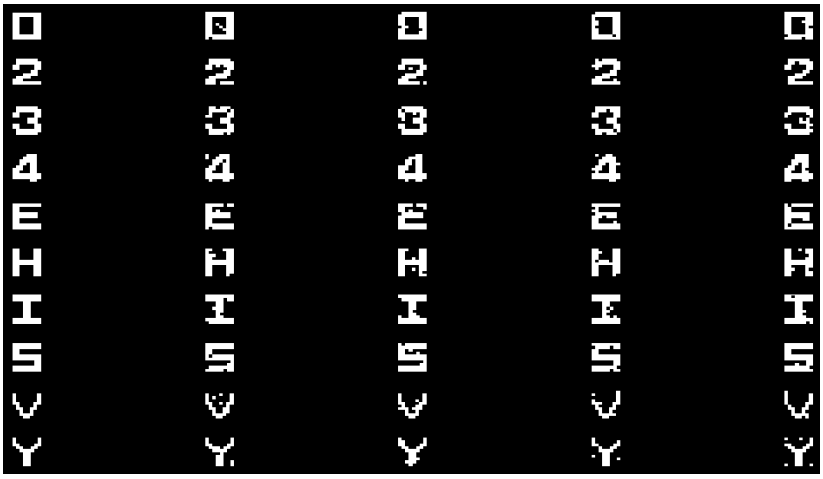


Figure 1: All characters used as part of training and testing dataset

2.2 Preparing inputs for training

Each input characters seen above is then reshaped into a column vector and then combined with other characters. The inputs are reshaped using the 'reshape' function in MATLAB so that it can be used in the training of the BPNN. Hence, there is a total of 80 rows in Figure 2 due to the size of each image being 10 x 8. Each column seen in Figure 2 corresponds to one of the 50 characters seen in Figure 1. The ordering of the columns is done such that the first five columns correspond to the five images of the first letter created and so on.

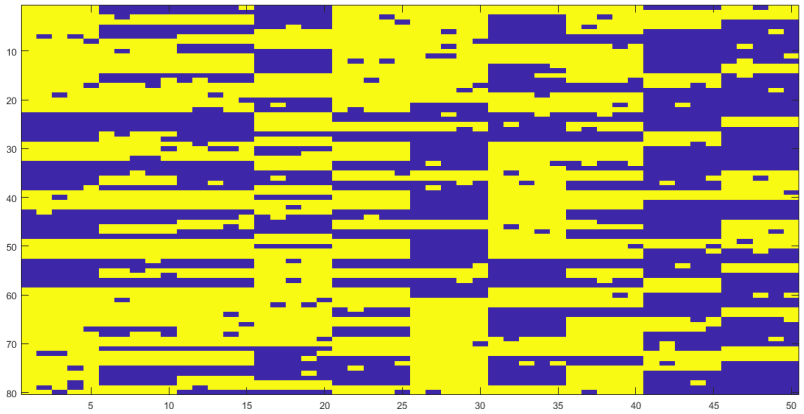


Figure 2: Pre-processed input for training neural network

2.3 Designation of Targets

Since a BPNN is one of many supervised learning algorithms, it requires a target output to be specified so that the network can establish successful mapping between the provided inputs and the target output through iterative backpropagation learning. Each 'step' from left to right represent the five samples of each input character in the order that they appear in Figure 1 and Figure 2. It is important to note that the target output clearly identifies the four duplicates of each character created as being the character itself. This helps the network map the samples with noise to the correct noise-free character by providing a reference to compare to.

Figure 3: Target output for training

3.0 Training Neural Network

The training process is done in the neural network GUI by initializing the NN toolbox with the command nntool. The input data and the target data are imported from the workspace.

The following parameters are varied one at a time while the others are kept fixed:

1. Number of Hidden Layers
2. Number of Neurons in Each Hidden Layer
3. Type of Training Algorithms
4. Transfer Function of Each Layer

Meanwhile the maximum number of epochs is set to 200 and the validation checks before the training is ended is kept constant at 6 validation checks.

Firstly, the variation of number of neurons and number of layers is investigated to determine the optimal network architecture that yields the lowest MSE and highest regression. The results can be viewed in section 4.1. Due to time constraints and limited computational resources and memory, the maximum number of hidden layers used was capped at two layers while the maximum number of neurons in each layer was limited to 50 neurons. Using the network that is considered to be the optimal case, the architecture is kept constant while the training algorithm is varied. A total of six training algorithms was tested for the optimal architecture. For the sake of thoroughness, the type of transfer function used in each layer of the network is also varied and then investigated.

Once the network is created and trained it is simulated using the test data to determine the efficiency of training on the network and its overall accuracy and reliability. The results are rounded before comparing the predicted output and the actual output.

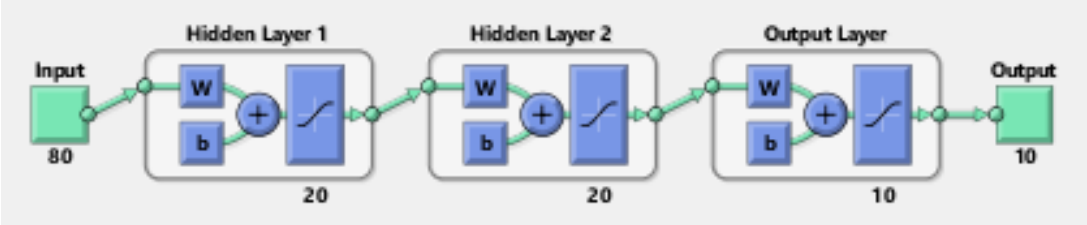


Figure 4: Neural Network architecture

4.0 Results

4.1 Variation of Number of Neurons and Number of Layers

Table 1: Experimental study of influence of neurons and hidden layers on neural network performance

Models with 1 Hidden Layer						
Case	Layers	No. of Neurons	Iterations	Elapsed Time	MSE	Regression
1	1	10	10	3	9.01E-05	0.99
	2	10				
2	1	20	11	2	1.17E-06	0.99959
	2	10				
3	1	30	9	4	1.34E-04	0.99984
	2	10				
4	1	40	9	10	6.78E-07	1
	2	10				
5	1	50	9	14	4.85E-03	0.99569
	2	10				
Models with 2 Hidden Layers						
Case	Layers	No. of Neurons	Iterations	Elapsed Time	MSE	Regression
6	1	10	11	0	1.39E-02	0.98738
	2	10				
	3	2				
7	1	20	10	4	3.30E-08	1
	2	20				
	3	2				
8	1	30	10	13	1.37E-05	0.99995
	2	30				
	3	2				
9	1	40	10	29	1.48E-05	0.99969
	2	40				
	3	2				
10	1	50	10	69	4.40E-07	1
	2	50				

	3	2				

4.2 Variation of Training Algorithm

Table 2: Experimental study of influence of training algorithms on neural network performance

Case 7				
Training Algorithm	Iterations	Elapsed Time	MSE	Regression
BFGS Quasi-Newton	42	165	6.03E-04	0.94082
Resilient Backpropagation	23	0	1.18E-03	0.99377
Scaled Conjugate Gradient	45	0	1.55E-04	0.87808
Fletcher-Powell Conjugate Gradient	22	0	4.07E-02	0.80712
Polak-Ribiére Conjugate Gradient	25	0	6.46E-06	0.94334
Variable Learning Rate Backpropagation	118	0	6.49E-03	0.98748
Levenberg-Marquardt	10	4	3.30E-08	1

4.3 Variation of Transfer Function in Each Layer

Table 3: Experimental study of influence of transfer function on neural network performance

Case 7 Levenberg-Marquardt						
Case	Layer	Transfer Function	Iterations	Elapsed Time	MSE	Regression
11	1	PureLin	9	7	2.34E+00	0.18083
	2	PureLin				
	3	PureLin				
12	1	LogSig	17	9	2.41E-01	0.68825
	2	LogSig				
	3	LogSig				
13	1	TanSig	10	4	3.30E-08	1
	2	TanSig				
	3	TanSig				
14	1	Tansig	11	4	8.31E-09	1
	2	Logsig				
	3	Tansig				

5.0 Discussion

Referring to the appendix section 7.1 to 7.3 the details of the training process, performances and unrounded and rounded outputs can be found for each neural network case trained during the optimisation process. Figures 5 to 14 are for the variations of neuron number and hidden layer number, Figures 15 to 20 are for the variations of the training algorithm used, and Figures 21 to 23 are for the variations of the transfer function in each layer.

The main criteria for judging the performance of each individual model created is the Mean Squared Error (MSE) and Regression value. MSE is a measure of how close a fitted line is to data points. The

smaller the MSE, the closer the fit is to the data. Regression evaluates the scatter of the data points around the fitted regression line. A higher regression value is desired as it means there is smaller difference between the observed data and the fitted values.

The study in section 4.1 investigates the effect of the number of neurons in each hidden layer and the total number of hidden layers that the ANN model is comprised of. It was determined that for single layer ANN models a greater number of neurons are required in the sole layer in order for the model to accurately represent the non-linear problem. In this case 40 neurons in the hidden layer gave the best results in terms of the highest regression and the lowest MSE value. Contrastingly, for ANN models with two hidden layers each hidden layer only needs 20 neurons to perform at the optimal level. When comparing the best single layer model and two-layer model, it is noticed that using a deeper ANN which consist of multiple hidden layers did indeed improve the ability of the model to generalize the input information. Overall, the best two-layer model had a significantly lower error compared to the best single layer model.

Using the optimal model from section 4.1 which was found to be the two hidden layer model with 20 neurons in each hidden layer, the influence of different training algorithms is analysed. The previous optimal model utilized the default and recommended Levenberg-Marquardt training algorithm. As part of the analysis six new training algorithms are introduced. It should be noted that while the training algorithm is varied, the networks architecture and other hyperparameter settings will remain fixed. The results imply that the none of the new training algorithms tested are suitable for this application. The errors were generally much worse. Fletcher-Powell Conjugate Gradient was determined to be the worst algorithm that could be used for the training and fitting of this dataset. The best algorithm remains the Levenberg-Marquardt training algorithm.

The only remaining hyperparameter yet to be analysed in the transfer function of each layer. Initially the transfer functions were varied in a way that all layers used the same transfer function. The results show that by using the TanSig transfer function in all the layers of the two-hidden layer model from the previous section the MSE error could be minimized. However, the best model with the lowest possible error was achieved by using the TanSig transfer function in the first hidden layer and the output layer while the LogSig transfer function is used in the second hidden layer.

6.0 Conclusion

The neural network has been created and optimised to accurately recognise and classify the flowers based on data of the four attributes previously mentioned. The introduction of noise to the dataset improved the model's robustness and ability to generalise and then classify. The optimisation process which involved the variation of various parameters highlighted the influence of each parameter and the importance of the optimisation process in achieving an accurate model. Overall, after the various studies conducted the best network for this particular application is a two hidden layer neural network with 20 neurons in each hidden layer using Levenberg-Marquardt training algorithm and hyperbolic tangent sigmoid transfer function in all of its layers.

7.0 Appendix

7.1 Variation of Number of Neurons and Number of Hidden Layers

7.1.1 Case 1

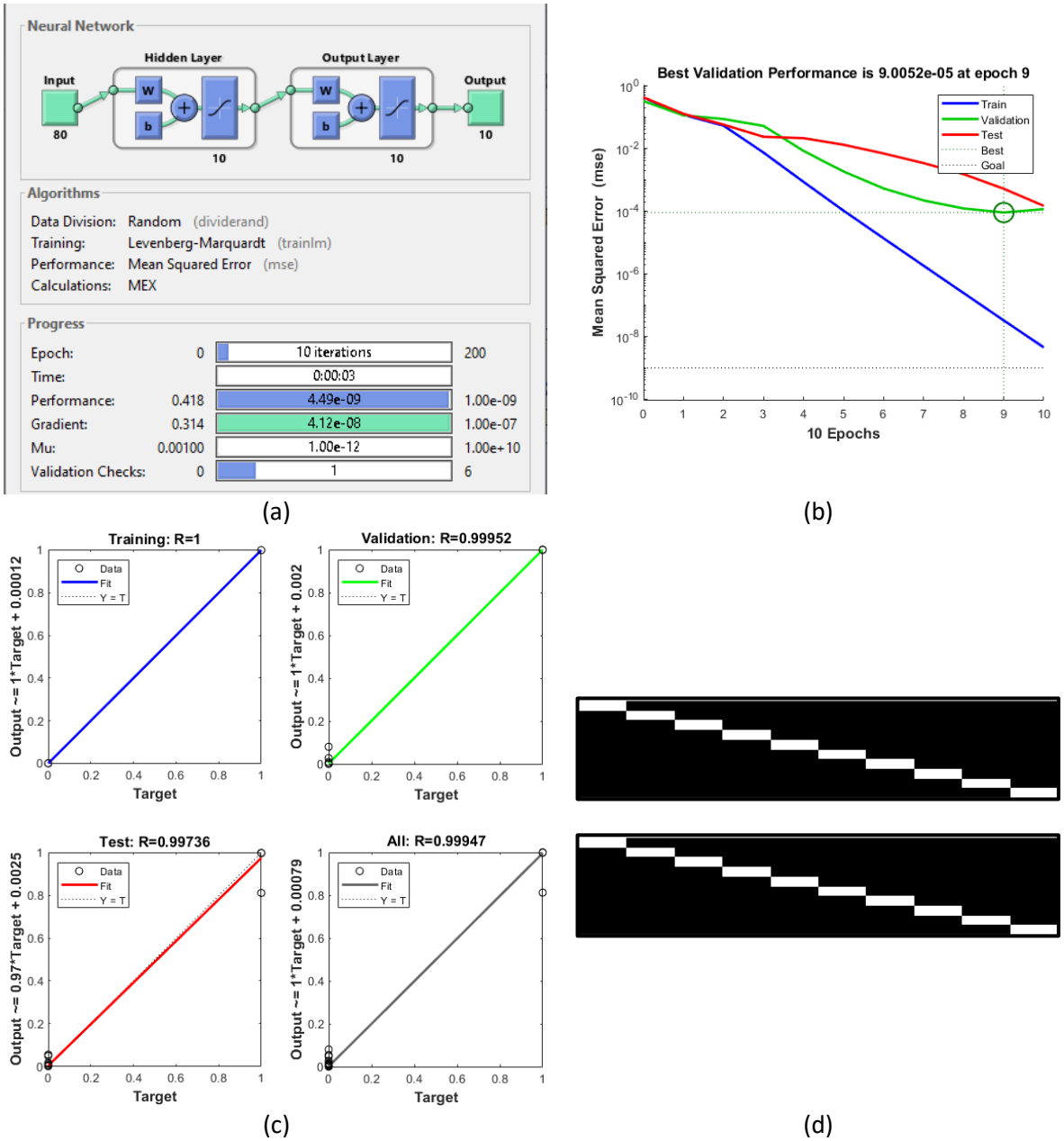


Figure 5: A 10 neuron single hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.1.2 Case 2

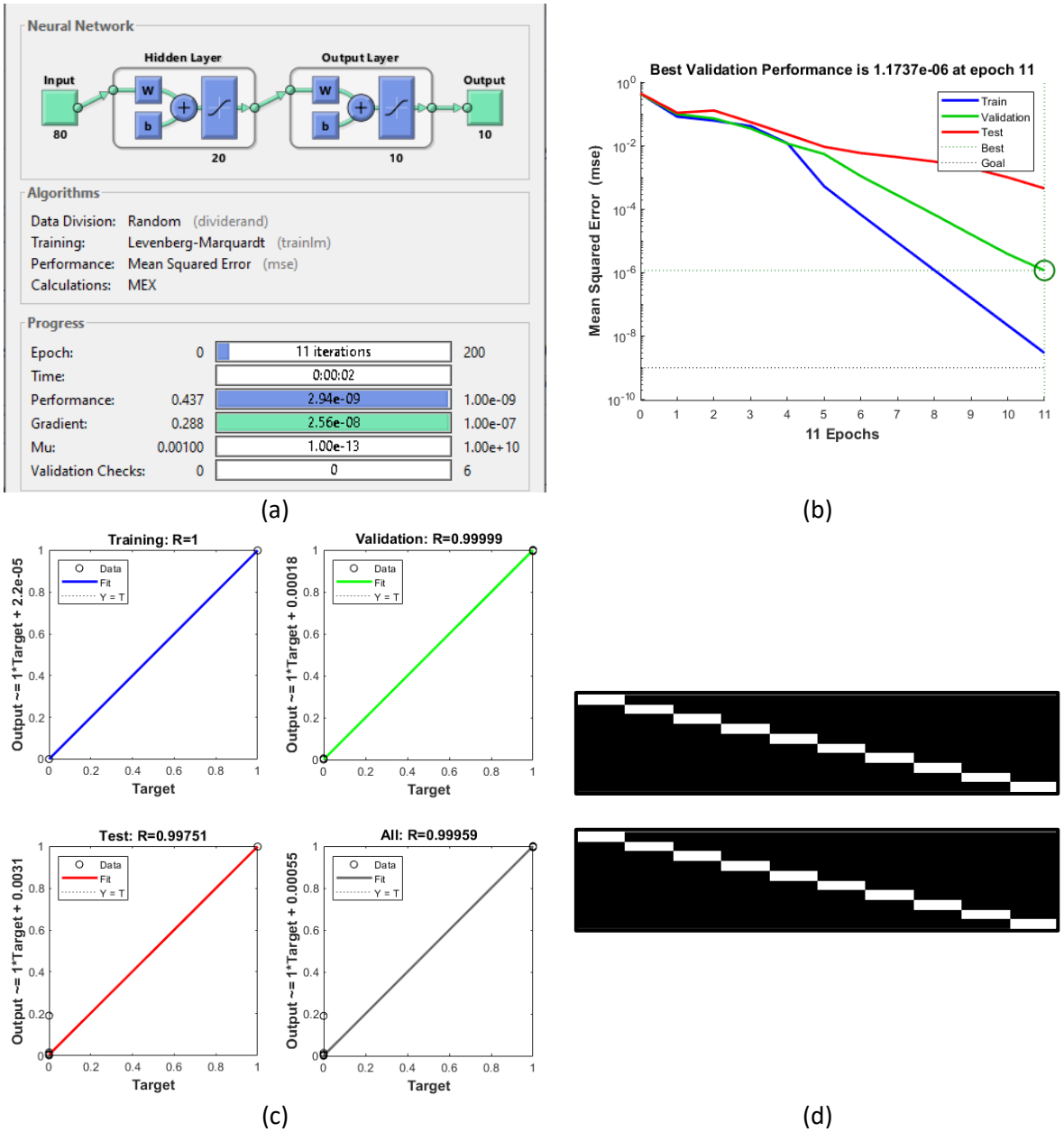


Figure 6: A 20 neuron single hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.1.3 Case 3

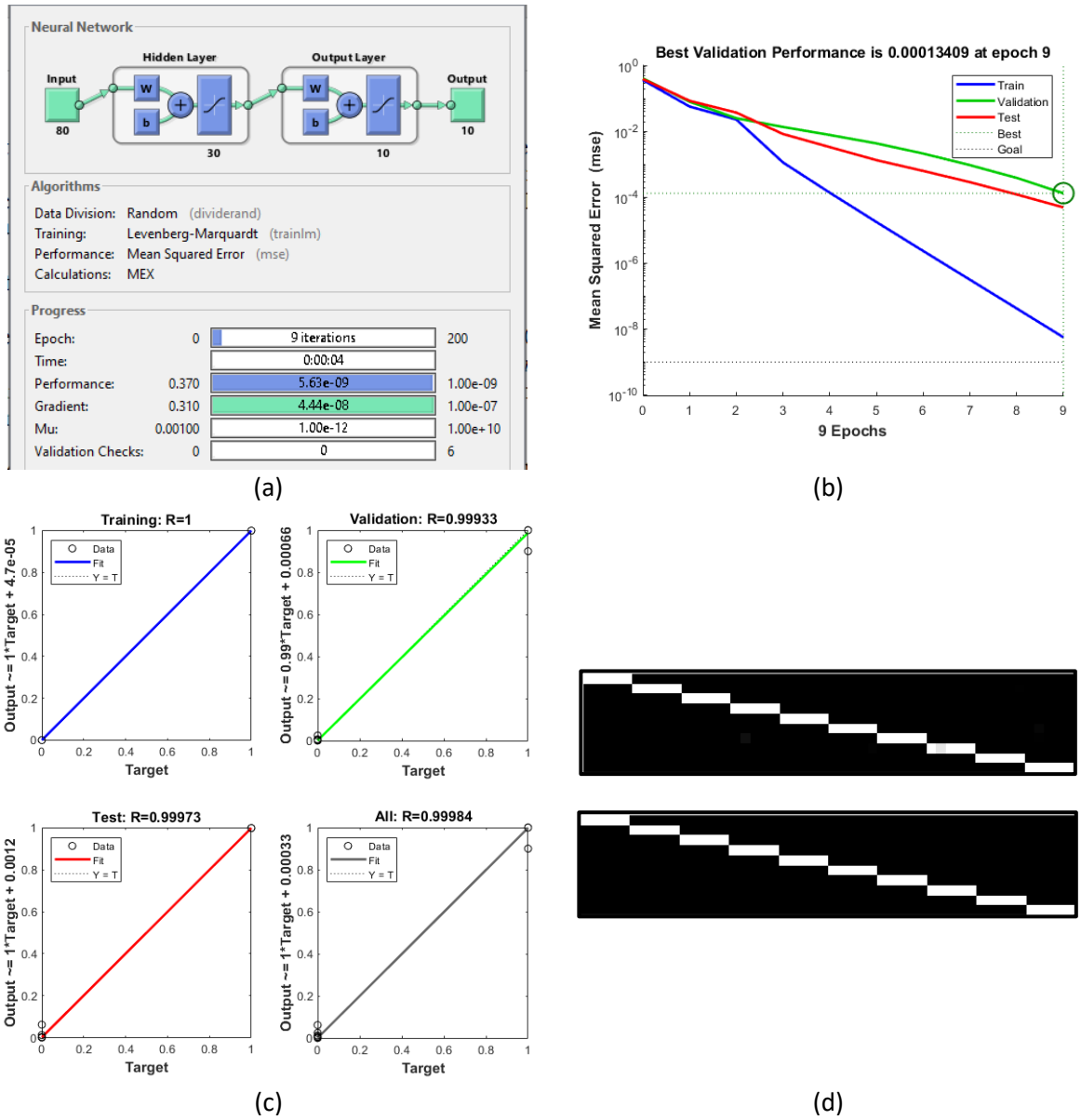
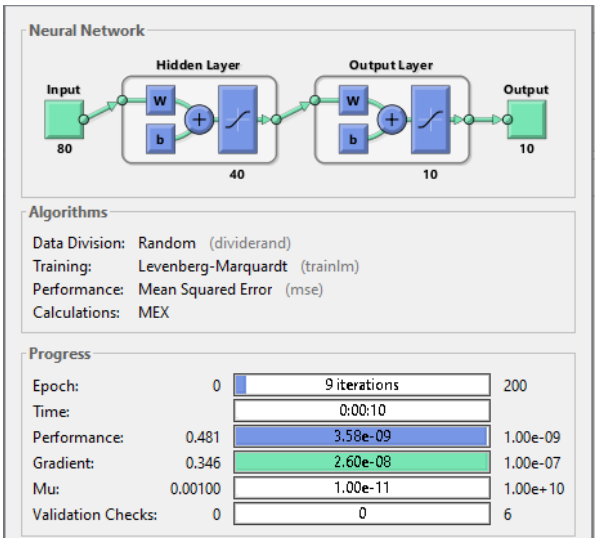
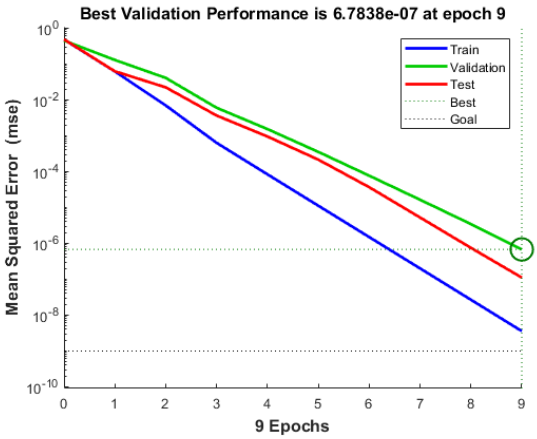


Figure 7: A 30 neuron single hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

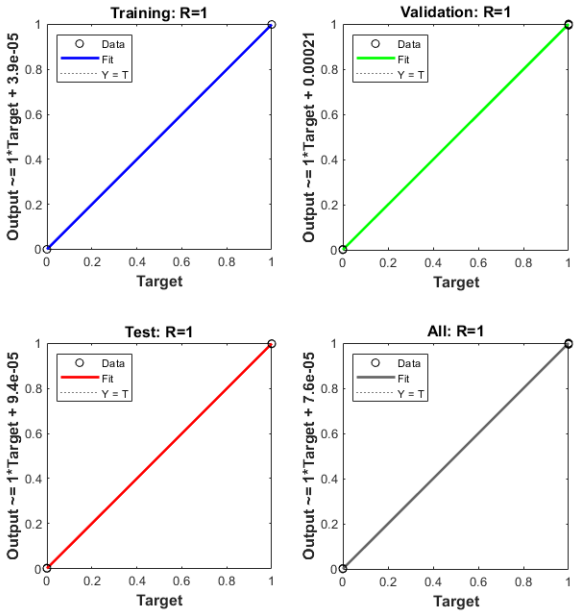
7.1.4 Case 4



(a)



(b)

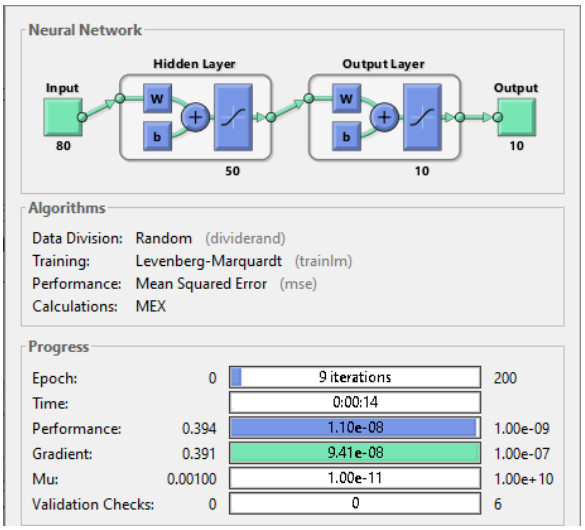


(c)

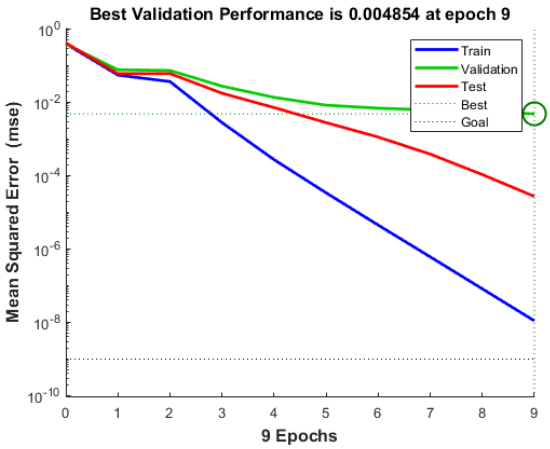
(d)

Figure 8: A 40 neuron single hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

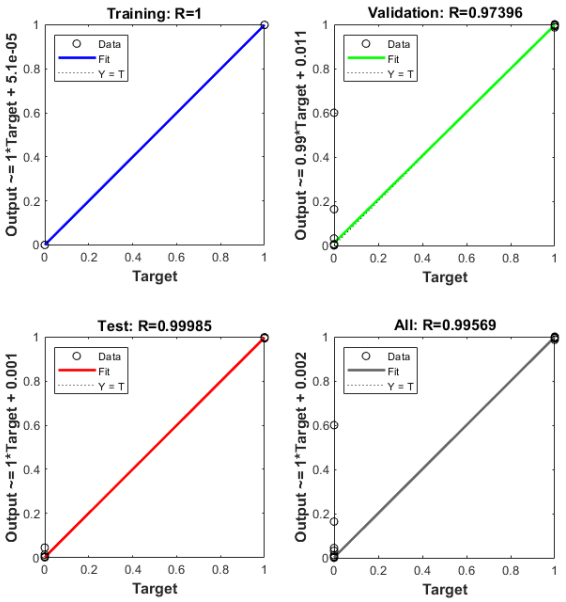
7.1.5 Case 5



(a)



(b)

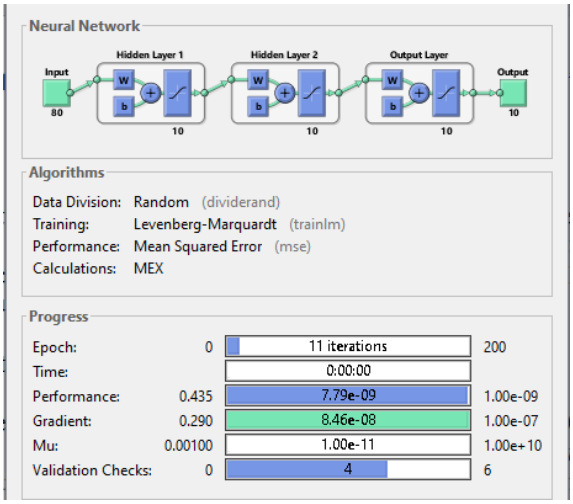


(c)

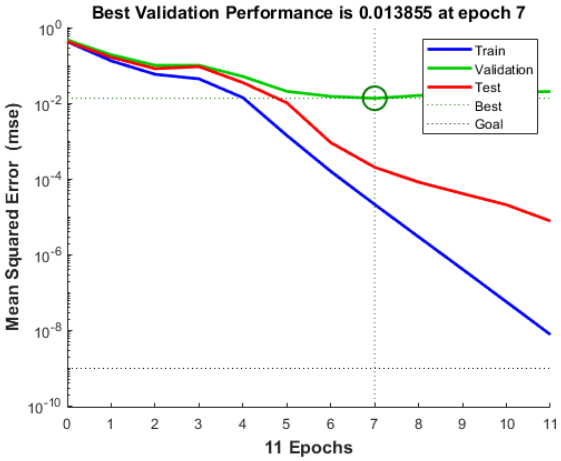
(d)

Figure 9: A 50 neuron single hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

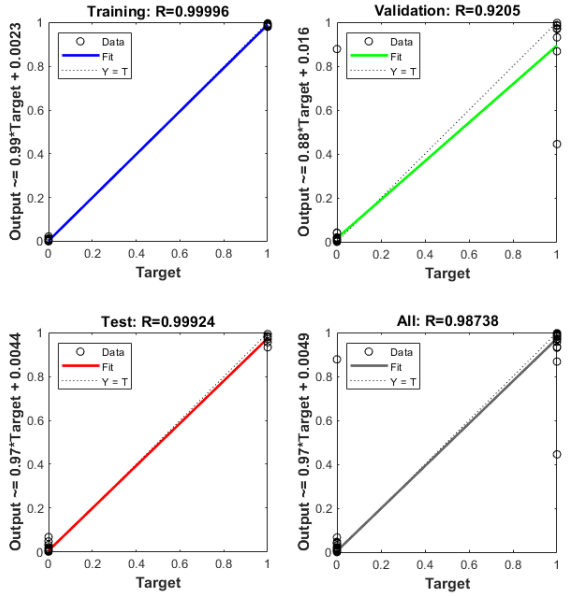
7.1.6 Case 6



(a)



(b)

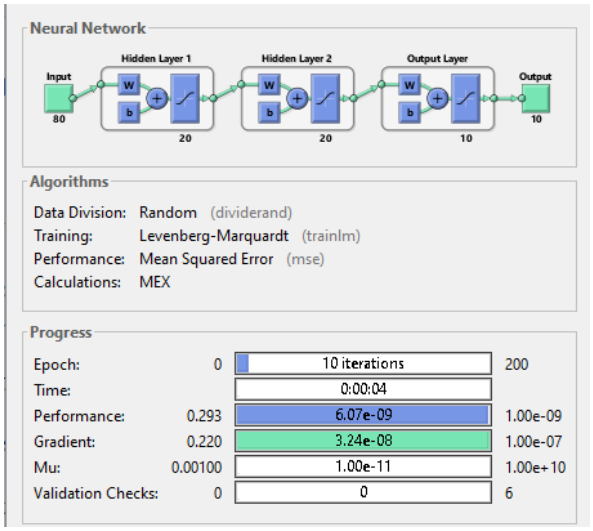


(c)

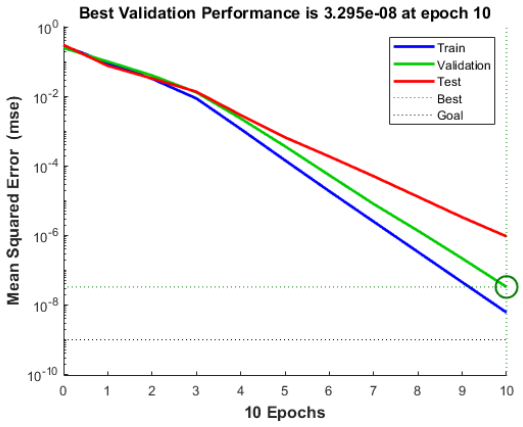
(d)

Figure 10: A 10 neuron double hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

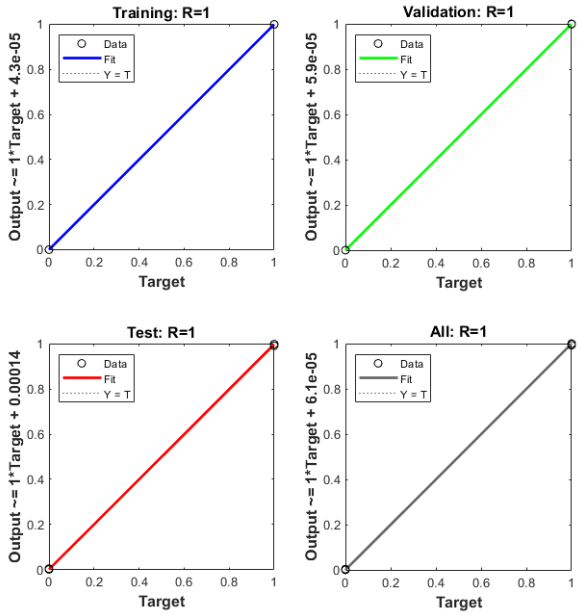
7.1.7 Case 7



(a)



(b)



(c)

(d)

Figure 11: A 20 neuron double hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.1.8 Case 8

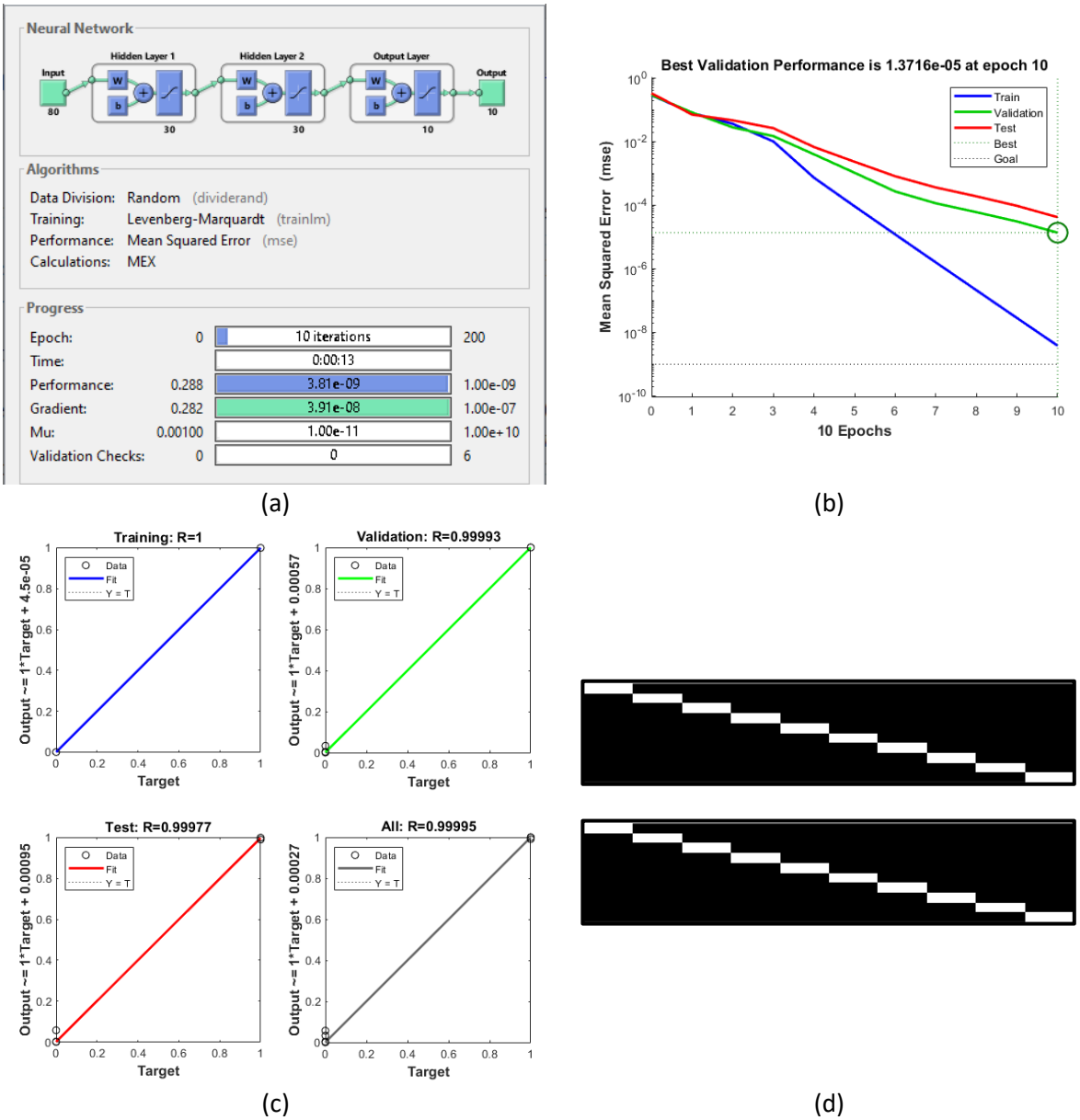
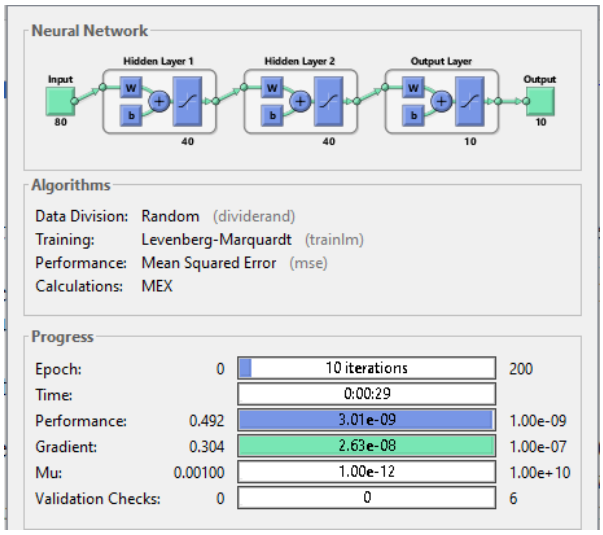
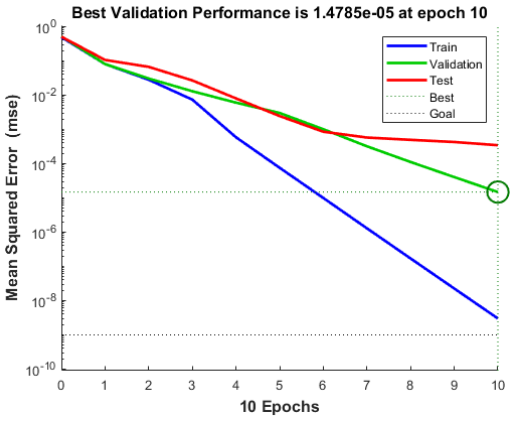


Figure 12: A 30 neuron double hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

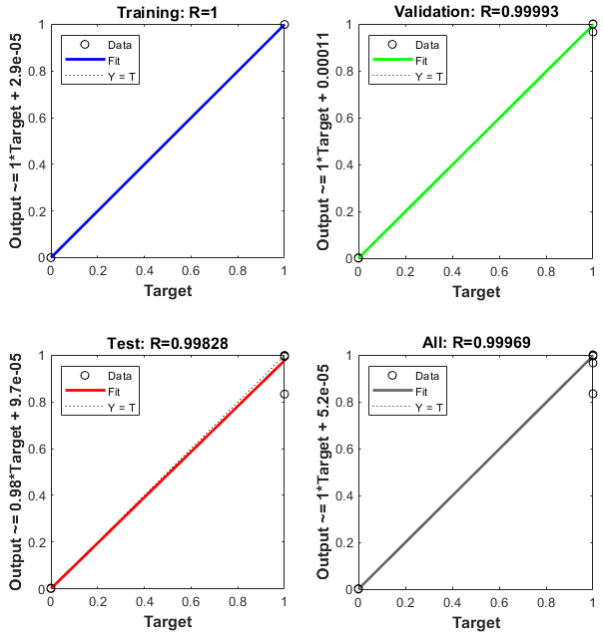
7.1.9 Case 9



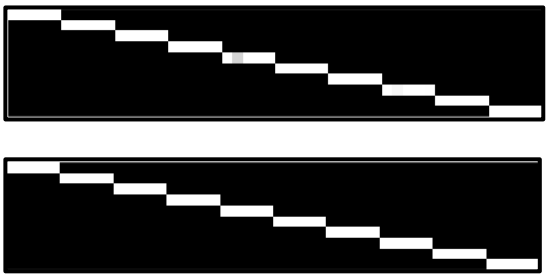
(a)



(b)



(c)



(d)

Figure 13: A 40 neuron double hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.1.10 Case 10

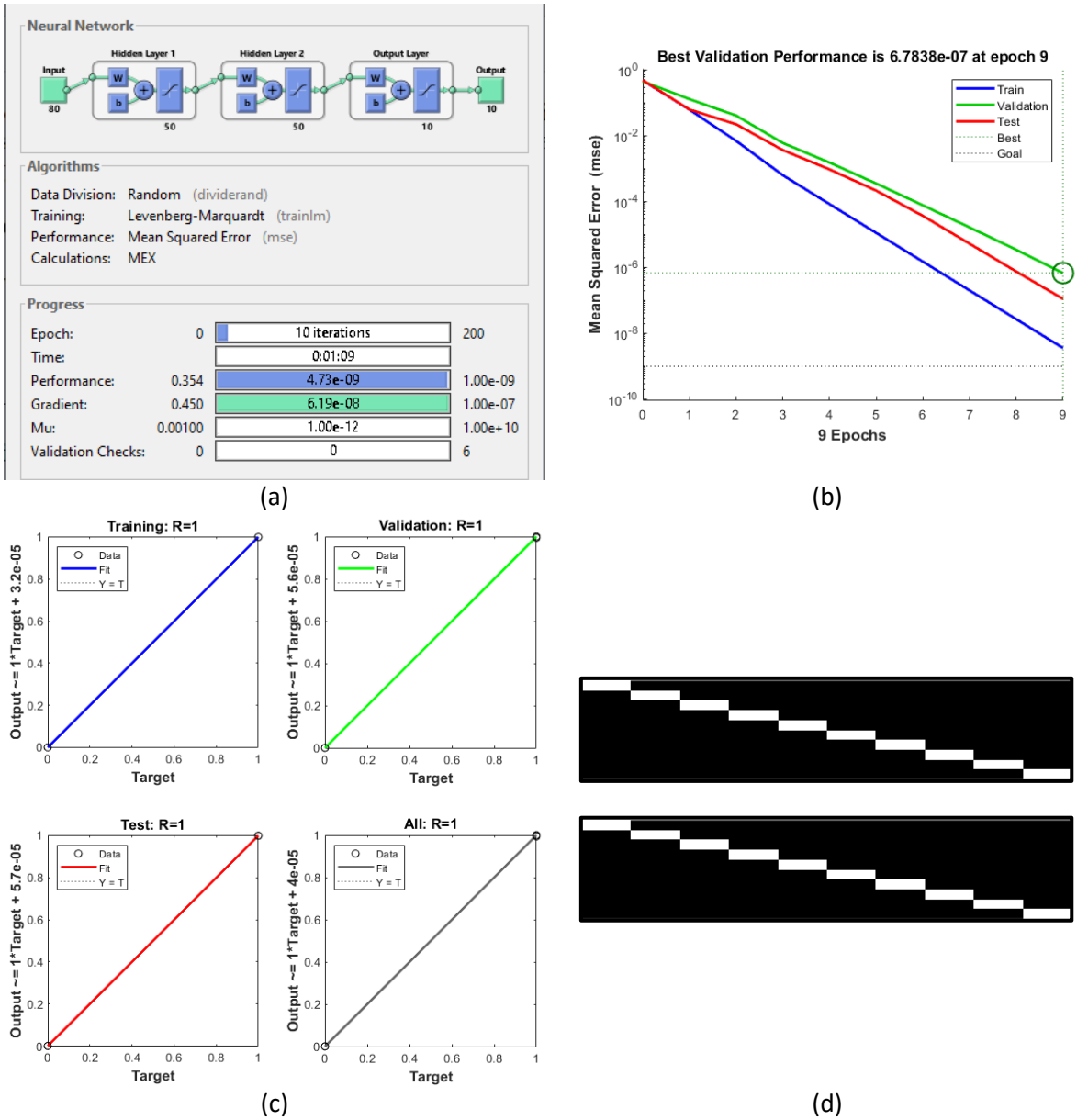
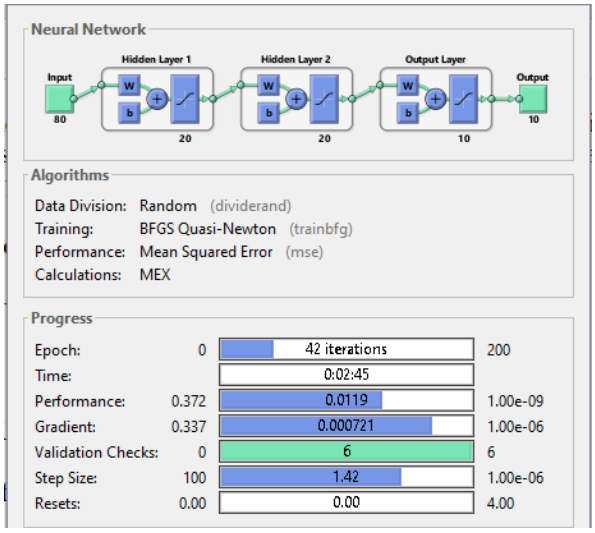


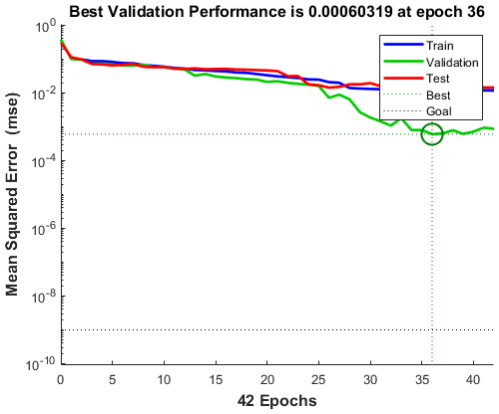
Figure 14: A 50 neuron double hidden layer neural network (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.2 Variation of Training Algorithms

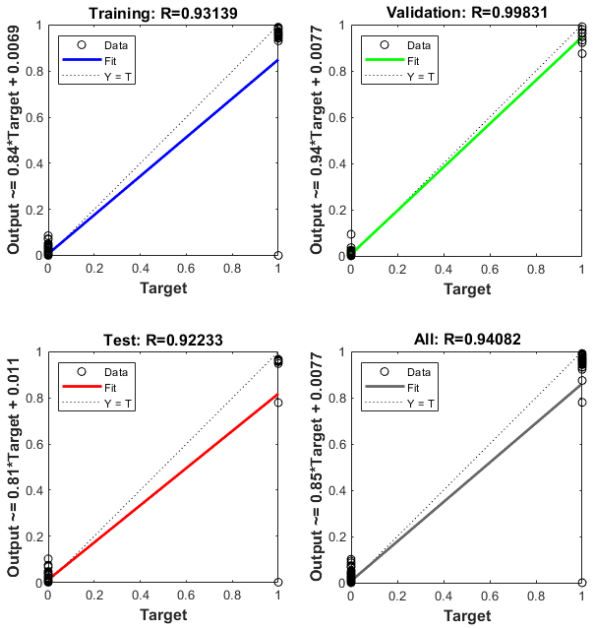
7.2.1 BFGS Quasi-Newton



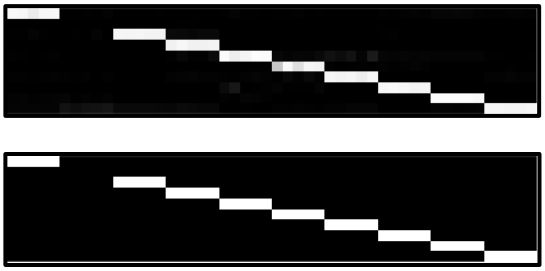
(a)



(b)



(c)



(d)

Figure 15: A 20 neuron double hidden layer neural network utilising BFGS Quasi-Newton training algorithm (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.2.2 Resilient Backpropagation

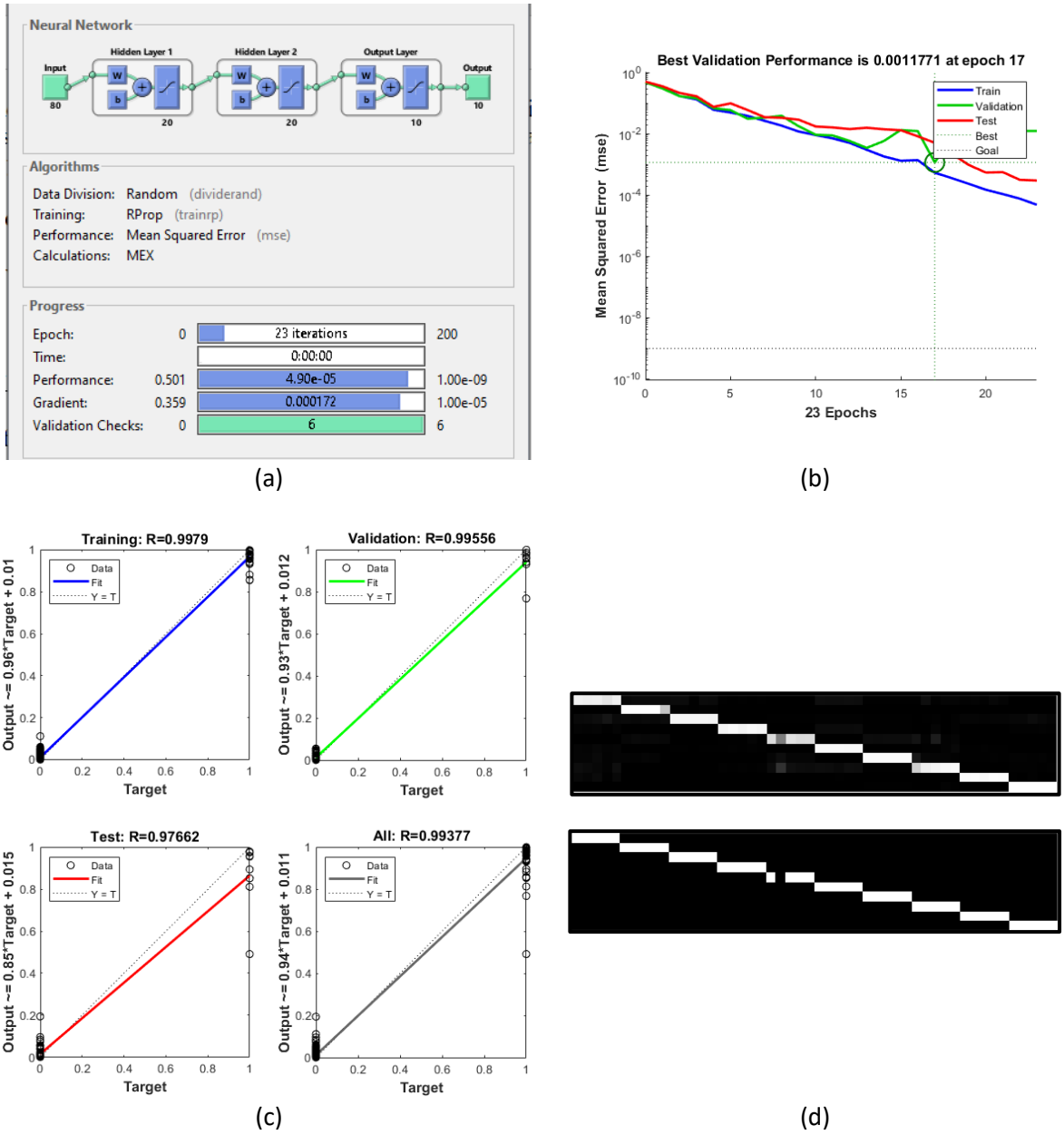
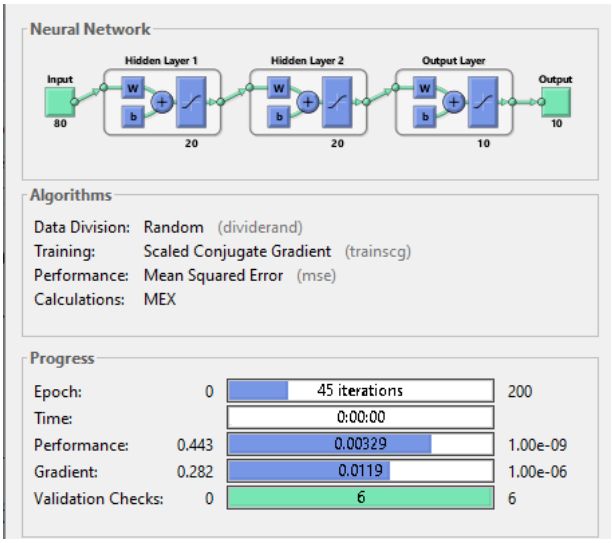
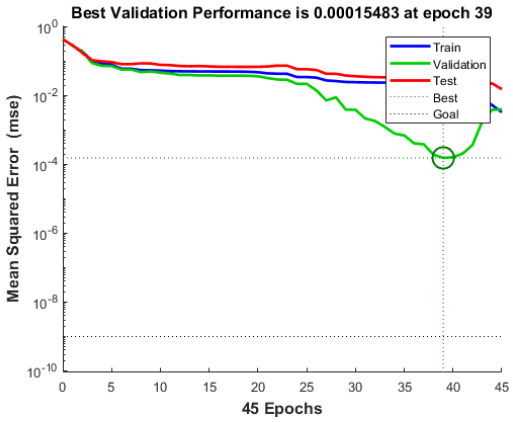


Figure 16: A 20 neuron double hidden layer neural network utilising Resilient Backpropagation training algorithm (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

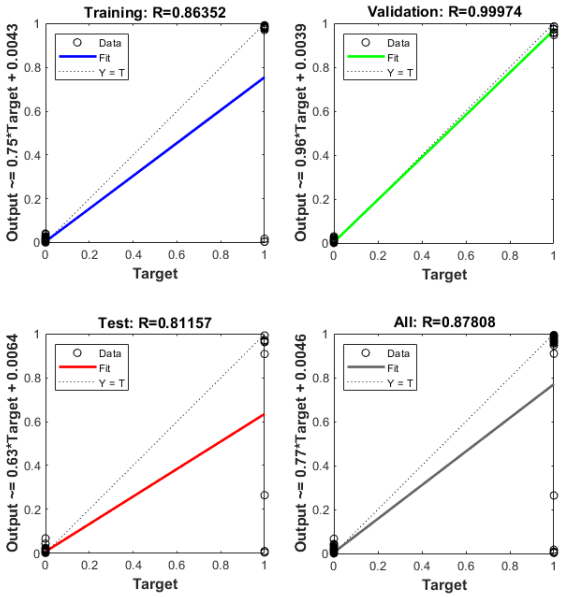
7.2.3 Scaled Conjugate Gradient



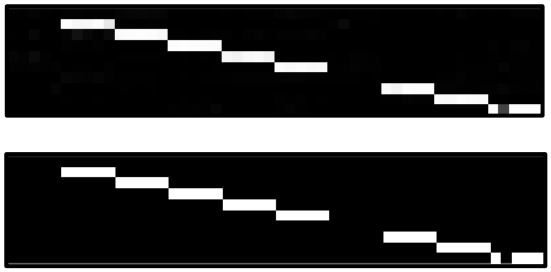
(a)



(b)



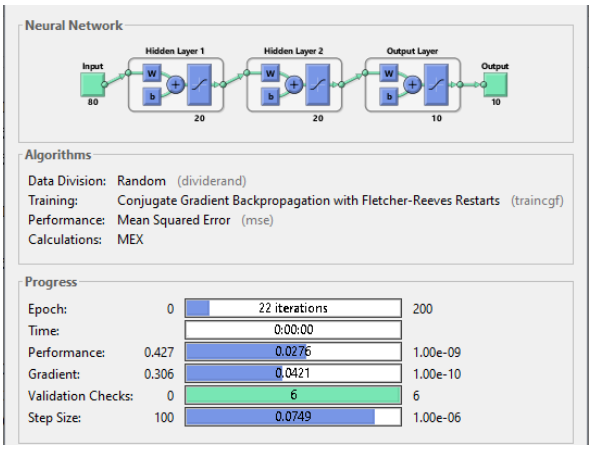
(c)



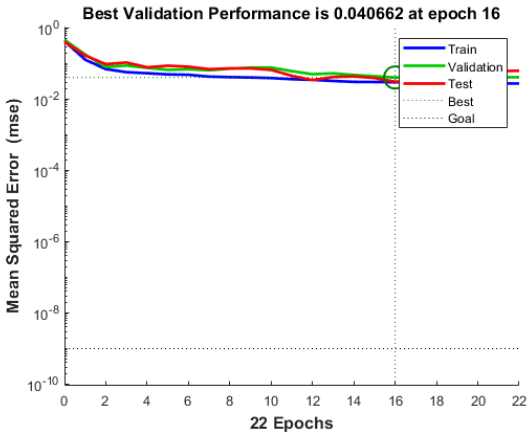
(d)

Figure 17: A 20 neuron double hidden layer neural network utilising Scaled Conjugate Gradient training algorithm (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

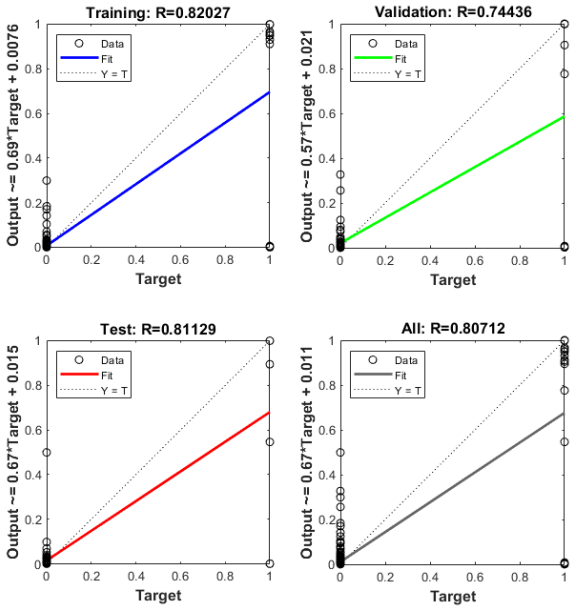
7.2.4 Fletcher-Powell Conjugate Gradient



(a)



(b)

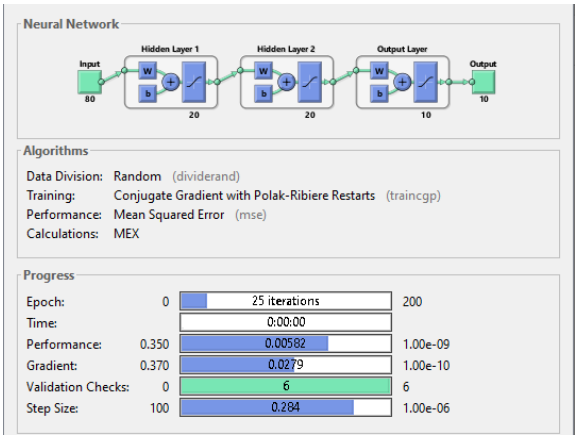


(c)

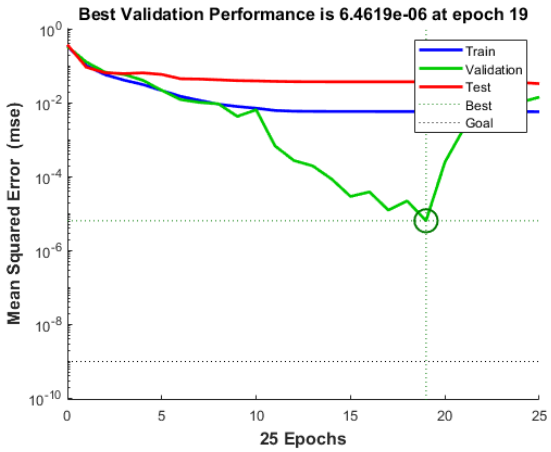
(d)

Figure 18: A 20 neuron double hidden layer neural network utilising Fletcher-Powell Conjugate Gradient training algorithm (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

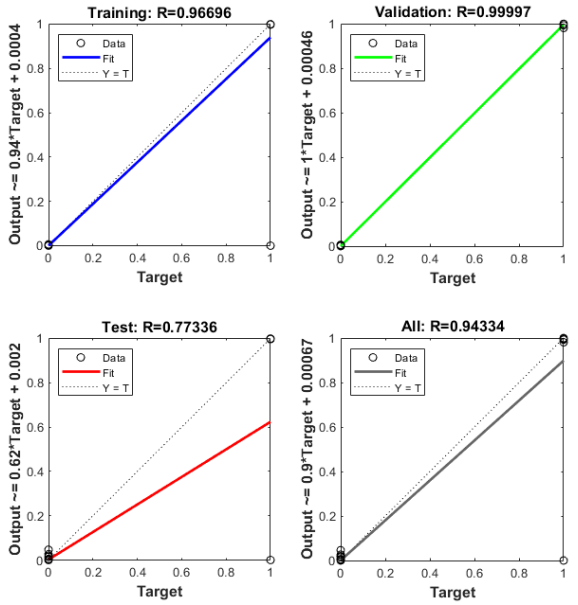
7.2.5 Polak-Ribiere Conjugate Gradient



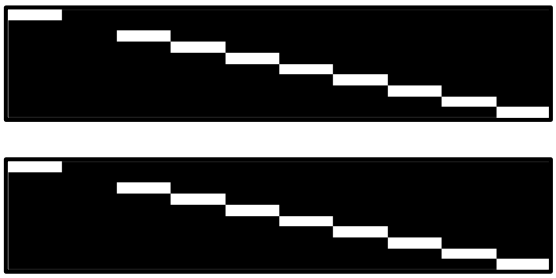
(a)



(b)



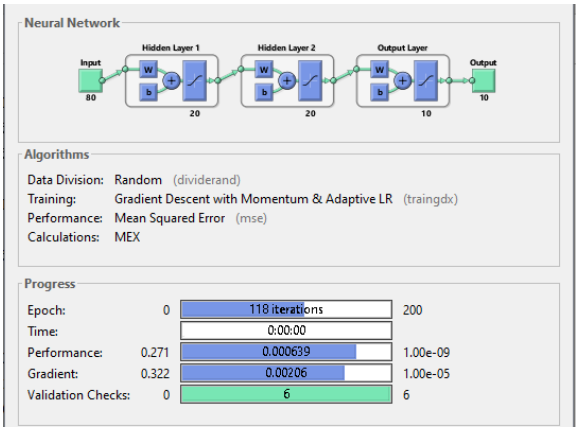
(c)



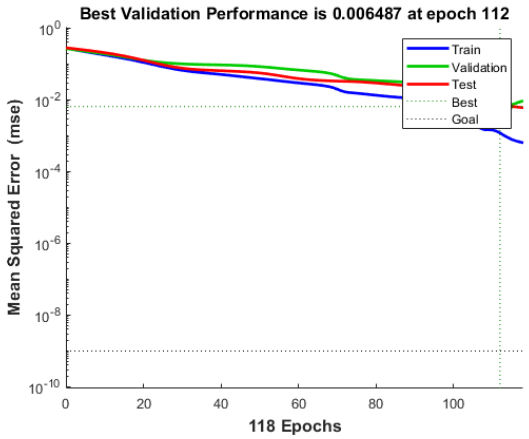
(d)

Figure 19: A 20 neuron double hidden layer neural network utilising Polak-Ribiere Conjugate Gradient training algorithm (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

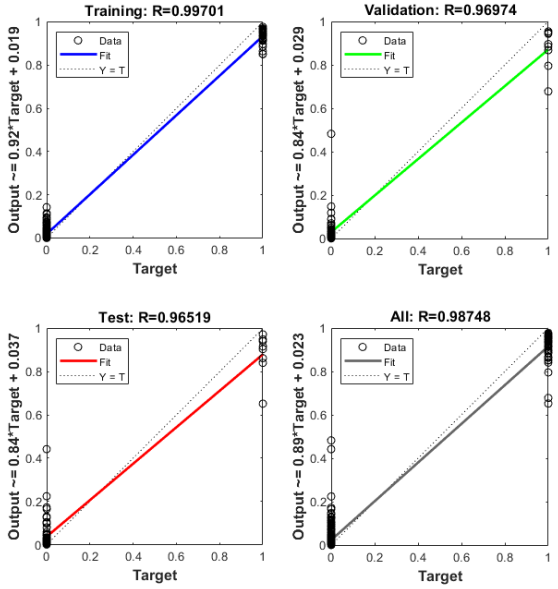
7.2.6 Variable Learning Rate Backpropagation



(a)



(b)



(c)

(d)

Figure 20: A 20 neuron double hidden layer neural network utilising Variable Learning Rate Backpropagation training algorithm (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.3 Variation of Transfer Function

7.3.1 Case 11

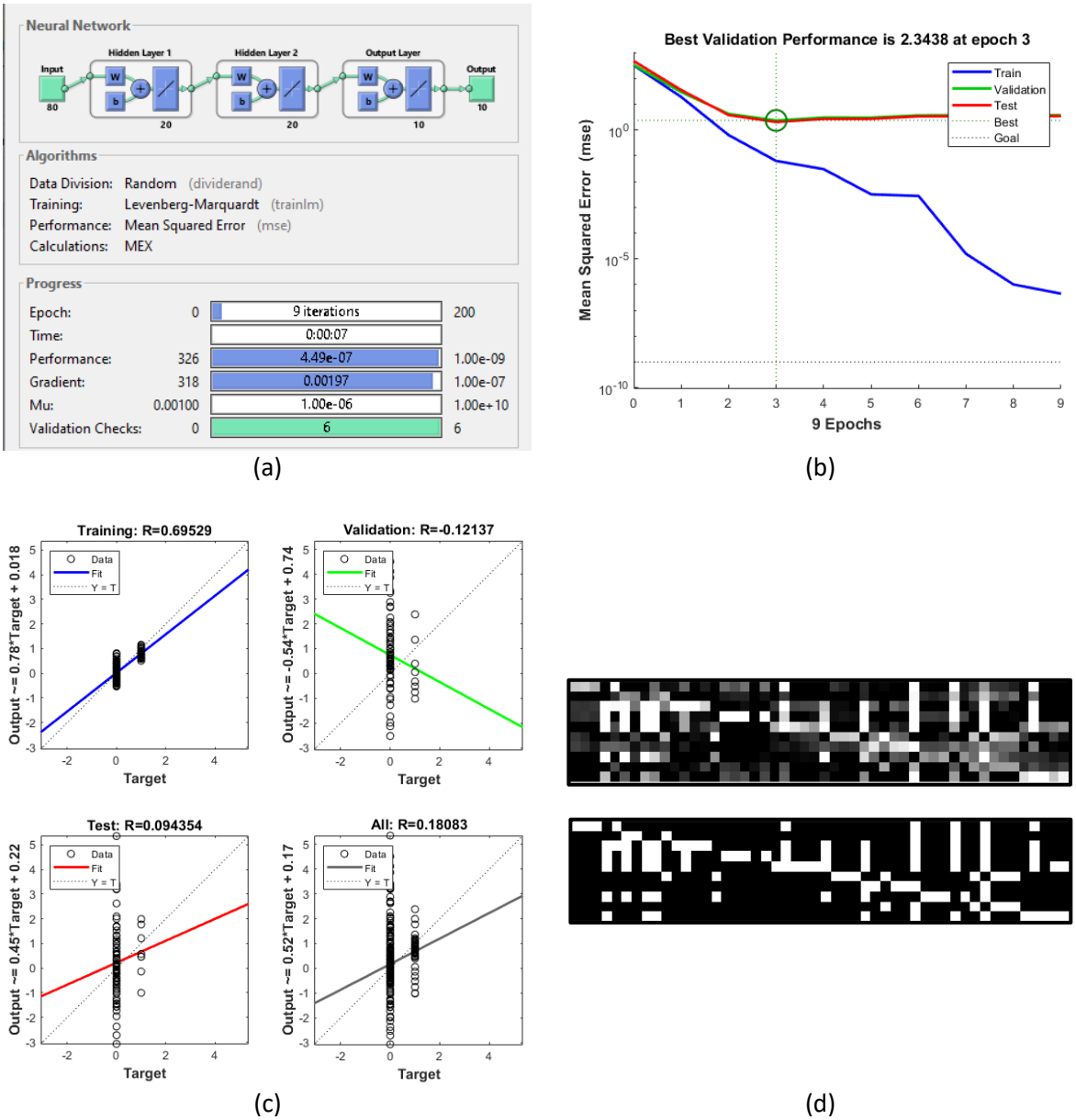
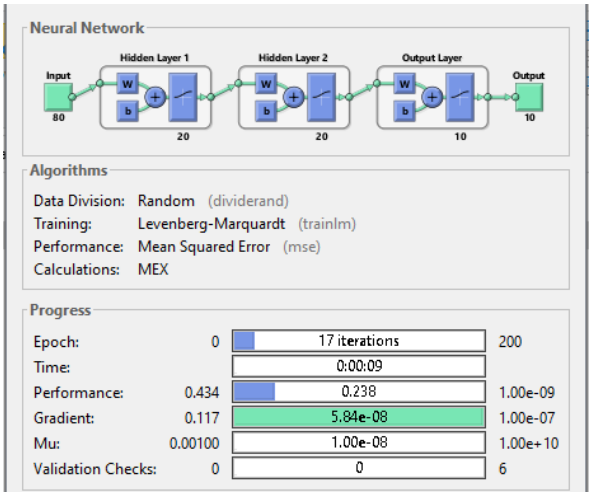
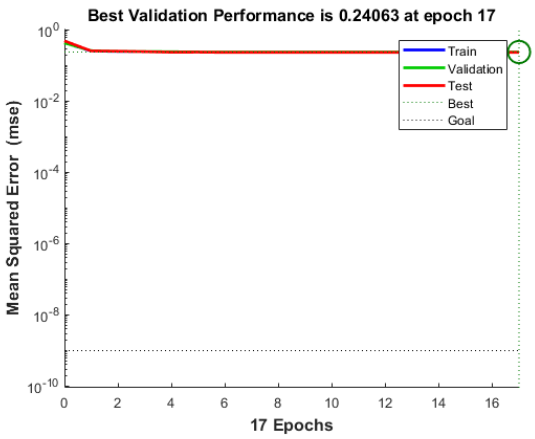


Figure 21: A 20 neuron double hidden layer neural network utilising Levenberg-Marquardt training algorithm with Purelin transfer function (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

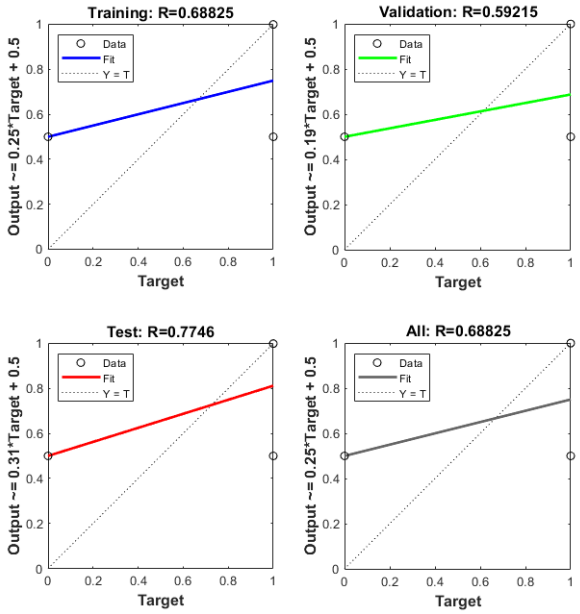
7.3.2 Case 12



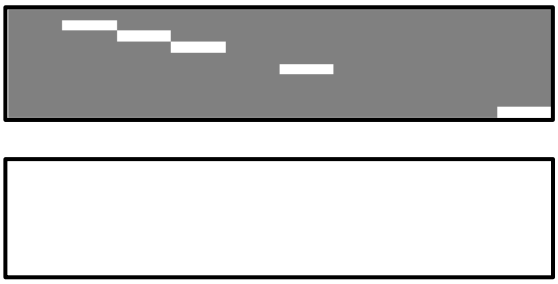
(a)



(b)



(c)



(d)

Figure 22: A 20 neuron double hidden layer neural network utilising Levenberg-Marquardt training algorithm with Logsig transfer function (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.3.4 Case 14

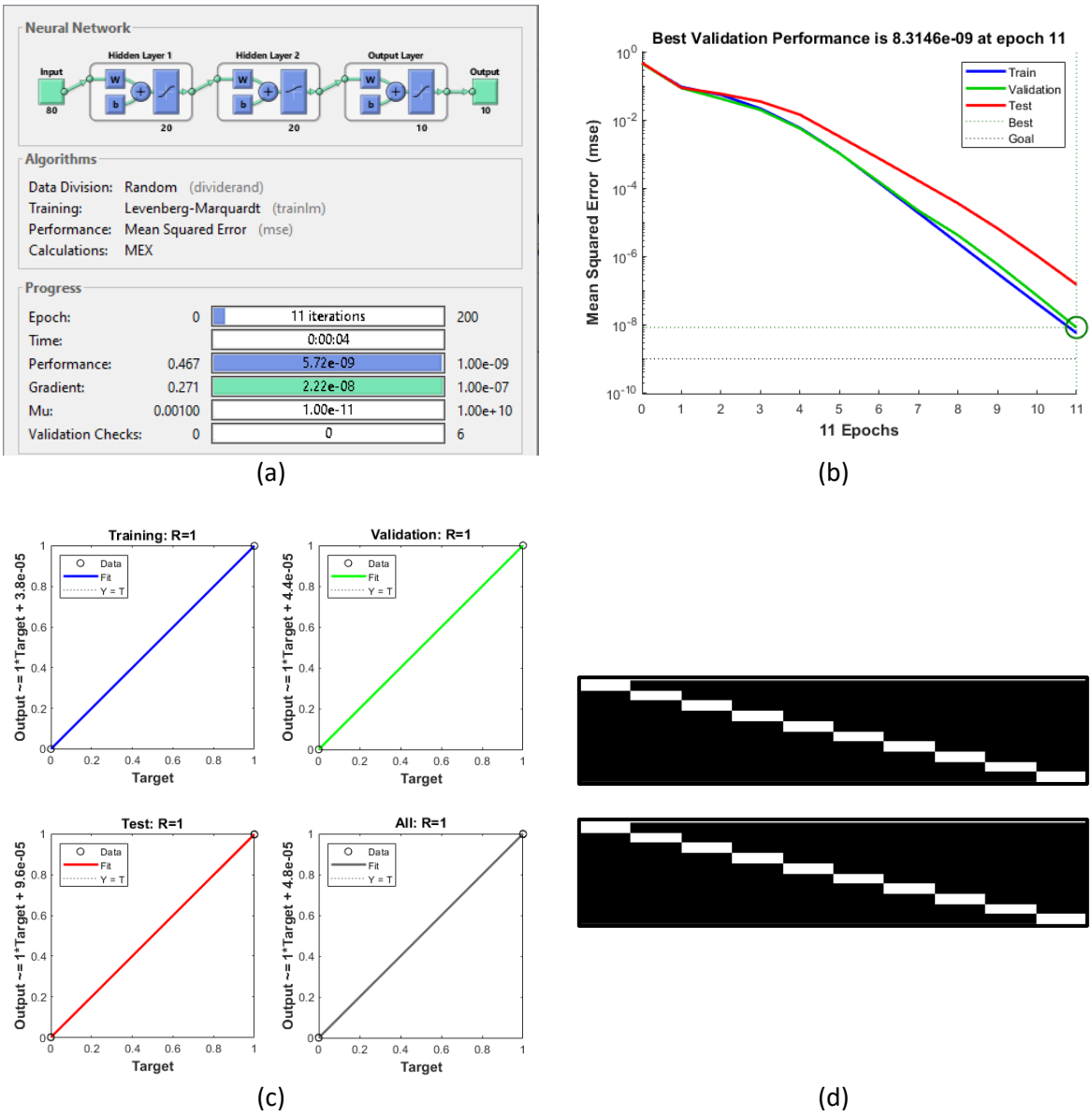


Figure 23: A 20 neuron double hidden layer neural network utilising Levenberg-Marquardt training algorithm with Tansig, Logsig and Tansig transfer function (a) training details, (b) MSE validation performance, (c) regression and (d) unrounded and rounded simulation output

7.4 Matlab Code for Random Noise Creation

```
function modified = randNoise(a)

pix_no = 4;
modified = a;
a_size = size(a);

for i = 1:pix_no
    row_rand = randi(a_size(1));
    col_rand = randi(a_size(2));

    if modified(row_rand, col_rand) == 1
        modified(row_rand, col_rand) = 0;
    else
        modified(row_rand, col_rand) = 1;
    end
end

end
```