

## Εισαγωγή

### **Νευρωνικά Δίκτυα**

Γενικότερα, τα νευρωνικά δίκτυα, γνωστά και ως τεχνητά νευρωνικά δίκτυα, αποτελούν υποσύνολο της μηχανικής μάθησης και βρίσκονται στην καρδιά των αλγορίθμων βαθιάς μάθησης. Το όνομα και η δομή τους είναι εμπνευσμένα από τον ανθρώπινο εγκέφαλο, μιμούμενοι τον τρόπο που λειτουργούν οι βιολογικοί νευρώνες.

Τα νευρωνικά δίκτυα βασίζονται σε δεδομένα εκπαίδευσης για να μάθουν και να βελτιώσουν την ακρίβειά τους με την πάροδο του χρόνου. Αξίζει να τονιστεί ότι αποτελούν ισχυρά εργαλεία στον τομέα της τεχνητής νοημοσύνης, επιτρέποντάς μας να ταξινομούμε και να ομαδοποιούμε δεδομένα με υψηλή ταχύτητα.

Στο μέρος A της άσκησης συναντούμε feedforward νευρωνικά δίκτυα. Τα περισσότερα βαθιά νευρωνικά δίκτυα είναι feedforward, που σημαίνει ότι ρέουν προς μία μόνο κατεύθυνση, από την είσοδο στην έξοδο. Ωστόσο, το μοντέλο είναι δυνατόν να εκπαιδευτεί μέσω του αλγορίθμου backpropagation, όπως έχουμε δει και στο θεωρητικό μέρος του μαθήματος.

Μεγάλη σύγχυση παρατηρείται στη χρήση των εννοιών Deep Learning και νευρωνικά δίκτυα. Ως αποτέλεσμα, αξίζει να σημειωθεί ότι το «βαθύ» στη βαθιά μάθηση αναφέρεται απλώς στο βάθος των στρωμάτων σε ένα νευρωνικό δίκτυο.(Μέρος B:Εφαρμογή Deep Network Designer)

Τα νευρωνικά δίκτυα είναι επιπροσθέτως ιδανικά για να βοηθήσουν τους ανθρώπους να λύσουν πολύπλοκα προβλήματα σε πραγματικές καταστάσεις. Μπορούν να μάθουν και να μοντελοποιήσουν τις σχέσεις μεταξύ εισόδων και εξόδων, να κάνουν γενικεύσεις, να αποκαλύπτουν σημαντικές πληροφορίες και να επιχειρούν διάφορες προβλέψεις. Υπάρχουν αρκετά παραδείγματα χρήσης νευρωνικών δικτύων, με στόχο τη λήψη απόφασης: Ανίχνευση απάτης με πιστωτική κάρτα, Ιατρική και διάγνωση ασθενειών, στοχευμένο μάρκετινγκ, πρόβλεψη ενεργειακής ζήτησης, αναγνώριση προσώπου και ομιλίας και πολλά άλλα.

Τα νευρωνικά δίκτυα οργανώνονται σε στρώματα. Το πρώτο στρώμα είναι η είσοδος, τα κρυφά στρώματα περιέχουν τους νευρώνες επεξεργασίας, και το τελευταίο είναι το στρώμα εξόδου. Οι νευρώνες ενός στρώματος συνδέονται με τους νευρώνες του επόμενου στρώματος, κάθε σύνδεση έχει ένα βάρος που ρυθμίζεται κατά την εκπαίδευση. Κάθε νευρώνας χρησιμοποιεί μια συνάρτηση ενεργοποίησης για να ρυθμίσει την έξοδό του. Η εκπαίδευση πραγματοποιείται σε επαναλαμβανόμενες εποχές με έναν καθορισμένο ρυθμό μάθησης που προσαρμόζεται κατάλληλα. Τα δεδομένα διαιρούνται σε σύνολα εκπαίδευσης, επικύρωσης και δοκιμής για να εκπαιδευτεί, να βελτιστοποιηθεί και να αξιολογηθεί το δίκτυο. Τέλος, ως μετρικές απόδοσης χρησιμοποιούμε την ακρίβεια, τον ρυθμό σφάλματος, την απώλεια, κ.ά., για να αξιολογήσουμε πόσο καλά λειτουργεί το δίκτυο. Συνοπτικά, η εφαρμογή των παραπάνω μπορεί να διαφέρει ανάλογα με τον τύπο του προβλήματος (ταξινόμηση, αναγνώριση εικόνων, πρόβλεψη χρονοσειρών).

## Μέρος Α

### Α1) Δημιουργία απλού νευρωνικού δικτύου, με χρήση της συνάρτησης network

#### Βήματα:

1. Συλλογή δεδομένων
2. Δημιουργία δικτύου
3. Διαμόρφωση δικτύου
4. Αρχικοποίηση βαρών και bias
5. Εκπαίδευση δικτύου
6. Validation του δικτύου
7. Χρήση του δικτύου

#### 1<sup>ο</sup> Παράδειγμα

Add-ons: Deep Learning Toolbox

Dataset 1: ionosphere.mat

Αρχείο: a1\_1.m

```
% BHMA 1: Συλλογή δεδομένων (Φόρτωση του dataset ionosphere)
% url: https://www.mathworks.com/help/stats/sample-data-sets.html
% Στο Command Window: load ionosphere.mat

% Dataset Description:
% X is a 351x34 real-valued matrix of predictors.
% Y is a categorical response.
% "b" for bad radar returns and "g" for good radar returns.

% Είδος προβλήματος: This is a binary classification problem.

% Για να μην εμφανίζεται error στη συνάρτηση train μετέπειτα:
numericLabels = double(strcmp(Y, 'g')); % 'g' -> 1, 'b' -> 0

% Ορισμός εισόδων και στόχων
inputs = X';
targets = numericLabels';

% BHMA 2: Δημιουργία νευρωνικού δικτύου
% Το νευρωνικό δίκτυο που δημιουργείται αποτελείται από 1 κρυφό
% στρώμα (με 10 νευρώνες)
hiddenLayerSize = 10;

% Το νευρωνικό δίκτυο είναι προς τα εμπρός
net = feedforwardnet(hiddenLayerSize);
```

```

% Συναρτήσεις μεταφοράς στα στρώματα του νευρωνίου
net.layers{1}.transferFcn = 'tansig'; % Στρώμα 1: υπερβολική
εφαπτομένη
net.layers{2}.transferFcn = 'logsig'; % Στρώμα 2: λογαριθμική

% BHMA 3: Διαμόρφωση δικτύου
% Το νευρωνικό δίκτυο θα διαιρεθεί στα 3 υποσύνολα που ακολουθούν.
net.divideFcn = 'dividerand';

% 70% for training.
% 15% to validate that the network is generalizing and to stop
training before overfitting.
% 15% to independently test network generalization.
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% BHMA 4: Αρχικοποίηση βαρών και bias
net = init(net);

% BHMA 5: Εκπαίδευση δικτύου
net.trainParam.epochs = 100; % αριθμός εποχών
net = train(net, inputs, targets);

% BHMA 6: Validation του δικτύου
outputs = net(inputs); % έξοδοι
errors = gsubtract(targets, outputs); % σφάλματα
performance = perform(net, targets, outputs); % απόδοση του δικτύου

% BHMA 7: Χρήση του δικτύου

% Δημιουργία πίνακα τυχαίων δεδομένων στο διάστημα (-1, 1)
% 34 αριθμοί λόγω του μεγέθους του πίνακα inputs
newData = 2 * rand(1, 34) - 1;

newData = newData'; % Μετατροπή σε στήλη

predictedClass = sim(net, newData); % πρόβλεψη της εξόδου για τα νέα
δεδομένα

figure(1)
sgtitle('outputs')
plot(outputs)

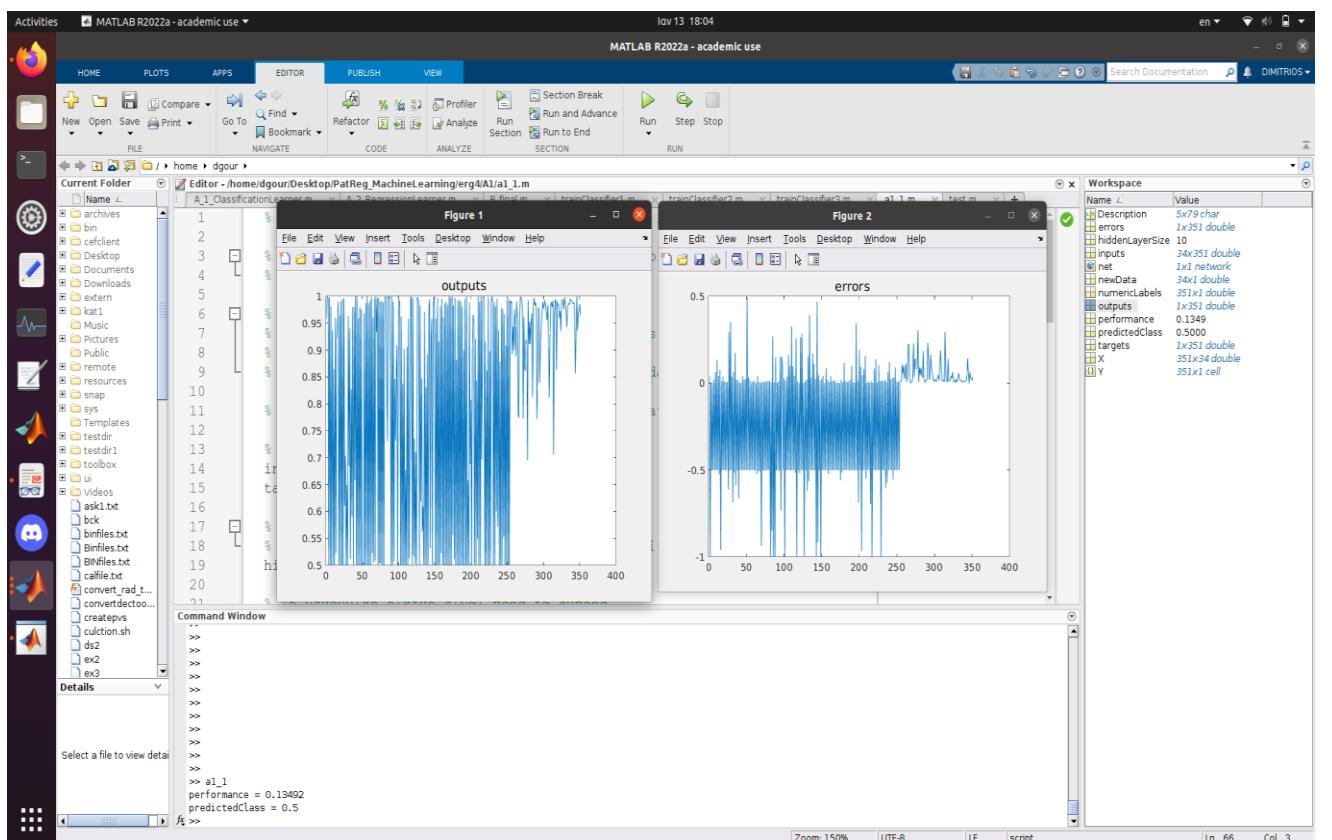
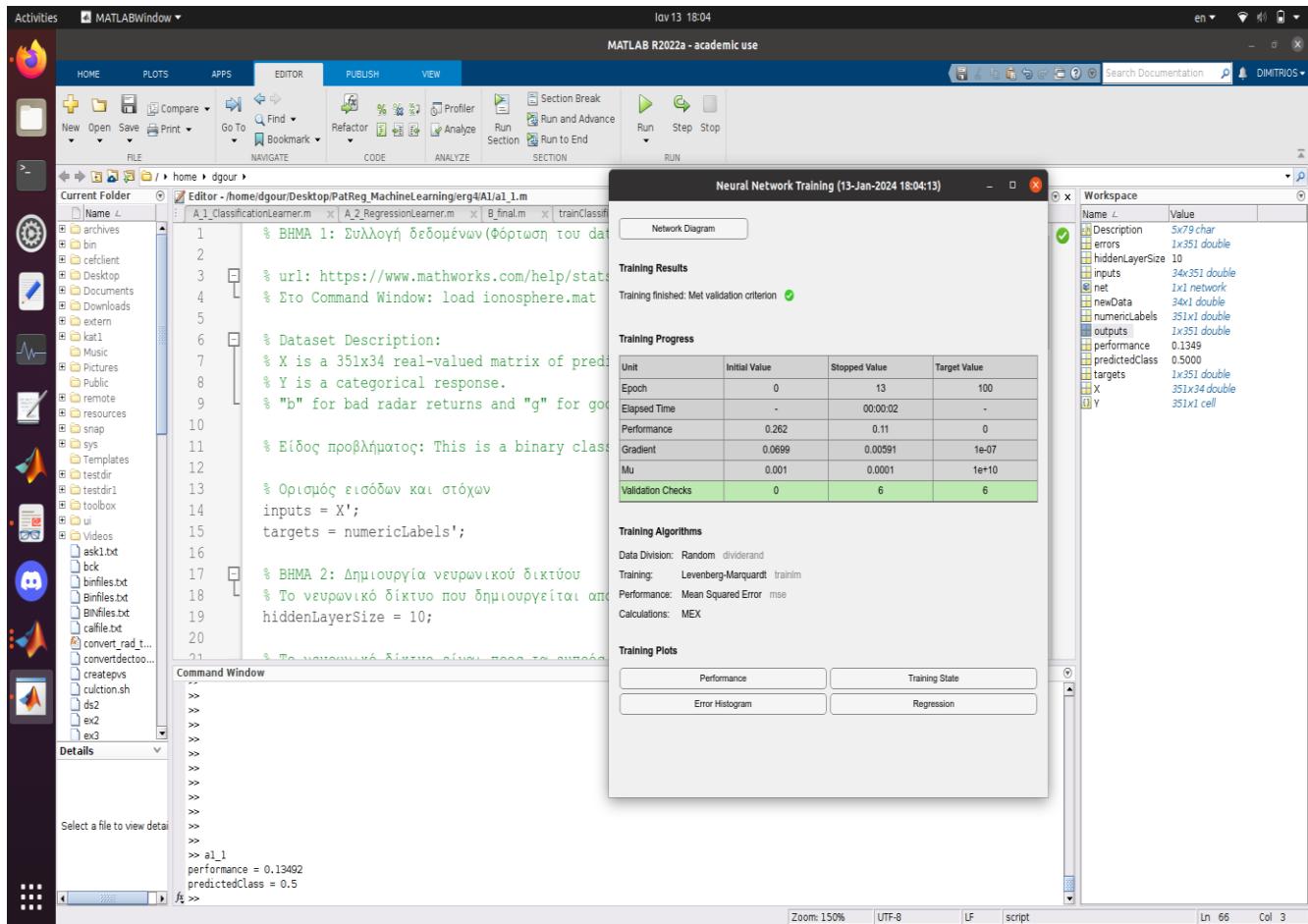
figure(2)
sgtitle('errors')
plot(errors)

disp(['performance = ', num2str(performance)]);

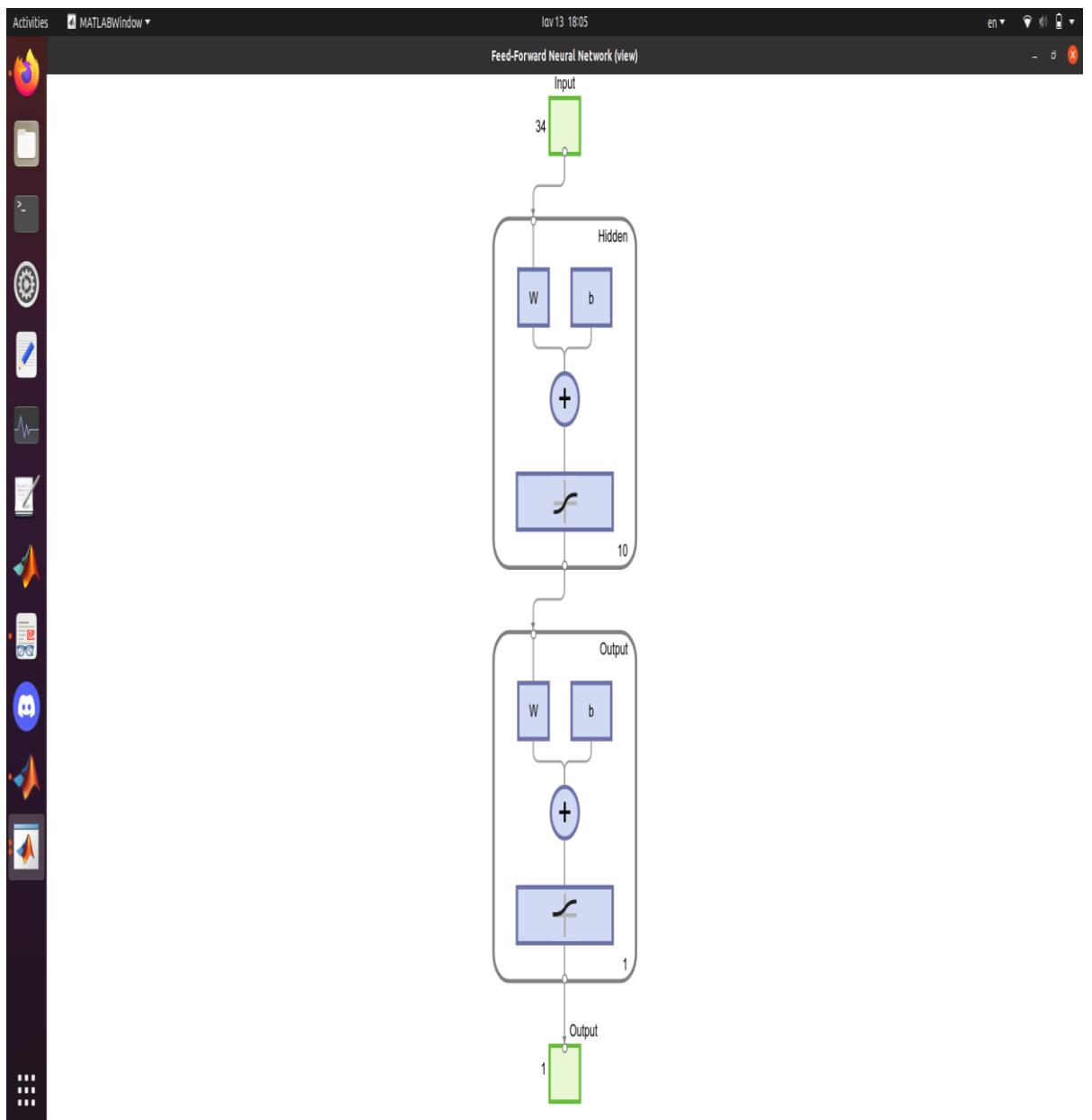
disp(['predictedClass = ', num2str(predictedClass)]);

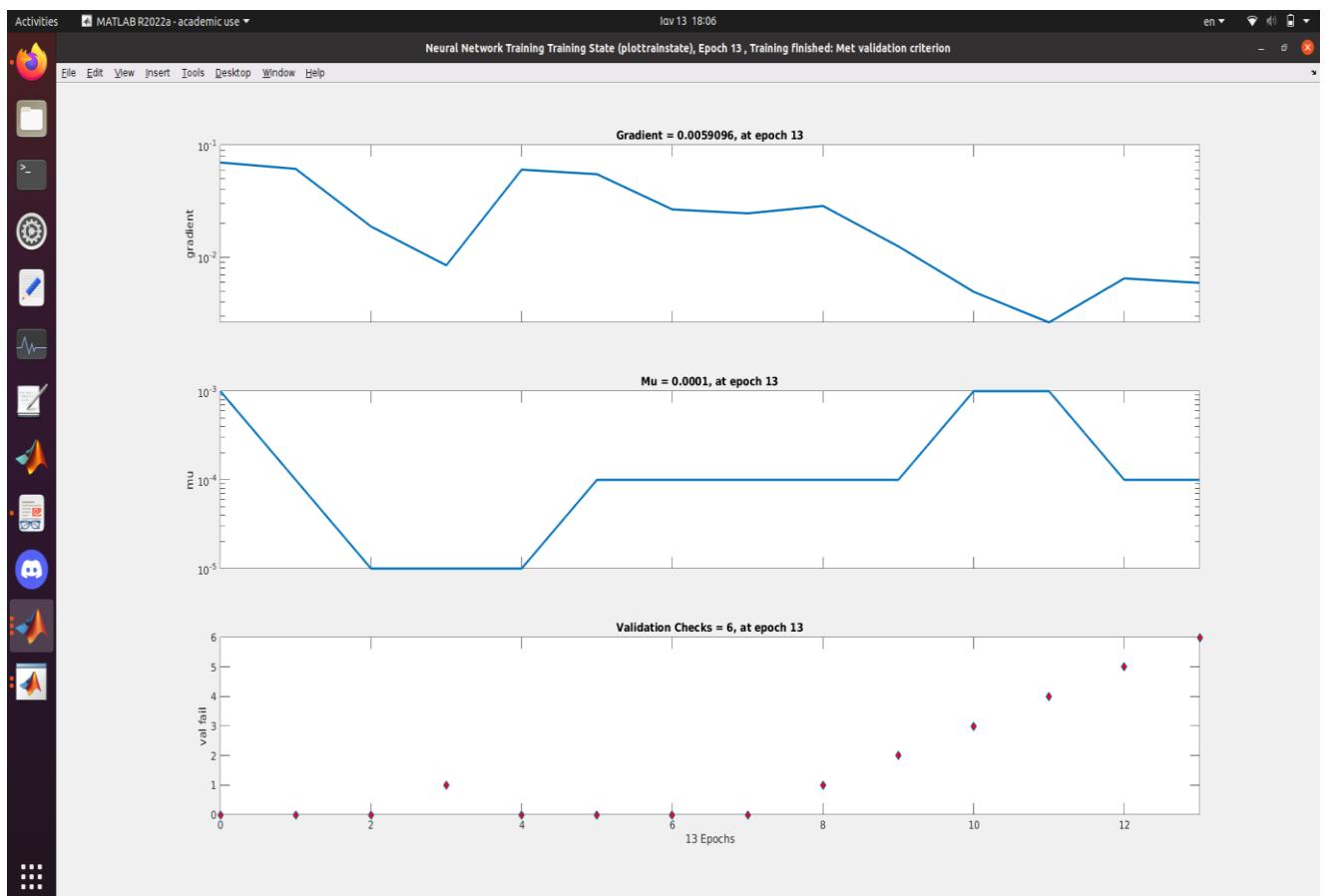
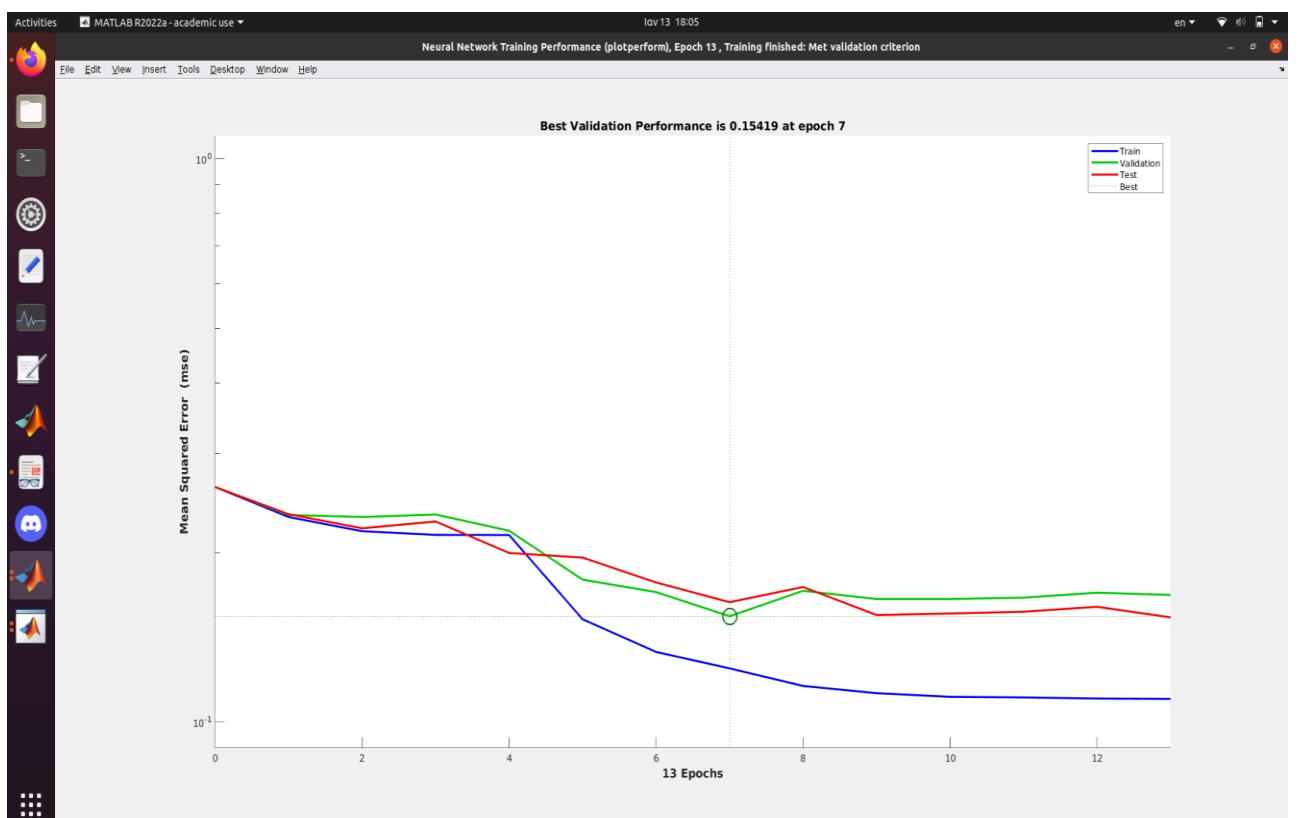
% Αποτελέσματα:
% performance = 0.13492 -> Μέσος όρος του τετραγωνικού σφάλματος ανά
% παράδειγμα
% predictedClass = 0.5

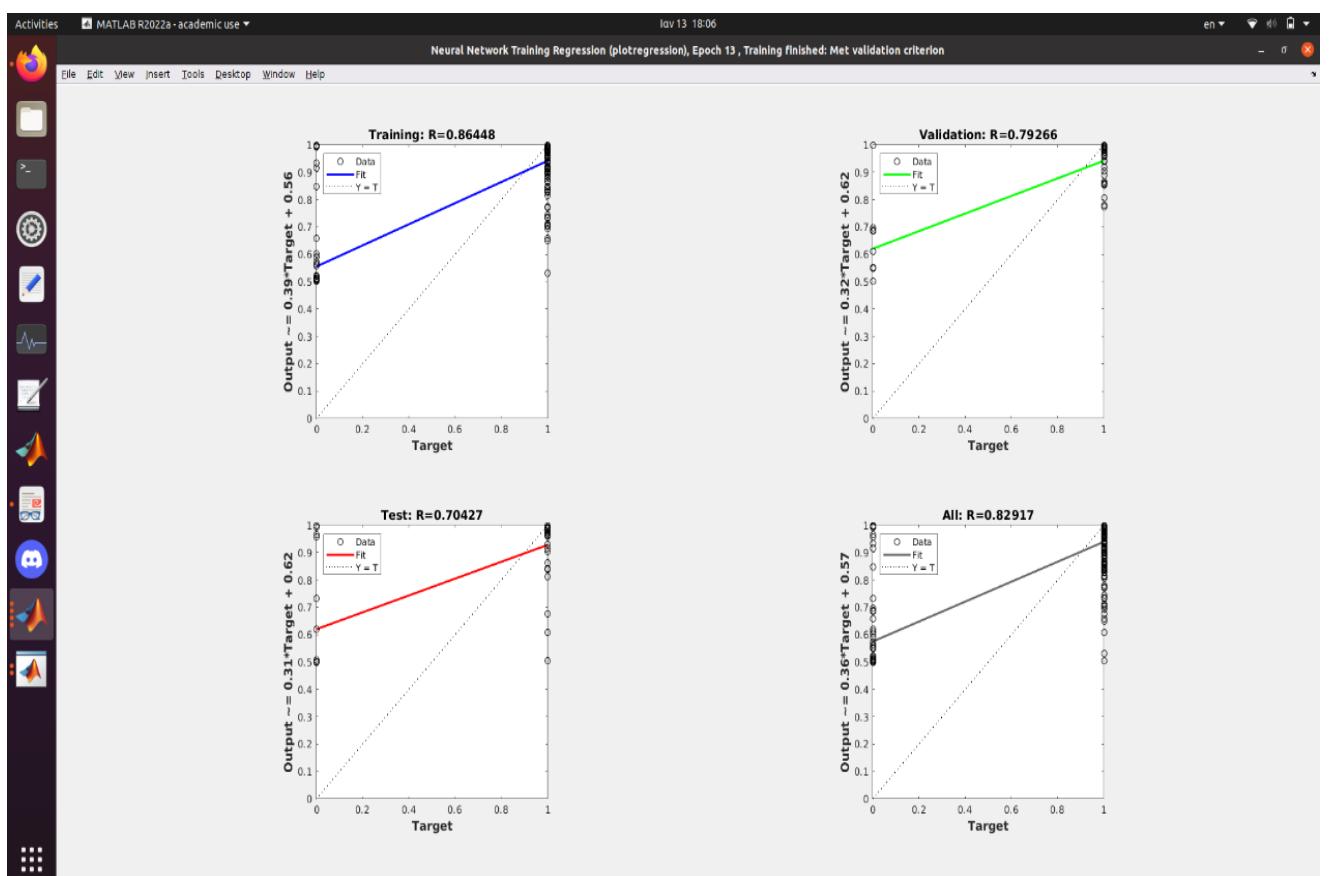
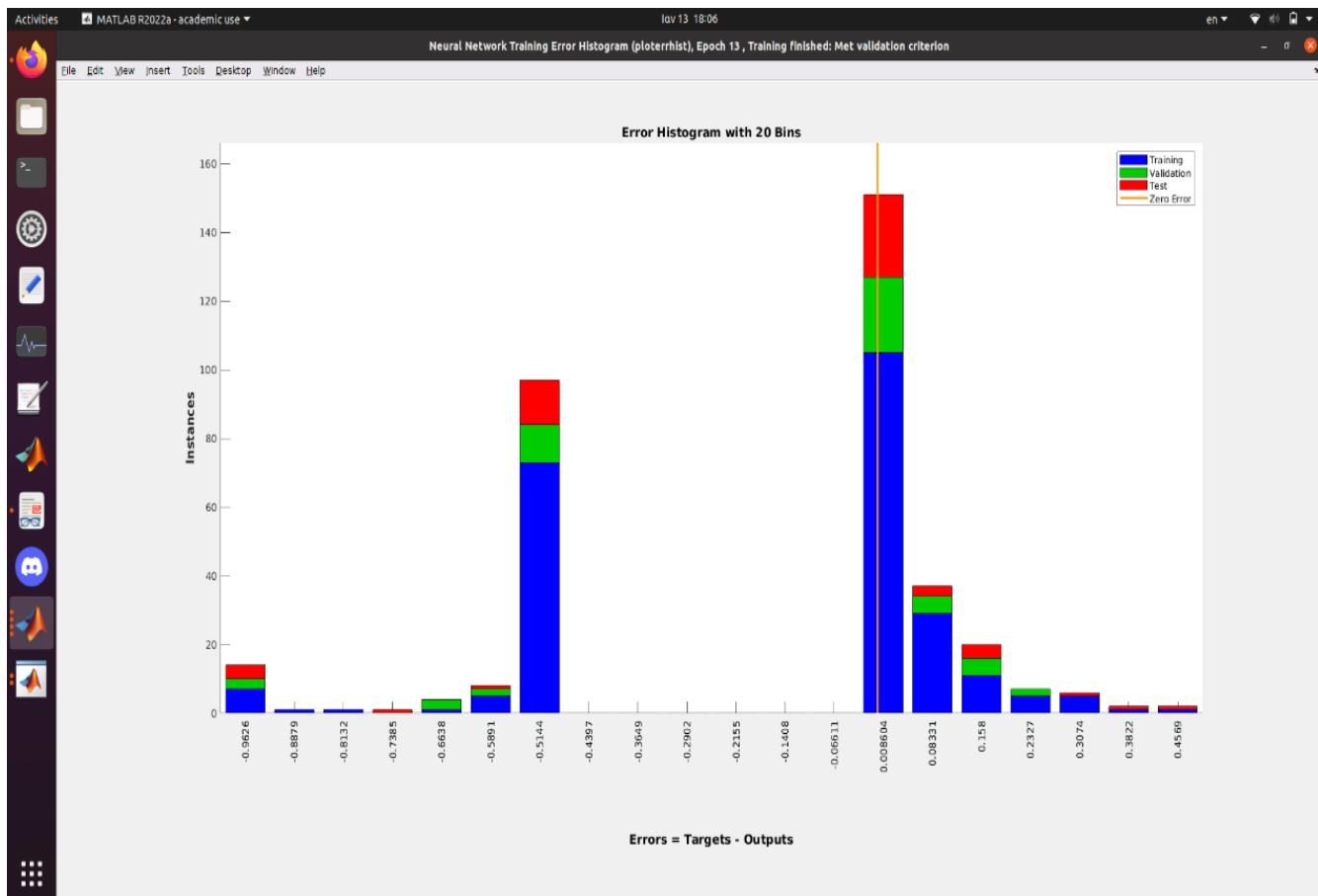
```



```
>>
>>
>>
>>
>>
>> a1_1
performance = 0.13492
predictedClass = 0.5
>>
```







### **Σχολιασμός αποτελεσμάτων:**

Ο κώδικας είναι αναλυτικά σχολιασμένος και χωρισμένος σε βήματα, ώστε να γίνεται κάθε στιγμή κατανοητή η λειτουργία που επιτελούμε. Επίσης, παραπάνω υπάρχουν screenshots από την εκτέλεση του προγράμματος.

Αναλυτικότερα, παρατηρούμε την εκπαίδευση των δεδομένων, και σημαντικά στοιχεία για τις εξόδους και τα errors. Επιπροσθέτως, παρουσιάζουμε το νευρωνικό δίκτυο που υλοποιήσαμε με τη συνάρτηση network,έχοντας ως συναρτήσεις μεταφοράς τις εξής: κρυφό στρώμα: υπερβολική εφαπτομένη, στρώμα εξόδου: λογαριθμική.

Έπειτα, παρουσιάζουμε το διάγραμμα της Best Validation Performance, που σημειώνεται την 7<sup>η</sup> εποχή εκπαίδευσης. Στο διάγραμμα, όπως φαίνεται και στο legend, υπάρχουν τα εξής: train, validation, test, best. Στη συνέχεια, εμφανίζονται διαγράμματα που αφορούν την κλίση αλλά και ελέγχους επικύρωσης. Επίσης, παρουσιάζεται ιστόγραμμα(με 20 bins) που δείχνει το σημείο όπου μηδενίζονται τα σφάλματα, και το ποσοστό των training, validation και test συνόλων ως instances.

Τέλος, μετά το τέλος της εκπαίδευσης του δικτύου παρουσιάζονται 4 διαγράμματα, στο ίδιο παράθυρο. Στην αρχή του κώδικα έχουν οριστεί οι στόχοι(targets) του δικτύου. Σε αυτό το σημείο παρουσιάζουμε τη σχέση output-target, για την εκπαίδευση, το testing, και το validation. Ενώ, στο τελευταίο διάγραμμα παρουσιάζουμε την παραπάνω σχέση και για τις 3 παραπάνω έννοιες σε συνδυασμό.

## 2<sup>ο</sup> Παράδειγμα

**Dataset 2:** cho\_dataset

**Αρχείο:** al\_2.m

```
% BHMA 1: Συλλογή δεδομένων (Φόρτωση του dataset cho_dataset)
% Δεδομένα που σχετίζονται με τη χοληστερόλη.

% url: https://www.mathworks.com/help/deeplearning/gs/sample-data-
sets-for-shallow-neural-networks.html
% Στο Command Window: load cho_dataset

% Είδος προβλήματος: Function Fitting, Function approximation and
Curve fitting.
% Function fitting is the process of training a neural network on a
% set of inputs in order to produce an associated set of target
outputs.
% Once the neural network has fit the data, it forms a
generalization of
% the input-output relationship and can be used to generate outputs
for
% inputs it was not trained on.

% Ορισμός εισόδων και στόχων
inputs = choInputs(1:3, :); % Για να μην εμφανίζεται error στη
συναρτηση train μετέπειτα
targets = choTargets;

% Τα choInputs, choTargets έχουν ίδιο αριθμό στηλών (264).
% Στο 1ο παράδειγμα παρατηρήσαμε ίδιο αριθμό γραμμών.

% BHMA 2: Δημιουργία νευρωνικού δικτύου
% Το νευρωνικό δίκτυο που δημιουργείται αποτελείται από 2 κρυφά
στρώματα
% (με 20 και 10 νευρώνες το καθένα αντίστοιχα)
hiddenLayerSize = [20,10];

% Το νευρωνικό δίκτυο είναι προς τα εμπρός
net = feedforwardnet(hiddenLayerSize);

% Συναρτήσεις μεταφοράς στα στρώματα του νευρωνίου
net.layers{1}.transferFcn = 'tansig'; % Στρώμα 1: υπερβολική
εφαπτομένη
net.layers{2}.transferFcn = 'logsig'; % Στρώμα 2: λογαριθμική
net.layers{3}.transferFcn = 'logsig'; % Στρώμα 3: λογαριθμική

% BHMA 3: Διαμόρφωση δικτύου
% Το νευρωνικό δίκτυο θα διαιρεθεί στα 3 υποσύνολα που ακολουθούν.
net.divideFcn = 'dividerand';

% 70% for training.
% 15% to validate that the network is generalizing and to stop
training before overfitting.
% 15% to independently test network generalization.
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% BHMA 4: Αρχικοποίηση βαρών και bias
```

```

net = init(net);

% BHMA 5: Εκπαίδευση δικτύου
% Αριθμός εποχών: Αριθμός των επαναλήψεων που το νευρωνικό δίκτυο θα
% εκπαιδευτεί.
% Ένας μικρότερος αριθμός εποχών σημαίνει λιγότερες επαναλήψεις της
% διαδικασίας εκπαίδευσης.
net.trainParam.epochs = 20; % στο προηγούμενο παράδειγμα 100
net = train(net, inputs, targets);

% BHMA 6: Validation του δικτύου
outputs = net(inputs); % έξοδοι
errors = gsubtract(targets, outputs); % σφάλματα
performance = perform(net, targets, outputs); % απόδοση του δικτύου

% BHMA 7: Χρήση του δικτύου

% Δημιουργία πίνακα τυχαίων δεδομένων στο διάστημα (-1, 1)
newData = 2 * rand(3, 1) - 1;

predictedClass = sim(net, newData); % πρόβλεψη της εξόδου για τα νέα
% δεδομένα

figure(1)
sgtitle('outputs')
plot(outputs)

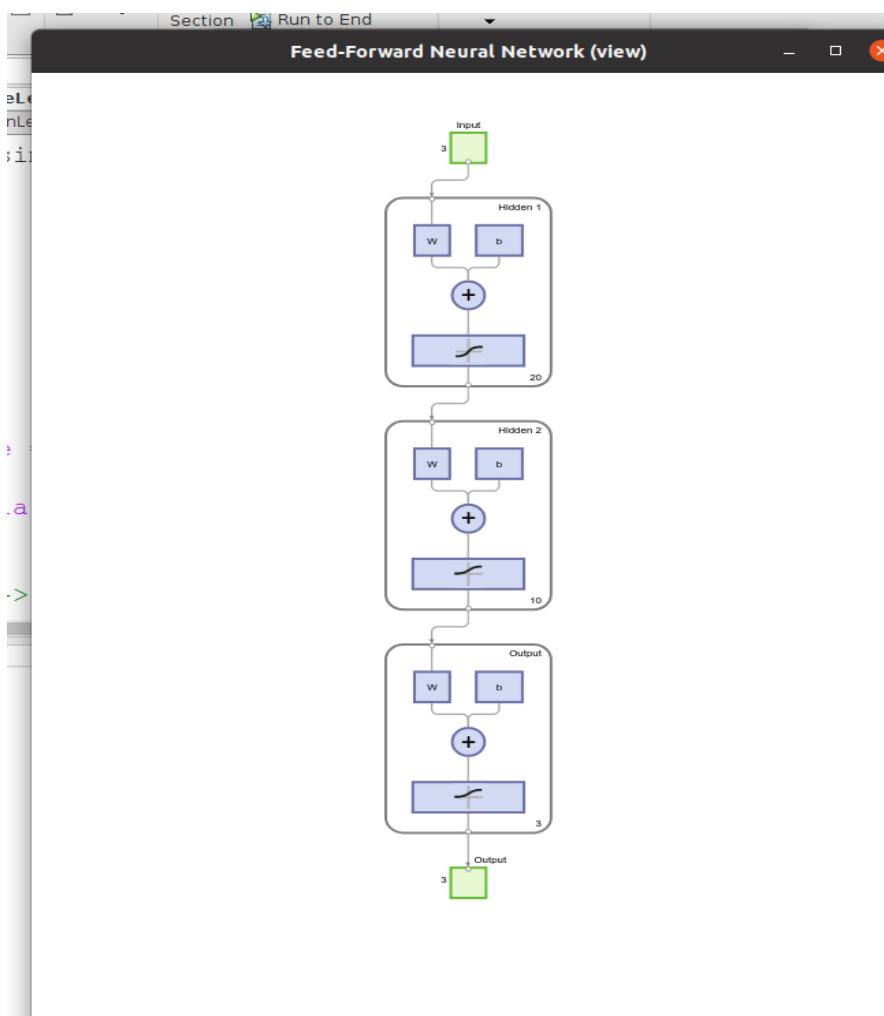
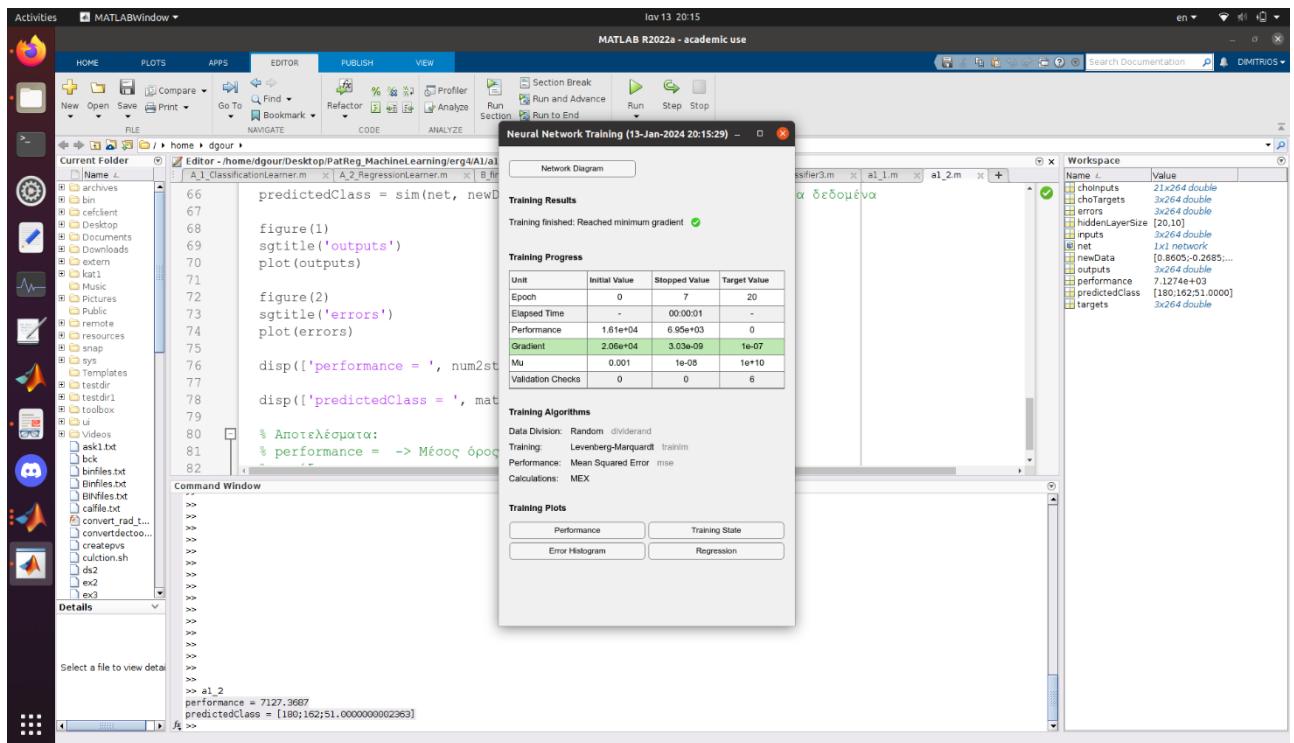
figure(2)
sgtitle('errors')
plot(errors)

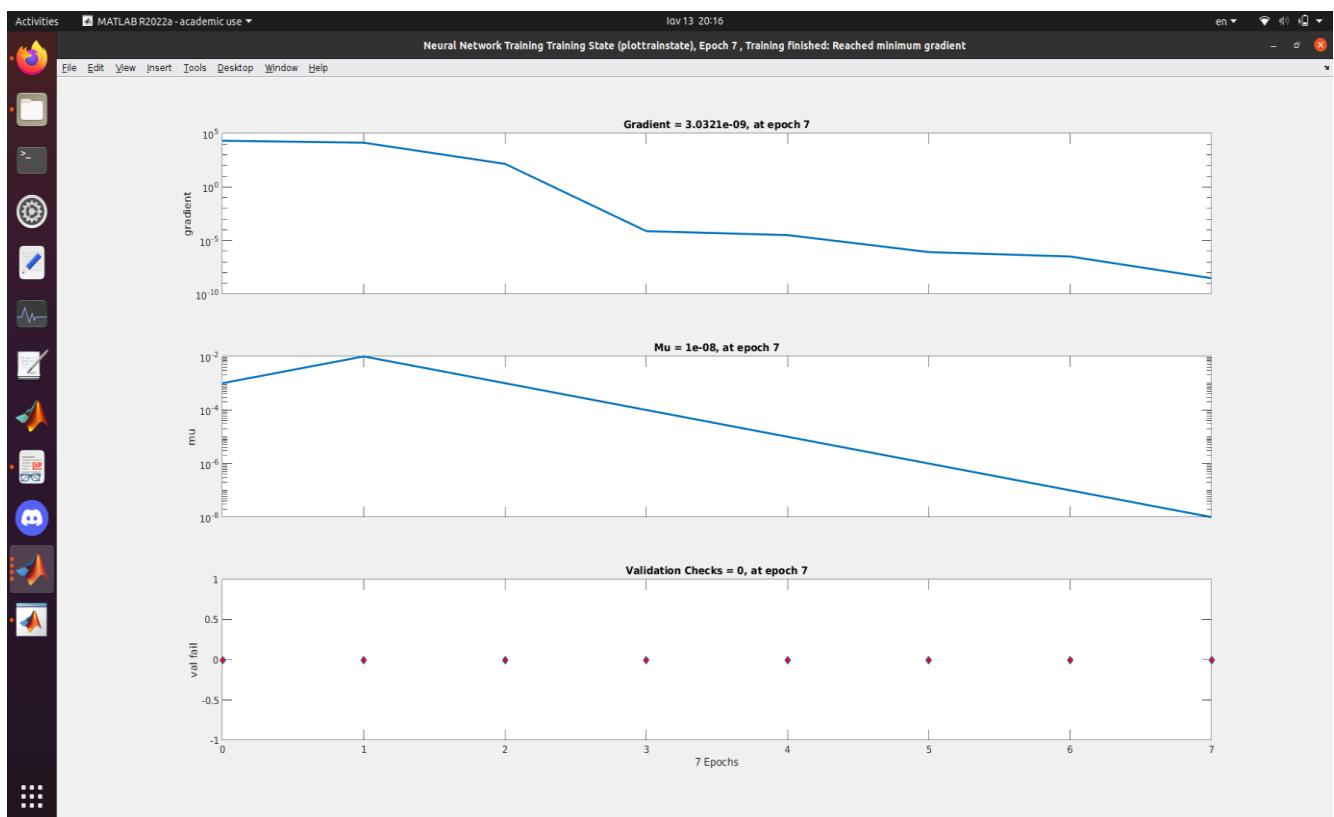
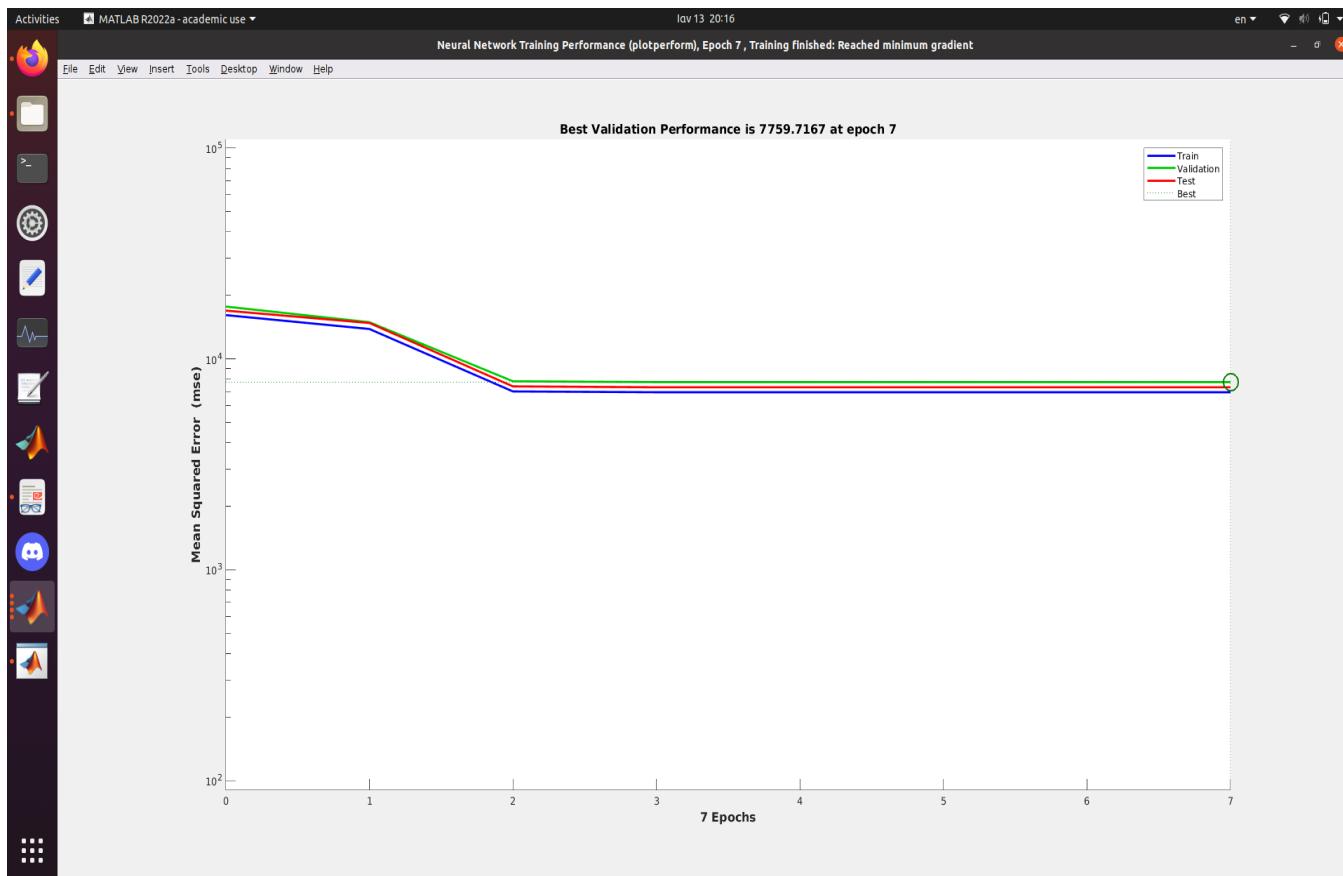
disp(['predictedClass = ', num2str(predictedClass)]);

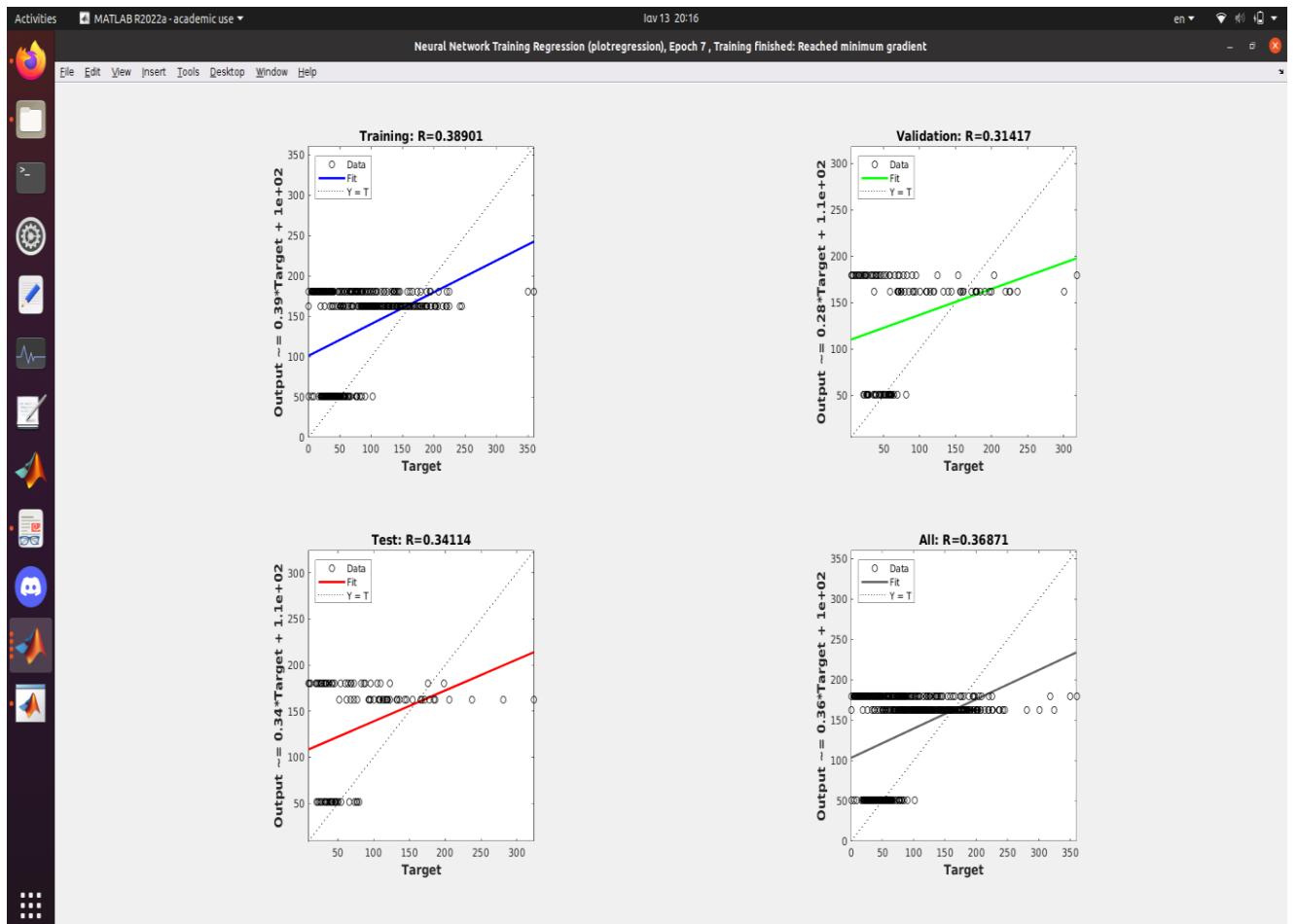
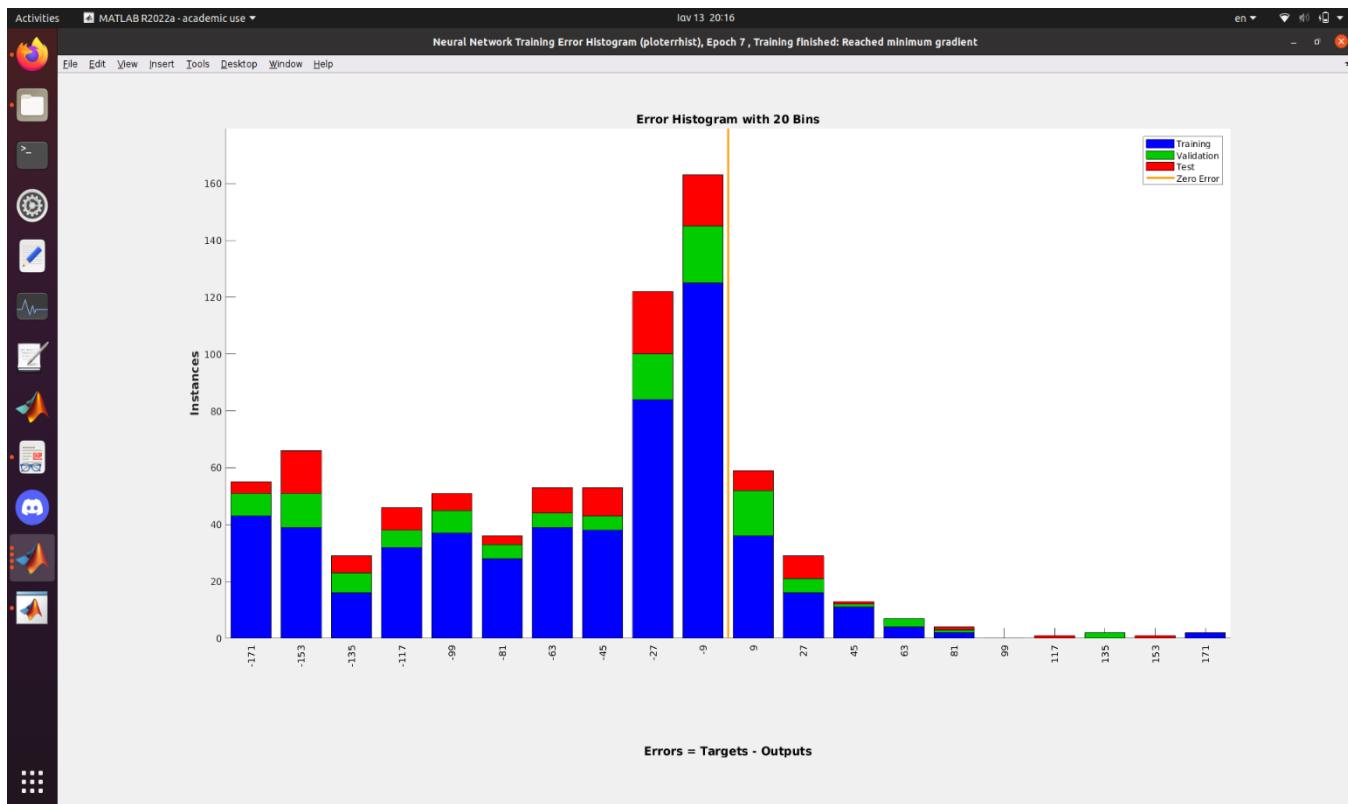
disp(['performance = ', mat2str(performance)]);

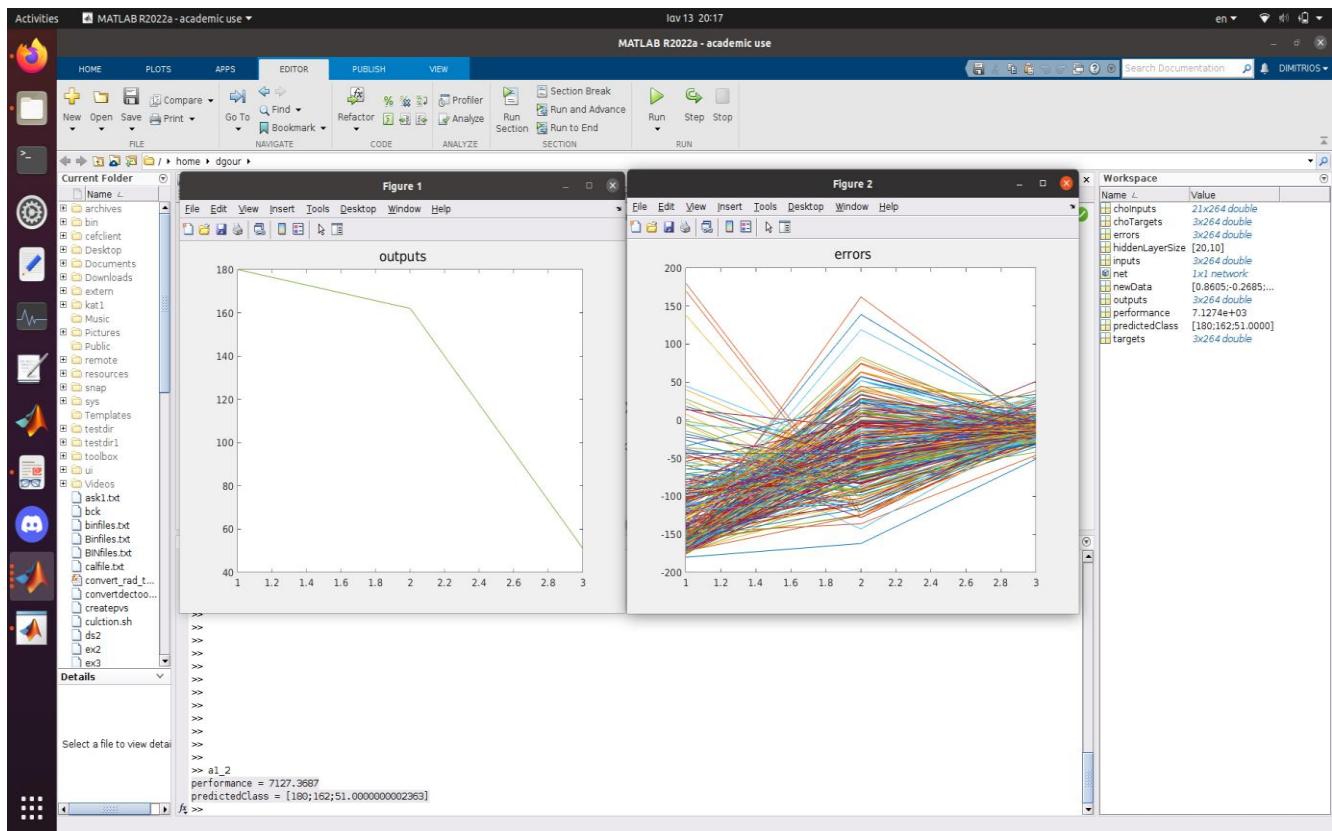
% Αποτελέσματα:
% performance = 7127.3687 -> Μέσος όρος του τετραγωνικού σφάλματος
% ανά παράδειγμα
% predictedClass = [180;162;51.0000000002363]

```









## Σχολιασμός αποτελεσμάτων

Στο 2<sup>ο</sup> παράδειγμα ακολουθούμε τα βήματα του 1<sup>ου</sup> παραδείγματος, ώστόσο οφείλουμε να σχολιάσουμε κάποιες διαφοροποιήσεις. Συγκεκριμένα, το πρόβλημα ανήκει στην κατηγορία Function Fitting, Function approximation and Curve fitting. Αυξήσαμε τα κρυφά στρώματα κατά 1. Επίσης, μειώσαμε τον αριθμό των εποχών εκπαίδευσης, από 100 σε 20. Και σε αυτό το παράδειγμα το δίκτυο σημειώνει τη μέγιστη απόδοση του κατά την 7<sup>η</sup> εποχή εκπαίδευσης.

Όσο αφορά τα αποτελέσματα:

- 1<sup>ο</sup> παράδειγμα:  
Performance(Μέσος όρος του τετραγωνικού σφάλματος ανά παράδειγμα)=0.13492  
predictedClass = 0.5
- 2<sup>ο</sup> παράδειγμα:  
Performance(Μέσος όρος του τετραγωνικού σφάλματος ανά παράδειγμα)=7127.3687  
predictedClass = [180;162;51.0000000002363]

Το SSE του 1<sup>ου</sup> παραδείγματος είναι αρκετά χαμηλό, ενώ καθώς το πρόβλημα είναι κατηγοριοποίησης αναγνωρίζονται σωστά το 50% των δειγμάτων. Επιπροσθέτως, παρατηρούμε ότι η predictedClass του 2ου παραδείγματος δίνεται σε διαφορετική μορφή. Το SSE στην 2<sup>η</sup> περίπτωση είναι αρκετά υψηλό.

## **A2) Δημιουργία, εκπαίδευση και αξιολόγηση νευρωνικών δικτύων μέσω εφαρμογών GUI του Matlab**

### **1. Clustering(nctool)**

**Dataset:** Iris Flowers

**Αρχείο:** nctool1.m

```
% Solve a Clustering Problem with a Self-Organizing Map
% Script generated by Neural Clustering app
% Created 14-Jan-2024 12:26:17
%
% This script assumes these variables are defined:
%
% irisInputs - input data.

x = irisInputs;
z = irisTargets;

% Create a Self-Organizing Map
dimension1 = 10;
dimension2 = 10;
net = selforgmap([dimension1 dimension2]);

% Train the Network
[net,tr] = train(net,x);

% Test the Network
y = net(x);

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotsomtop(net)
%figure, plotsomnc(net)
%figure, plotsomnd(net)
%figure, plotsomplanes(net)
%figure, plotsomhits(net,x)
%figure, plotsompos(net,x)
```

**Αρχείο:** nctool2.m

```
function [y1] = myNeuralNetworkFunction(x1)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 14-Jan-2024 12:27:15.
%
% [y1] = myNeuralNetworkFunction(x1) takes these arguments:
```

```

%   x = 4xQ matrix, input #1
% and returns:
%   y = 100xQ matrix, output #1
% where Q is the number of samples.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Layer 1
IW1_1 = [5.6666666666666660745 4.20000000000000001776
1.3666666666666664742 0.266666666666666297;5.7000000000000001776
3.7999999999999998224 1.699999999999999556 0.299999999999999889;5.5
3.5 1.3000000000000000444 0.2000000000000000111;6.3800000000000007816
3.1800000000000001599 3.8600000000000007638
1.199999999999999556;6.7000000000000001776 3.100000000000000888
4.7000000000000001776 1.5;6.9500000000000001776 3.1500000000000003553
4.8000000000000007105 1.449999999999999556;7 3.06666666666664298
5.266666666666666075 1.633333333333333037;7.2000000000000001776
3.100000000000000888 5.9000000000000003553
1.7000000000000001776;7.4000000000000003553 2.799999999999998224
6.0999999999999996447 1.89999999999999112;7.66666666666660745
2.7999999999999998224 6.73333333333333925
2.13333333333333037;5.4000000000000003553 3.89999999999999112
1.699999999999999556 0.400000000000000222;5.4000000000000003553
3.89999999999999112 1.300000000000000444
0.400000000000000222;5.790000000000000355 3.430000000000001599
2.3800000000000003375 0.59999999999999778;6.4600000000000008527
3.100000000000000888 4.1500000000000003553
1.289999999999998135;6.6500000000000003553 3.049999999999998224
4.4000000000000003553 1.39999999999999112;6.799999999999998224
2.799999999999998224 4.799999999999998224
1.39999999999999112;6.84999999999996447 3.049999999999998224
5.4500000000000001776 2.100000000000000888;7.09999999999996447 3
5.9000000000000003553 2.100000000000000888;7.7000000000000001776 3
6.099999999999996447 2.299999999999998224;7.8000000000000007105
3.799999999999998224 6.5500000000000007105
2.100000000000000888;5.09999999999996447 3.799999999999998224
1.89999999999999112 0.400000000000000222;5.349999999999996447
3.7000000000000001776 1.5 0.200000000000000111;5.4000000000000003553
3.39999999999999112 1.600000000000000888
0.3000000000000004441;6.0625 3.162500000000000888
3.2625000000000001776 0.962499999999991118;6.4000000000000003553
2.89999999999999112 4.299999999999998224
1.300000000000000444;6.549999999999998224 2.849999999999996447
4.599999999999996447 1.39999999999999112;6.7000000000000001776 3 5
1.69999999999999556;6.80000000000007105 3.049999999999998224
5.1500000000000003553 2.299999999999998224;6.7750000000000003553
3.2000000000000001776 5.7250000000000005329
2.375;7.2000000000000001776 3.600000000000000888
6.099999999999996447 2.5;5.125 3.849999999999996447
1.524999999999999112 0.25;5.0333333333332149
3.399999999999994671 1.63333333333335258
0.499999999999994449;5.34444444444441089 3.18888888888886619
2.3444444444444553 0.65555555555556905;6.099999999999996447
2.799999999999998224 4 1.300000000000000444;6.099999999999996447
2.799999999999998224 4.7000000000000001776
1.19999999999999556;6.299999999999998224 2.799999999999998224
5.099999999999996447 1.5;6.4000000000000003553 2.7000000000000001776
5.299999999999998224 1.89999999999999112;6.4000000000000003553
3.2000000000000001776 5.299999999999998224

```

2.2999999999999998224; 6.2999999999999998224 3.2999999999999998224 6  
 2.5; 6.7000000000000001776 3.2999999999999998224 5.7000000000000001776  
 2.1000000000000000888; 5.1166666666666662522 3.43333333333332682  
 1.43333333333331261 0.2166666666666664631; 4.75  
 3.2999999999999998224 1.6000000000000000888  
 0.200000000000000111; 4.7999999999999998224 3.399999999999999112  
 1.899999999999999112 0.200000000000000111; 5.7000000000000001776  
 2.6000000000000000888 3.5 1; 5.9874999999999998224  
 2.61250000000000002665 4.2999999999999998342 1.274999999999999112; 6  
 2.2000000000000001776 5 1.5; 6.2999999999999998224 2.5 5  
 1.899999999999999112; 6.5 3 5.2000000000000001776 2; 6.25  
 3.399999999999999112 5.5 2.3499999999999996447; 6.5 3  
 5.7999999999999998224 2.2000000000000001776; 5 3.2000000000000001776  
 1.199999999999999556 0.200000000000000111; 4.8999999999999994671  
 3.0666666666666664298 1.566666666666666519  
 0.1666666666666665741; 5.258333333333328596 2.899999999999999112  
 2.5000000000000004441 0.5666666666666665186; 6 2.2000000000000001776  
 4 1; 6.25 2.25 4.4500000000000001776  
 1.399999999999999112; 6.2999999999999998224 2.5 4.9000000000000003553  
 1.5; 6.25 2.75 4.8499999999999996447 1.800000000000000444; 6.5  
 3.2000000000000001776 5.0999999999999996447 2; 6.4000000000000003553  
 2.7999999999999998224 5.5999999999999996447  
 2.1500000000000003553; 6.7000000000000001776 2.5 5.7999999999999998224  
 1.800000000000000444; 4.5999999999999996447 3.600000000000000888 1  
 0.200000000000000111; 4.83333333333330373 3 1.399999999999999112  
 0.1999999999999998335; 5.1076923076923081979 2.90000000000000003553  
 2.2538461538461538325 0.48461538461538467004; 5.7999999999999998224  
 2.6500000000000003553 3.9500000000000001776  
 1.1999999999999556; 5.7999999999999998224 2.7000000000000001776  
 4.0999999999999996447 1; 5.9000000000000003553 3 4.2000000000000001776  
 1.5; 6.0666666666666664298 2.933333333331261 4.5999999999999996447  
 1.4333333333331261; 6.3499999999999996447 3.25  
 4.5999999999999996447 1.550000000000000444; 6.25  
 3.0875000000000003553 5.1124999999999998224  
 1.80000000000002665; 6.4500000000000001776 3.0499999999999998224 5.5  
 1.8000000000000444; 4.625 3.2250000000000005329  
 1.399999999999999112 0.22499999999999778; 4.875 2.9375 2.0625  
 0.450000000000000111; 5.5 2.399999999999999112 4  
 1.300000000000000444; 5.5999999999999996447 2.899999999999999112  
 3.600000000000000888 1.300000000000000444; 5.66666666666660745  
 2.96666666666667851 4.166666666666660745  
 1.26666666666666075; 5.7000000000000001776 2.7999999999999998224 4.5  
 1.3000000000000444; 6 3.399999999999999112 4.5  
 1.600000000000000888; 6 3 4.7999999999999998224  
 1.8000000000000444; 6.1374999999999992895 2.9875000000000002665  
 5.1624999999999996447 1.7750000000000003553; 6.0999999999999996447  
 2.600000000000000888 5.5999999999999996447  
 1.399999999999999112; 4.4000000000000003553 3.2000000000000001776  
 1.300000000000000444 0.200000000000000111; 4.36666666666662522  
 2.9666666666666341 1.26666666666666075  
 0.166666666666668517; 5.2285714285714286476 2.3428571428571425272  
 3.5285714285714280258 1.0714285714285713969; 5.533333333333332149  
 2.43333333333331261 3.7999999999999998224  
 1.06666666666666519; 5.6500000000000003553 2.75  
 4.1500000000000003553 1.300000000000000444; 5.5 2.600000000000000888  
 4.4000000000000003553 1.199999999999999556; 5.5 3 4.5  
 1.5; 5.9000000000000003553 3.2000000000000001776 4.7999999999999998224  
 1.800000000000000444; 5.9000000000000003553 3 5.0999999999999996447  
 1.800000000000000444; 6 2.7000000000000001776 5.0999999999999996447  
 1.600000000000000888; 4.5 2.2999999999999998224 1.300000000000000444  
 0.299999999999999889; 5.0999999999999996447 2.5 3

```

1.1000000000000000888;4.9500000000000001776 2.349999999999996447
3.299999999999998224 1;5 2 3.5 1;5.2000000000000001776
2.7000000000000001776 3.899999999999999112
1.399999999999999112;4.9000000000000003553 2.5 4.5
1.699999999999999556;5.599999999999996447 2.799999999999998224
4.9000000000000003553 2;5.799999999999998224 2.799999999999998224
5.099999999999996447 2.399999999999999112;5.799999999999998224
2.7000000000000001776 5.099999999999996447
1.899999999999999112;5.7000000000000001776 2.5 5 2];

% ===== SIMULATION =====

% Input 1
% no processing

% Layer 1
z1 = negdist_apply(IW1_1,x1);
a1 = compet_apply(z1);

% Output 1
y1 = a1;
end

% ===== MODULE FUNCTIONS =====

% Negative Distance Weight Function
function z = negdist_apply(w,p,~)
[S,R] = size(w);
Q = size(p,2);
if isa(w, 'gpuArray')
    z = iNegDistApplyGPU(w,p,R,S,Q);
else
    z = iNegDistApplyCPU(w,p,S,Q);
end
end
function z = iNegDistApplyCPU(w,p,S,Q)
z = zeros(S,Q);
if (Q<S)
    pt = p';
    for q=1:Q
        z (:,q) = sum(bsxfun(@minus,w,pt(q,:)).^2,2);
    end
else
    wt = w';
    for i=1:S
        z (i,:) = sum(bsxfun(@minus,wt(:,i),p).^2,1);
    end
end
z = -sqrt(z);
end
function z = iNegDistApplyGPU(w,p,R,S,Q)
p = reshape(p,1,R,Q);
sd = arrayfun(@iNegDistApplyGPUHelper,w,p);
z = -sqrt(reshape(sum(sd,2),S,Q));
end
function sd = iNegDistApplyGPUHelper(w,p)
sd = (w-p) .^ 2;
end

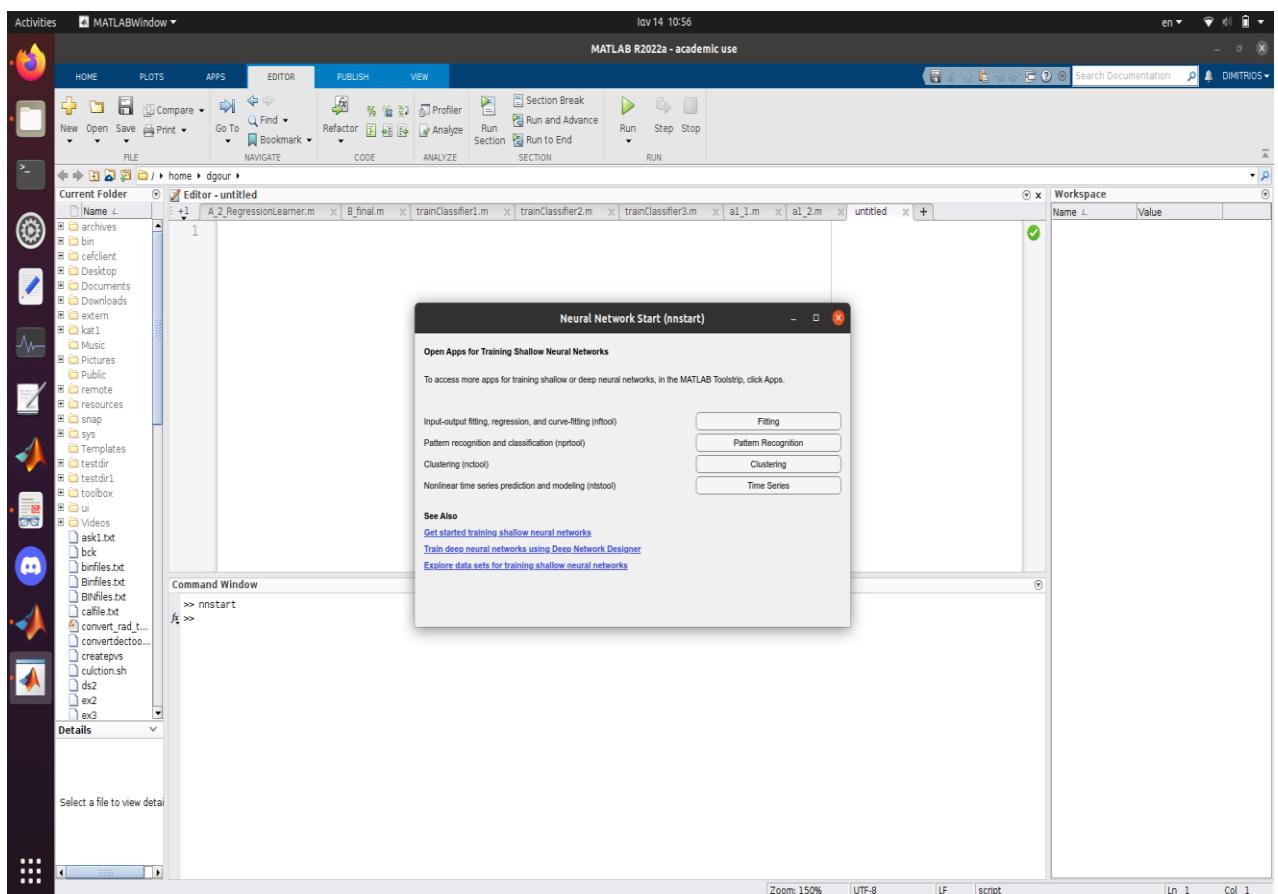
% Competitive Transfer Function
function a = compet_apply(n,~)

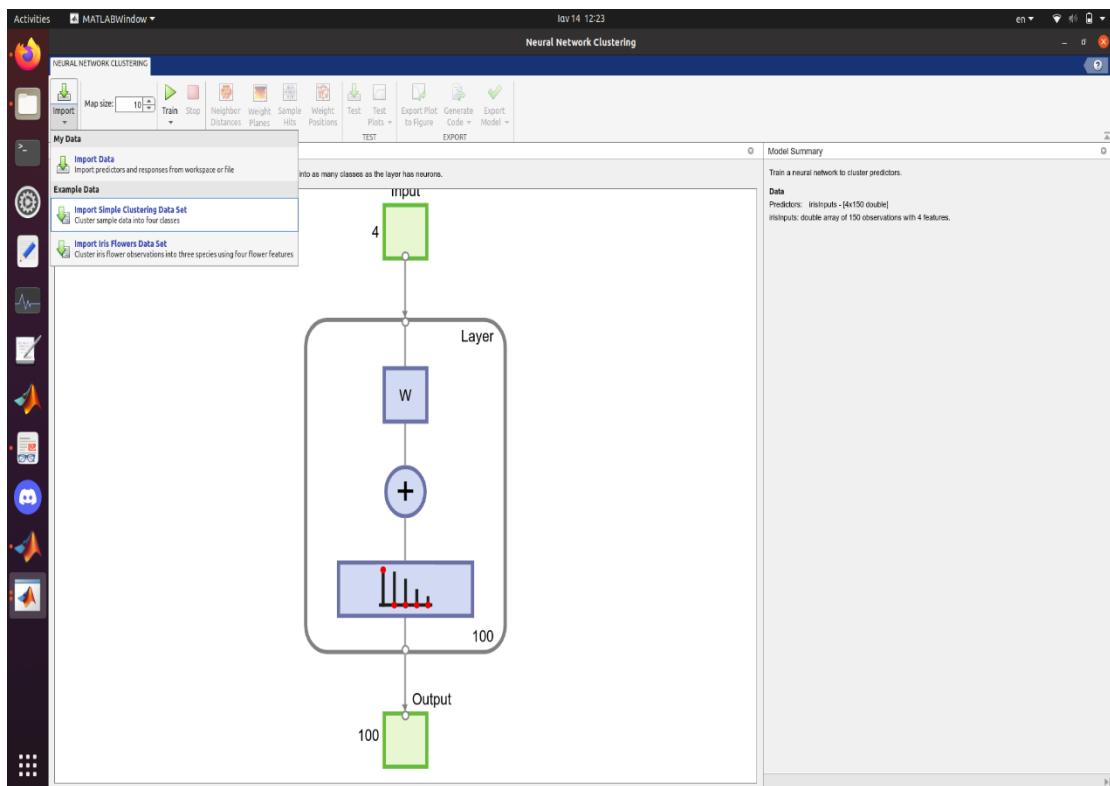
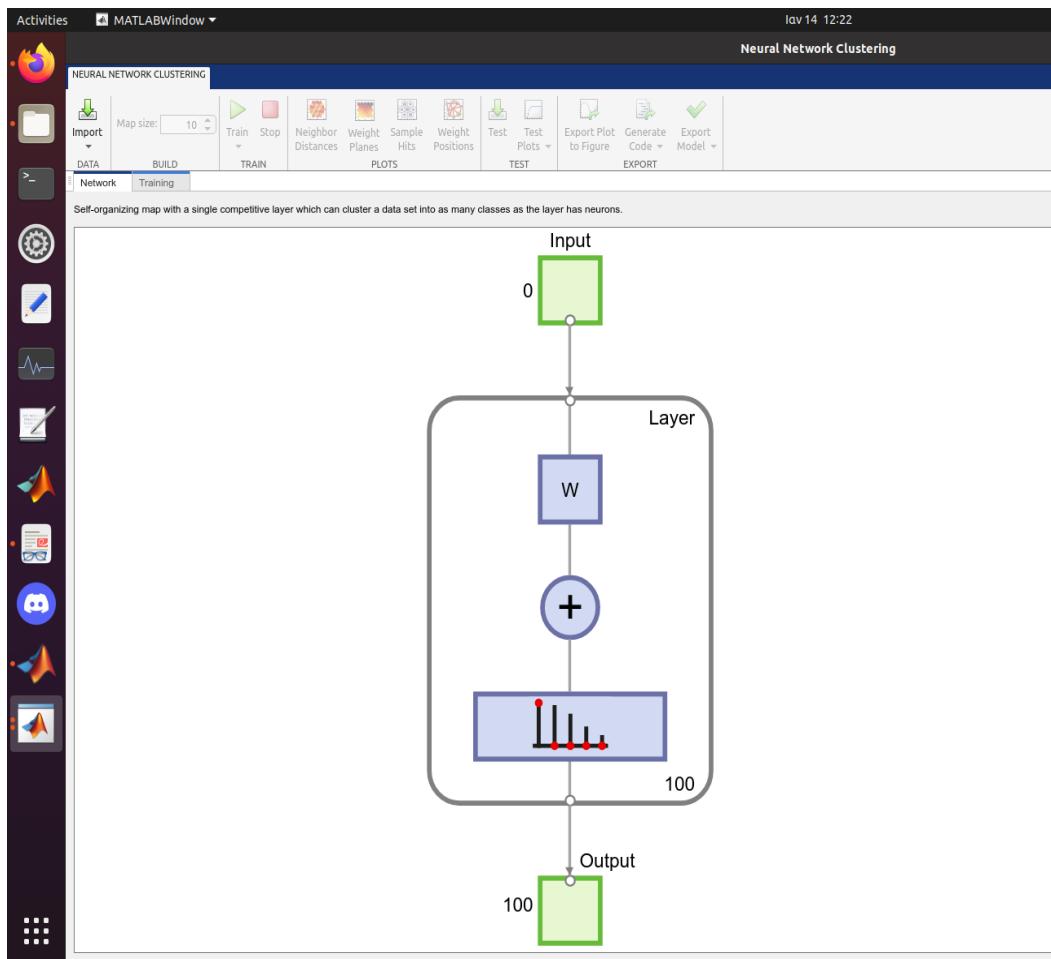
```

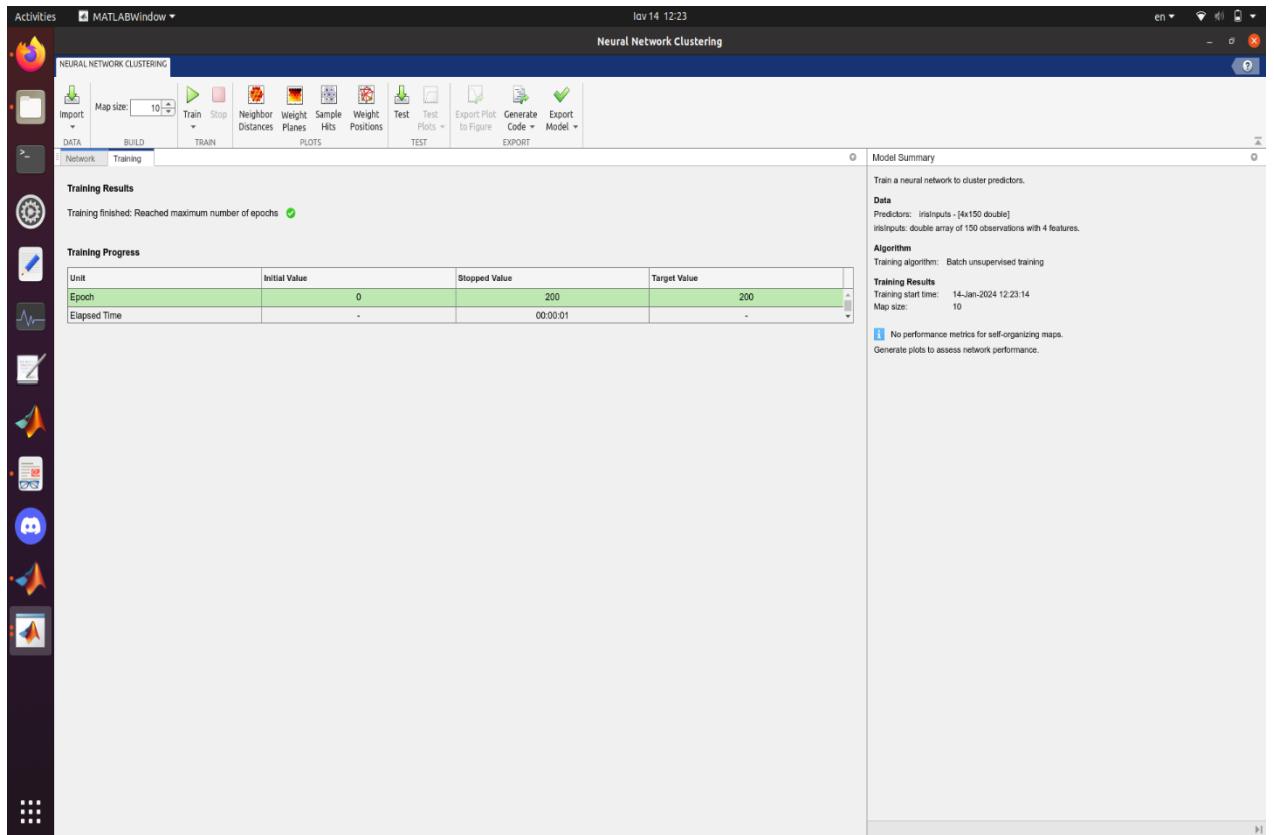
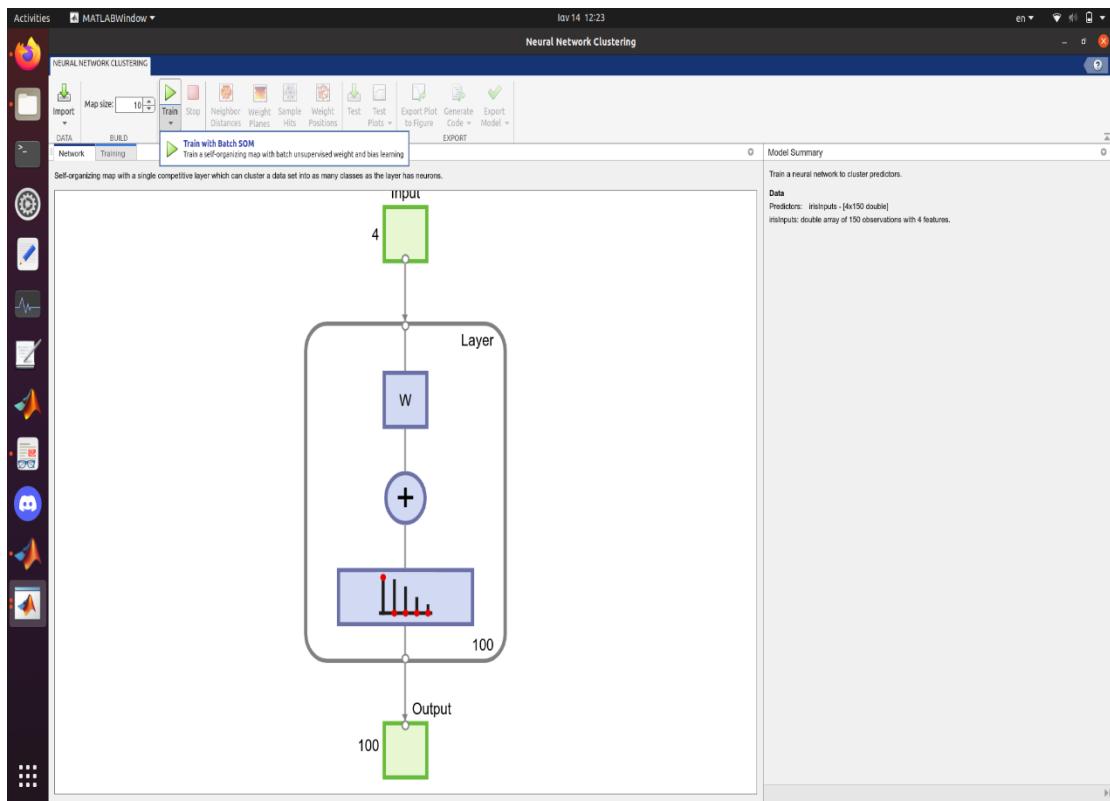
```

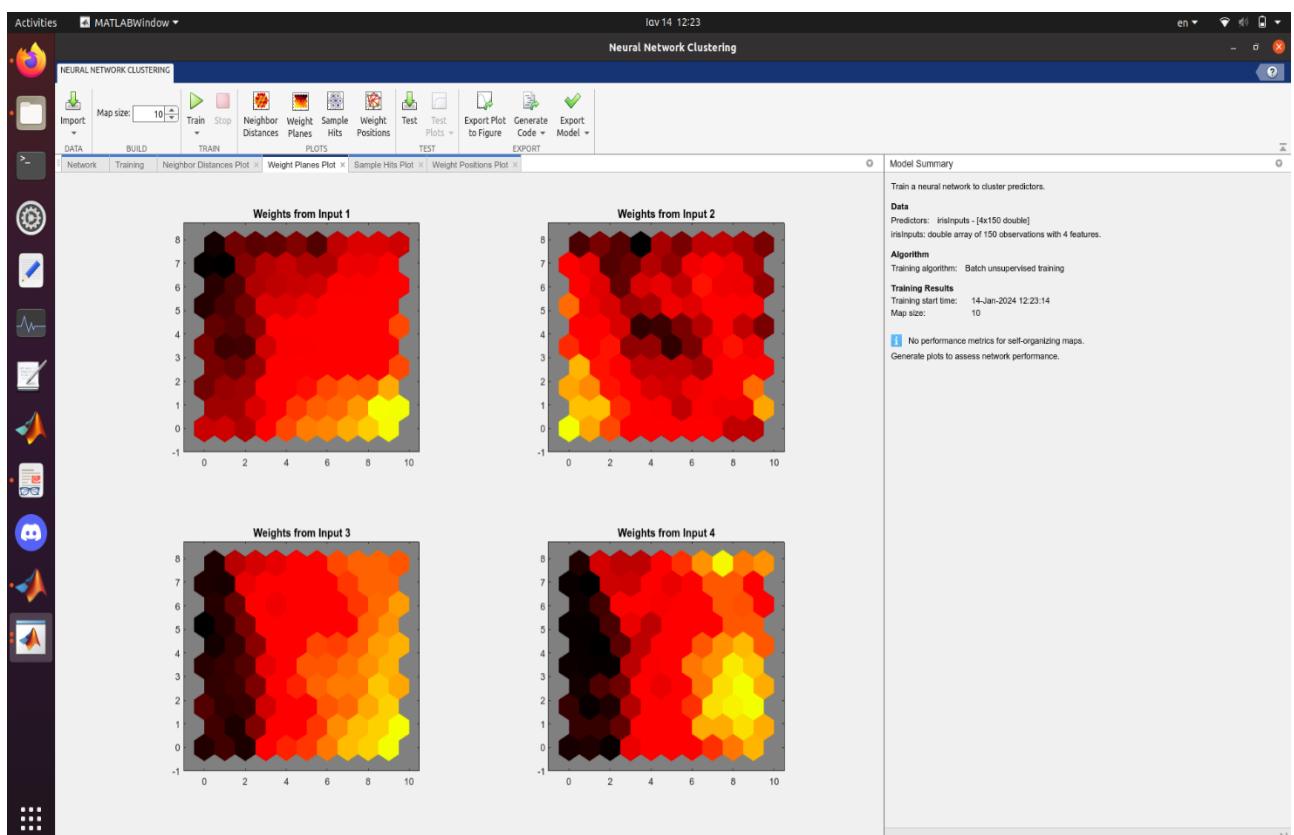
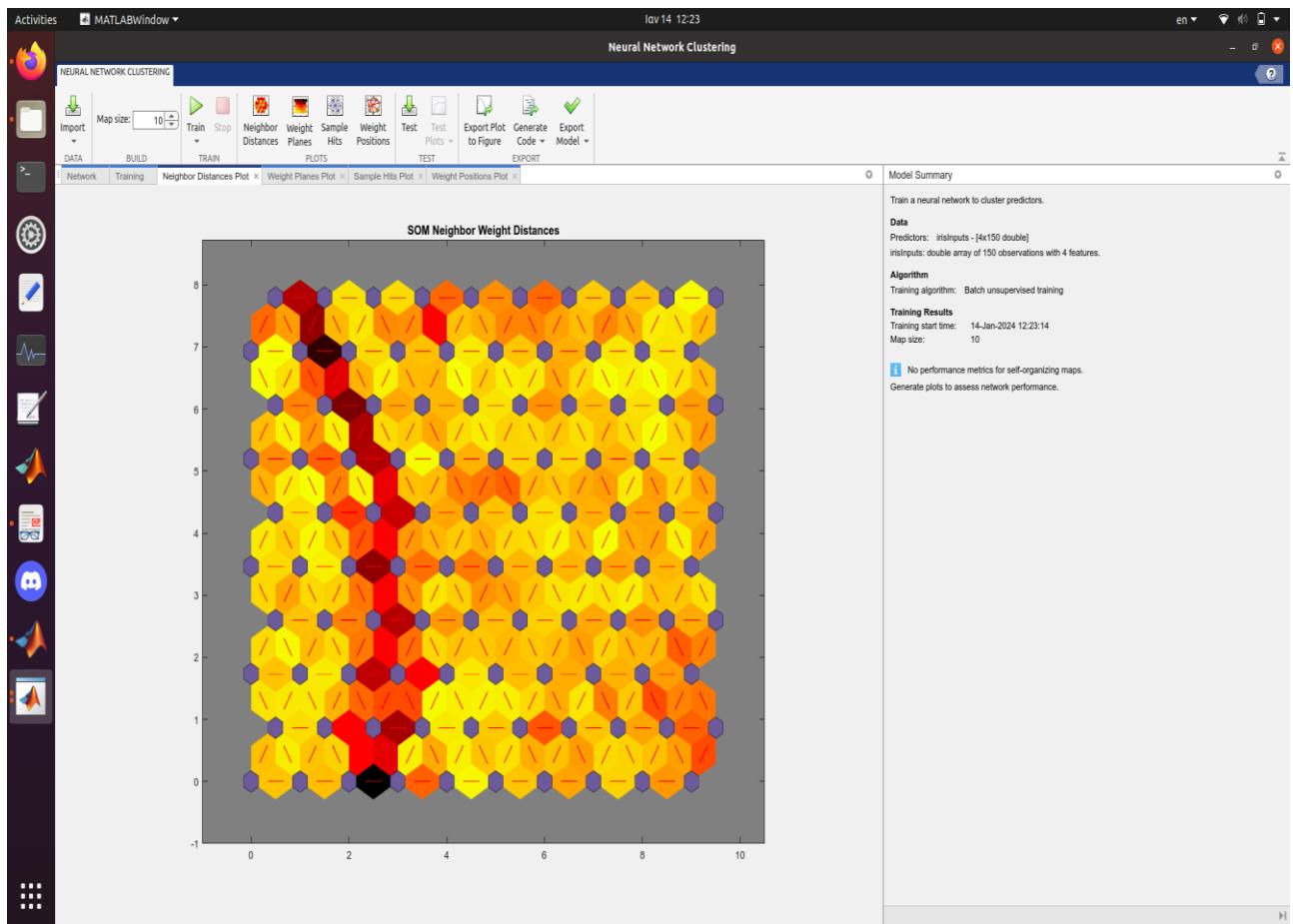
if isempty(n)
    a = n;
else
    [S,Q] = size(n);
    nanInd = any(isnan(n),1);
    a = zeros(S,Q, 'like',n);
    [~,maxRows] = max(n,[],1);
    onesInd = maxRows + S*(0:(Q-1));
    a(onesInd) = 1;
    a(:,nanInd) = NaN;
end
end

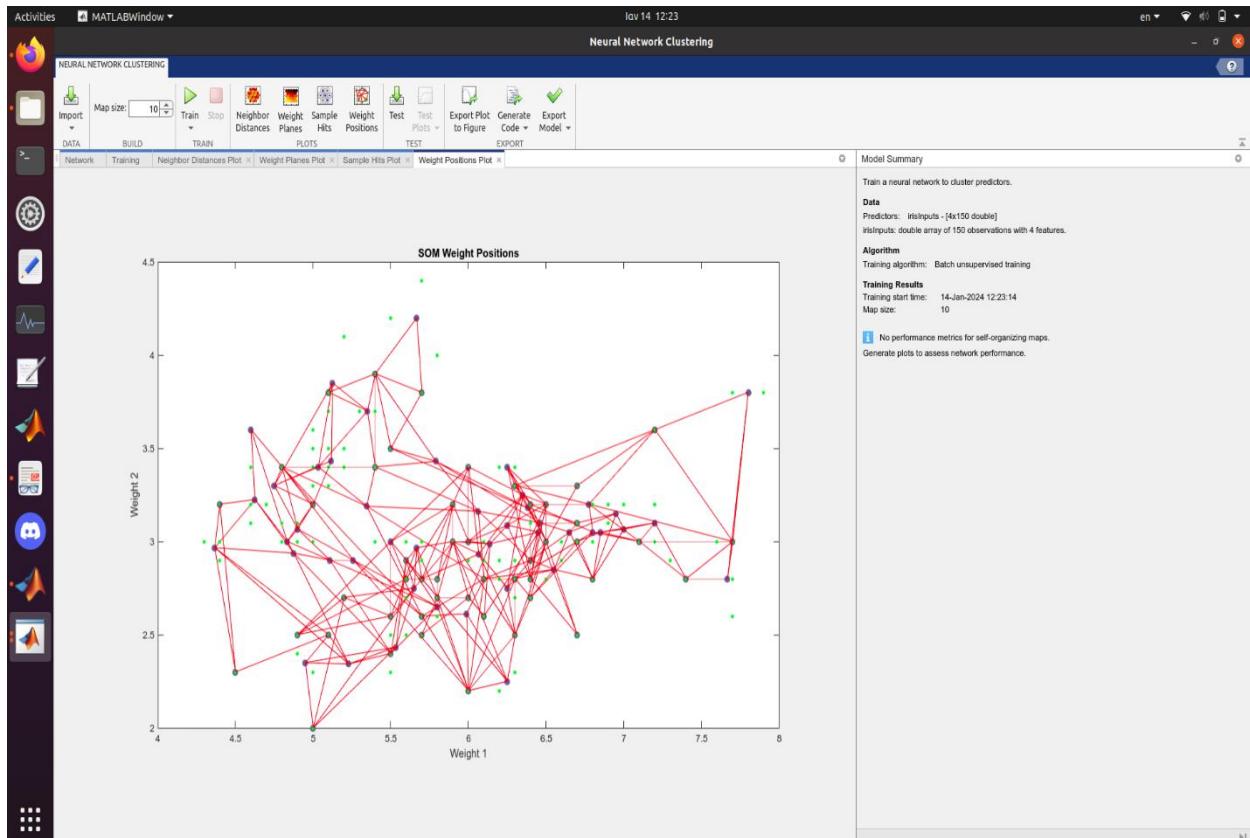
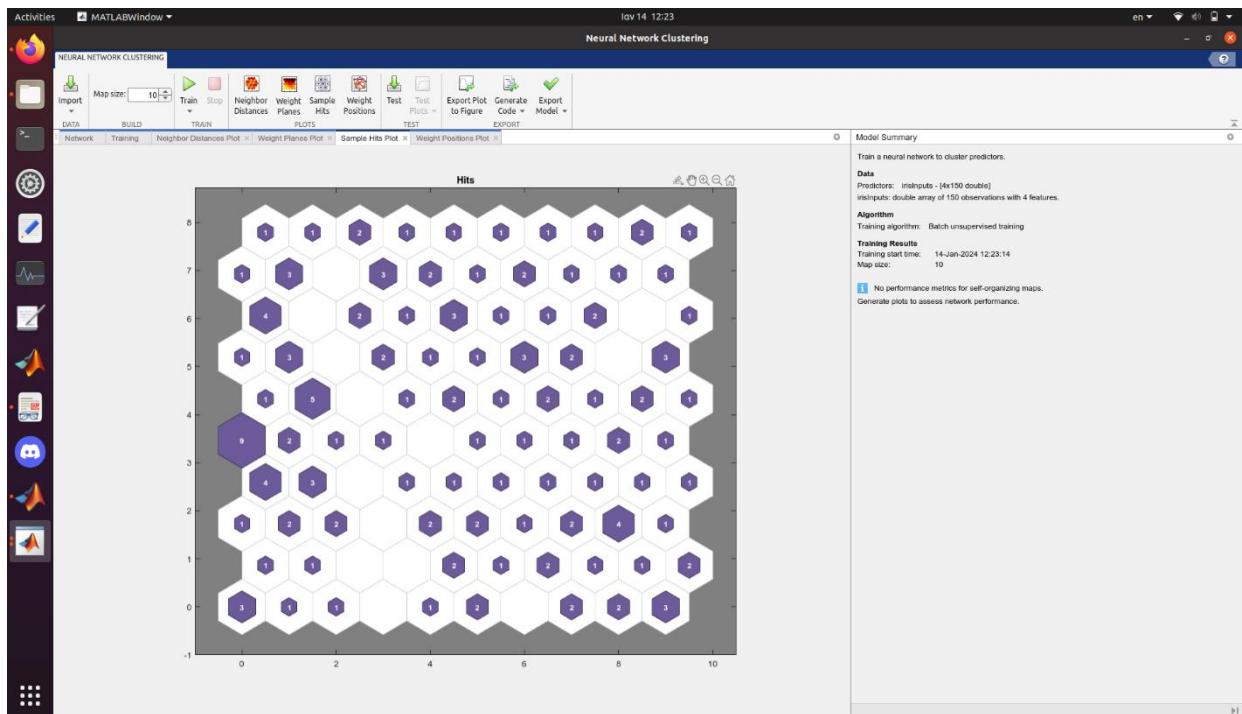
```

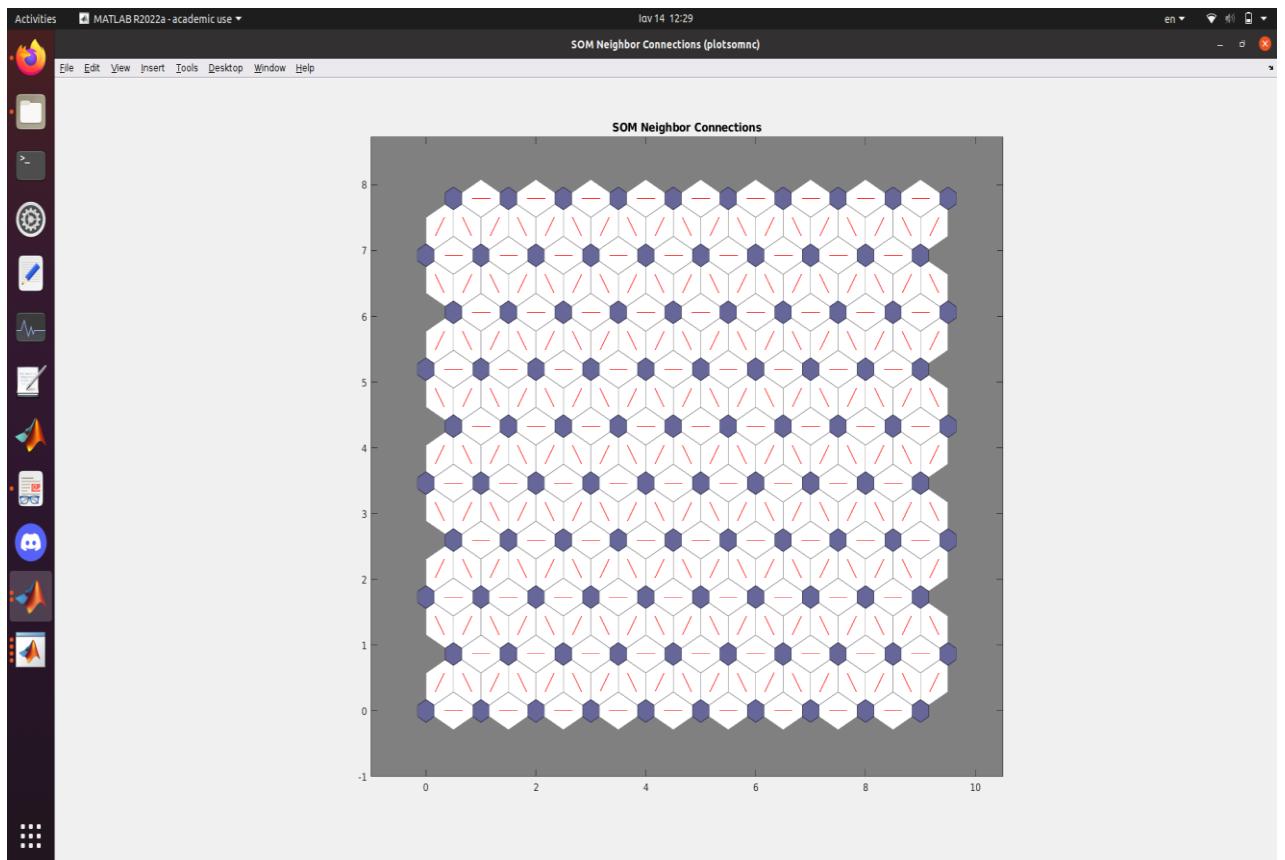












## 2. Time Series(ntstool)

NAR Network

**Dataset:** Simple NAR

**Αρχείο:** ntstool1.m

```
% Solve an Autoregression Time-Series Problem with a NAR Neural
Network
% Script generated by Neural Time Series app
% Created 14-Jan-2024 12:56:36
%
% This script assumes this variable is defined:
%
% simplenarTargets - feedback time series.

T = simplenarTargets;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
```

```

% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm'; % Levenberg-Marquardt backpropagation.

% Create a Nonlinear Autoregressive Network
feedbackDelays = 1:2;
hiddenLayerSize = 10;
net = narnet(feedbackDelays,hiddenLayerSize,'open',trainFcn);

% Prepare the Data for Training and Simulation
% The function PREPARETS prepares timeseries data for a particular
network,
% shifting time by the minimum amount to fill input states and layer
% states. Using PREPARETS allows you to keep your original time
series data
% unchanged, while easily customizing it for networks with differing
% numbers of delays, with open loop or closed loop feedback modes.
[x,xi,ai,t] = preparets(net,[],[],T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t,xi,ai);

% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotresponse(t,y)
%figure, ploterrcorr(e)
%figure, plotinerrcorr(x,e)

% Closed Loop Network
% Use this network to do multi-step prediction.
% The function CLOSELOOP replaces the feedback input with a direct
% connection from the output layer.
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,[],[],T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc)

% Step-Ahead Prediction Network
% For some applications it helps to get the prediction a timestep
early.
% The original network returns predicted y(t+1) at the same time it
is

```

```

% given y(t+1). For some applications such as decision making, it
would
% help to have predicted y(t+1) once y(t) is available, but before
the
% actual y(t+1) occurs. The network can be made to return its output
a
% timestep early by removing one delay so that its minimal tap delay
is now
% 0 instead of 1. The new network returns the same outputs as the
original
% network, but outputs are shifted left one timestep.
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = prepares(nets,[],{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)

```

## Αρχείο: ntstool2.m

```

function [y1,xf1] = myNeuralNetworkFunction(x1,xi1)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 14-Jan-2024 12:56:40.
%
% [y1,xf1] = myNeuralNetworkFunction(x1,xi1) takes these arguments:
%   x1 = 1xTS matrix, input #1
%   xi1 = 1x2 matrix, initial 2 delay states for input #1.
% and returns:
%   y1 = 1xTS matrix, output #1
%   xf1 = 1x2 matrix, final 2 delay states for input #1.
% where TS is the number of timesteps.

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = 0.162182308193243;
x1_step1.gain = 2.38741093727801;
x1_step1.ymin = -1;

% Layer 1
b1 = [-0.78050535166348911087;-2.0849245192338132782;-
2.8785643178901474393;0.89766331609471228248;0.088976040943150758178;-
0.44088044139418058887;-0.73581130670817540551;-
0.95103927214832273407;-3.4232842496954440925;-
5.6182386988462162947];
IW1_1 = [0.25649372671907622045
0.51306587587460850131;0.99039191925942049455 -
4.4165174406823366482;2.9822327072348175925 -2.9984809989611913572;-
2.5506368246075159512 3.7600706728469095452;-4.8183612741607850793

```

```

0.80615057471338102957;-2.4543392703677011646 -
0.66941920036414126471;-4.1759717946387926446 1.4773697178110323236;-
0.31233797708968025209 -0.62586427485519002811;-3.9501011793521545457
2.0869667894484149606;-2.7671126974044439883 -2.5346962515515634173];

% Layer 2
b2 = -2.5872014156057723255;
LW2_1 = [-4.1909607108564701505 -0.00017113419932886507783
0.00044023366310221820645 0.00013488906020620426859 -
0.00026635585721289872048 -0.000320214617603362475
0.00017808334703919804706 -3.1438488092274465835
8.1416521093992210009e-05 1.51164601487840633];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 2.38741093727801;
y1_step1.xoffset = 0.162182308193243;

% ===== SIMULATION =====

% Dimensions
TS = size(x1,2); % timesteps

% Input 1 Delay States
xd1 = mapminmax_apply(x1,x1_step1);
xd1 = [xd1 zeros(1,1)];

% Allocate Outputs
y1 = zeros(1,TS);

% Time loop
for ts=1:TS

    % Rotating delay state position
    xdts = mod(ts+1,3)+1;

    % Input 1
    xd1(:,xdts) = mapminmax_apply(x1(:,ts),x1_step1);

    % Layer 1
    tapdelay1 = reshape(xd1(:,mod(xdts-[1 2]-1,3)+1),2,1);
    a1 = tansig_apply(b1 + IW1_1*tapdelay1);

    % Layer 2
    a2 = b2 + LW2_1*a1;

    % Output 1
    y1(:,ts) = mapminmax_reverse(a2,y1_step1);
end

% Final delay states
finalxts = TS+(1: 2);
xits = finalxts(finalxts<=2);
xts = finalxts(finalxts>2)-2;
xf1 = [x1(:,xits) x1(:,xts)];
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)

```

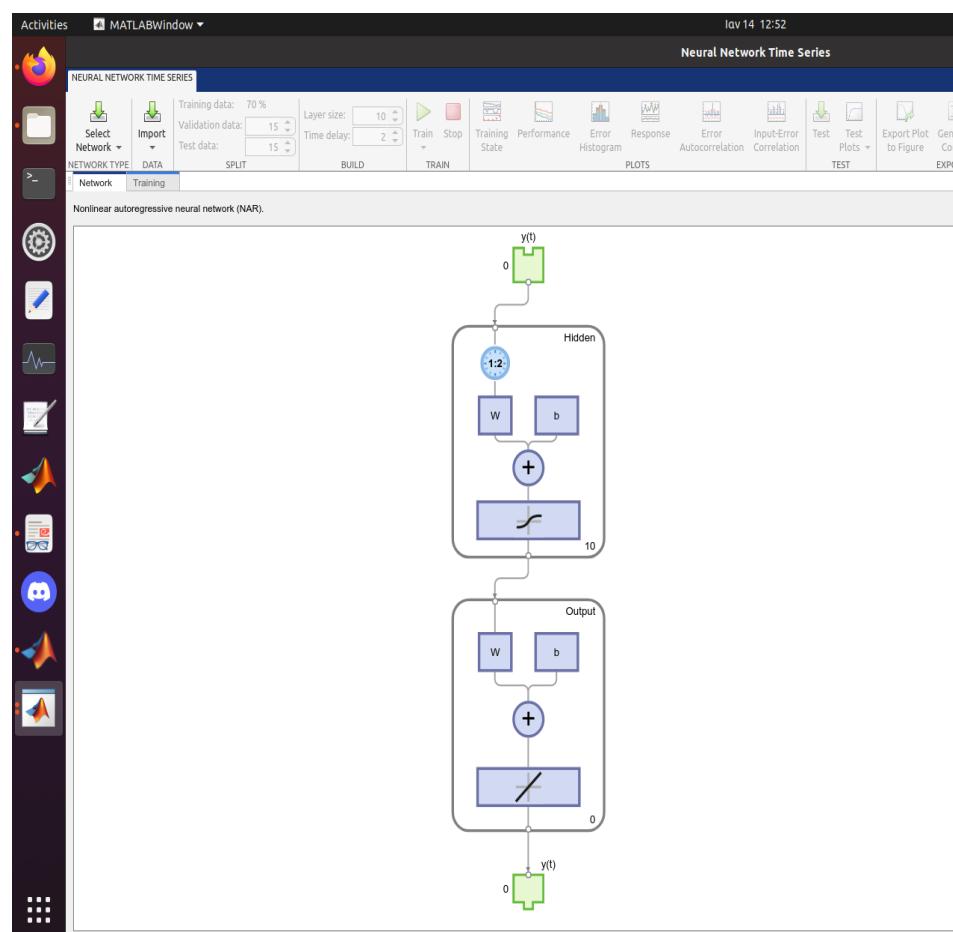
```

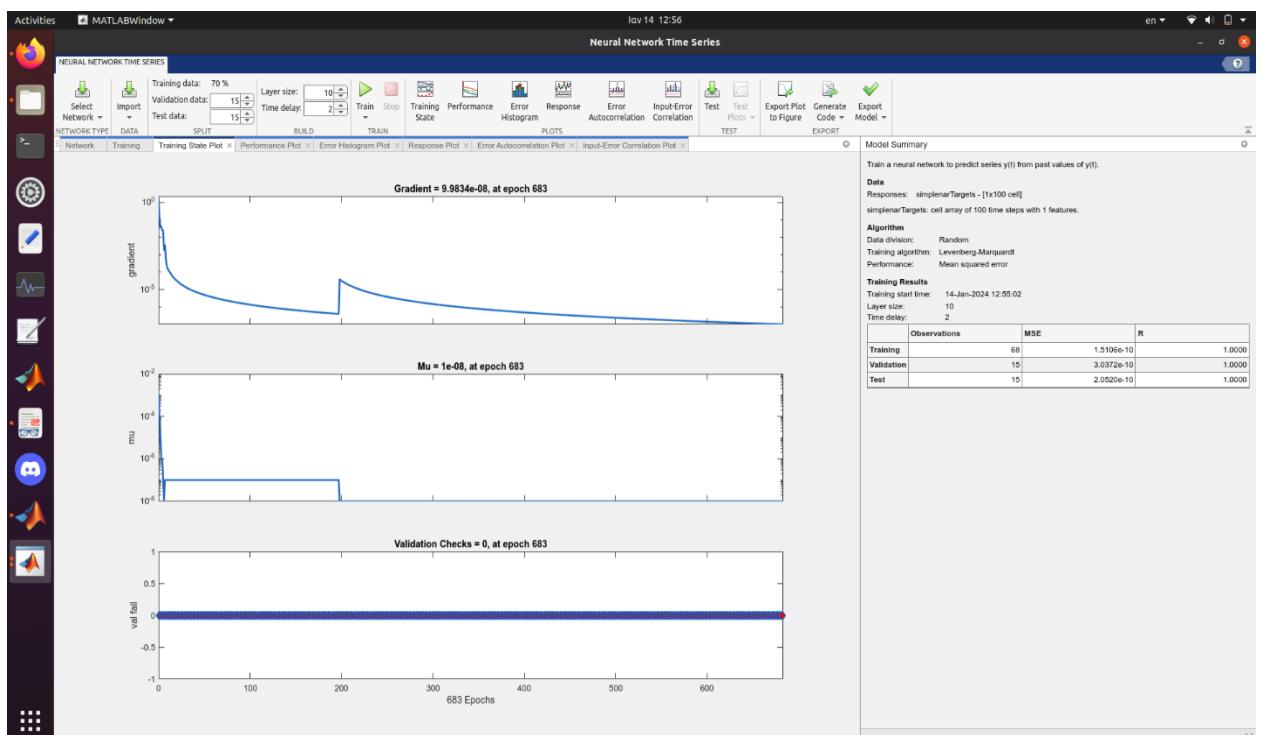
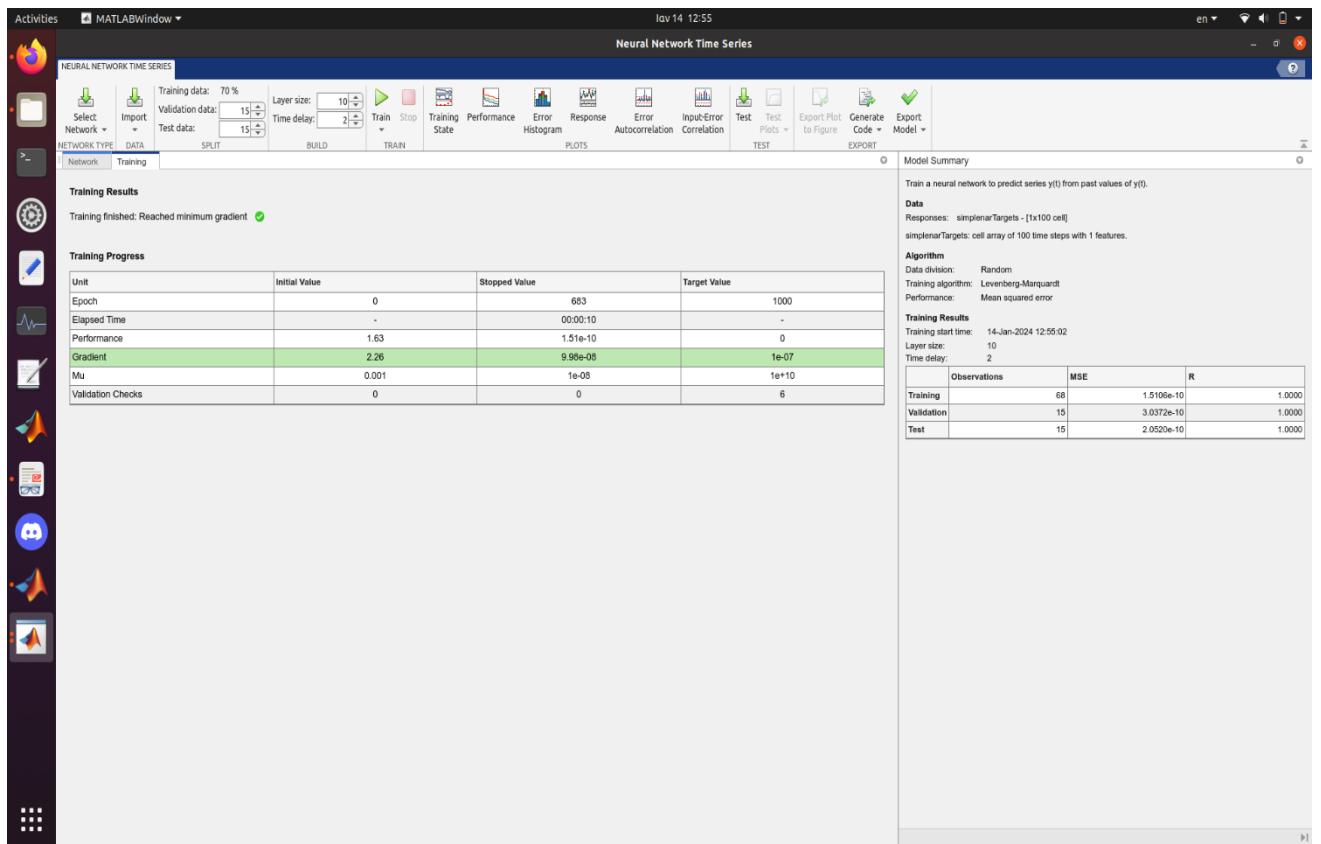
y = bsxfun(@minus,x=settings.xoffset);
y = bsxfun(@times,y=settings.gain);
y = bsxfun(@plus,y=settings.ymin);
end

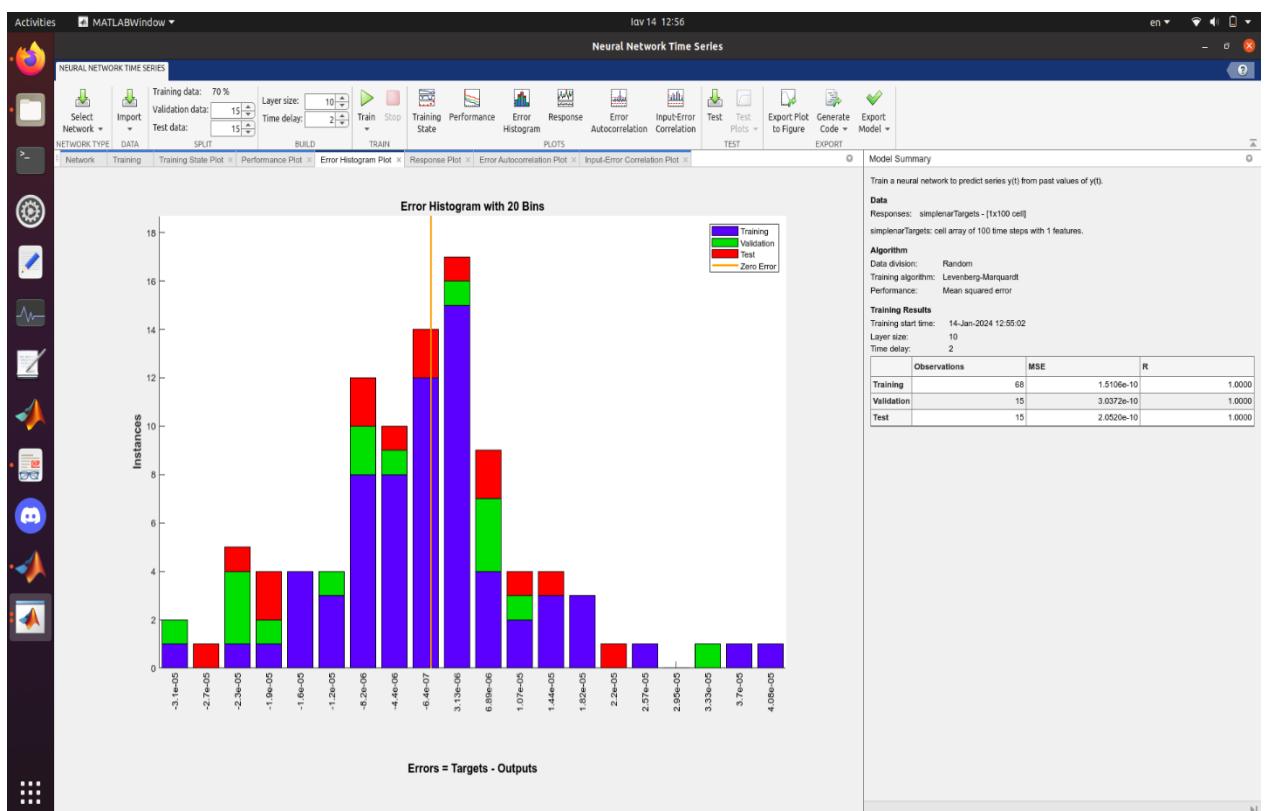
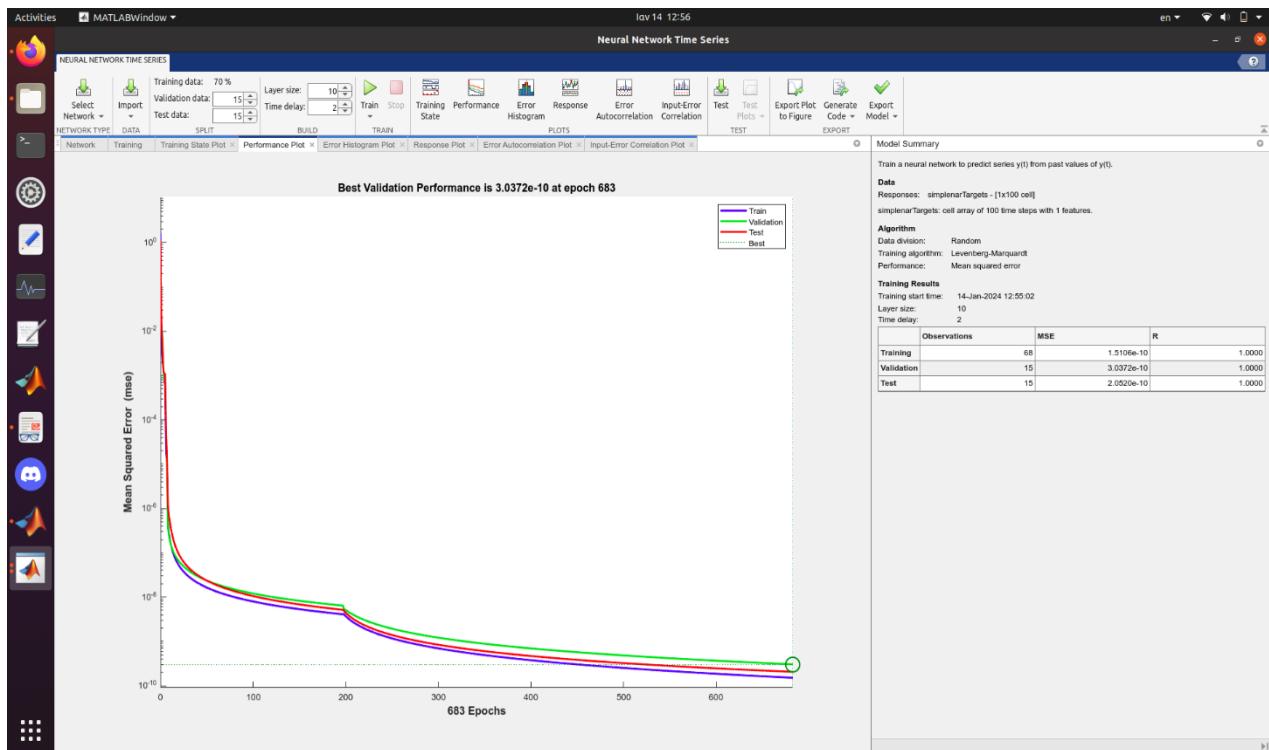
% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

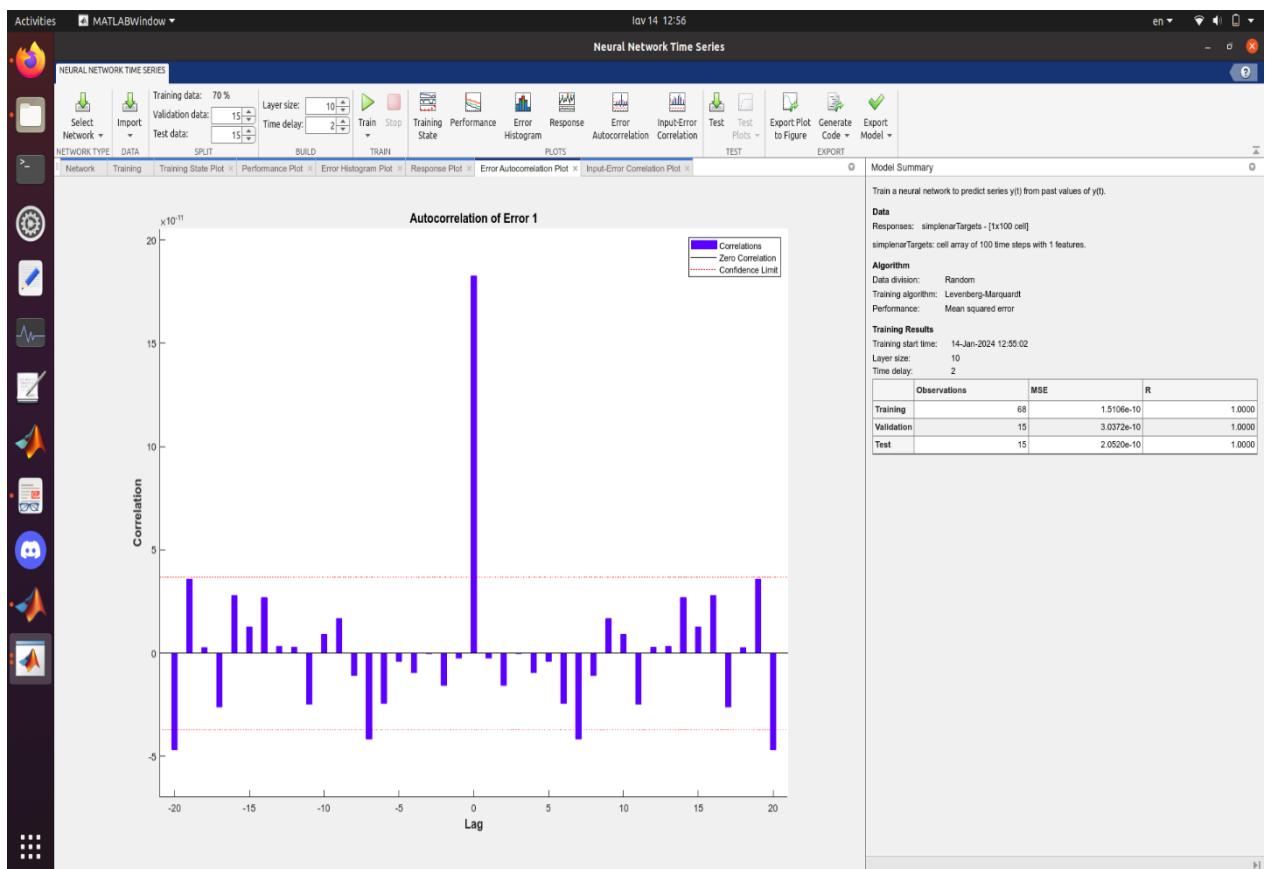
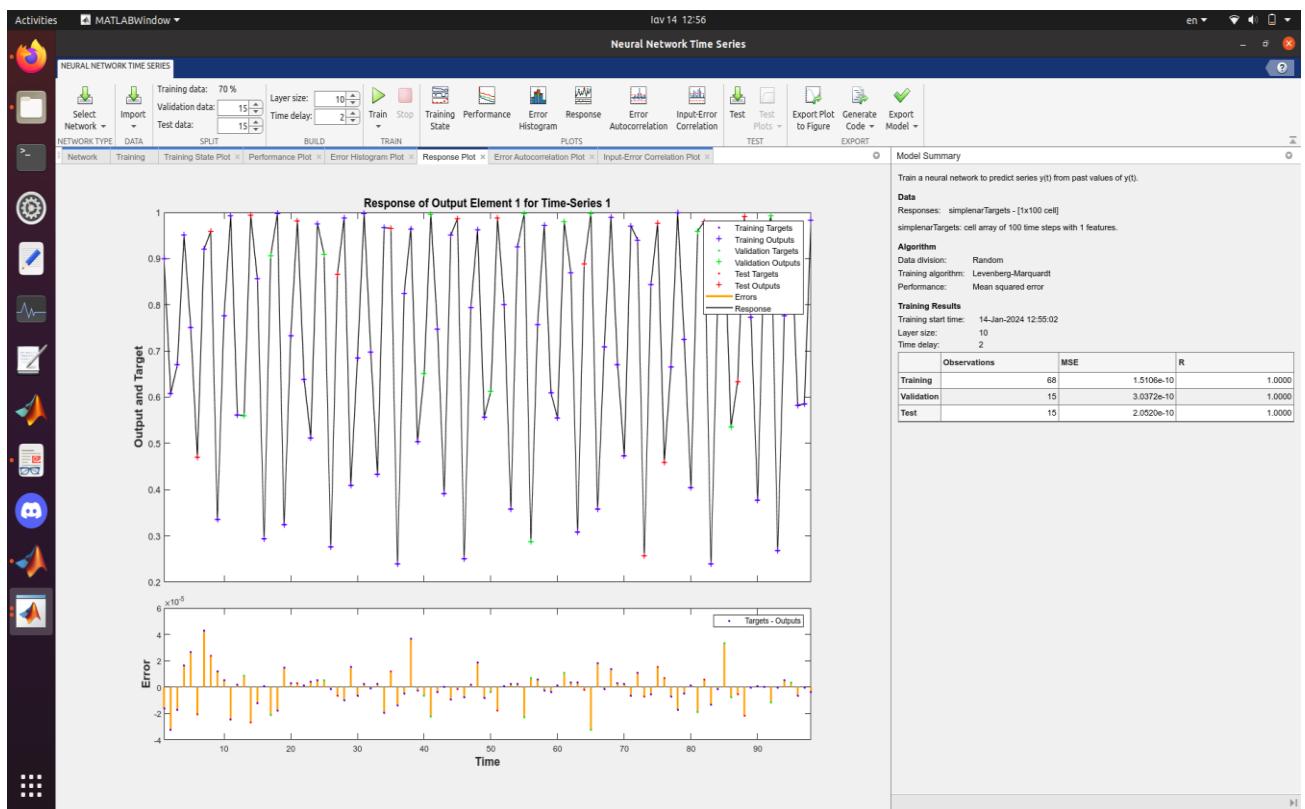
% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y=settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x=settings.xoffset);
end

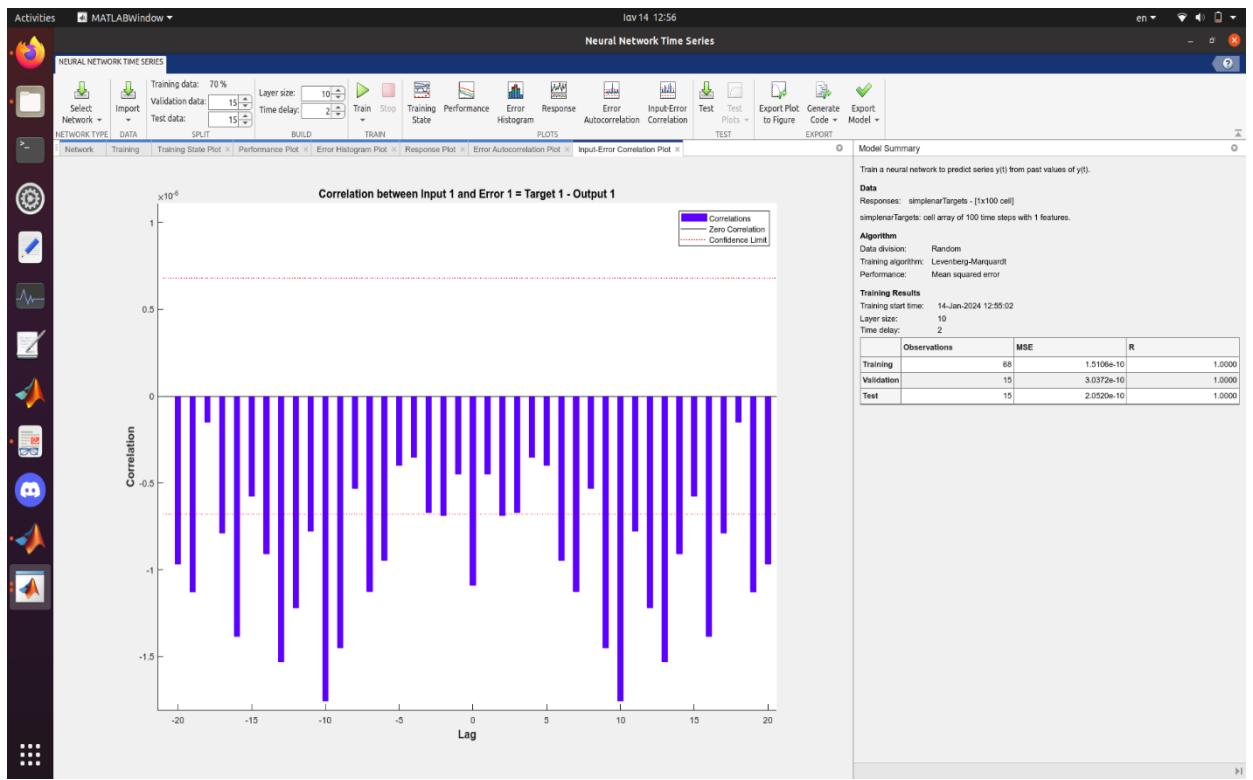
```











### 3. Fitting(nftool)

**Dataset:** Building Energy

Train with Bayesian Regularization

**Αρχείο:** nftool1.m

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 14-Jan-2024 13:02:30
%
% This script assumes these variables are defined:
%
% buildingInputs - input data.
% buildingTargets - target data.

x = buildingInputs;
t = buildingTargets;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainbr'; % Bayesian Regularization backpropagation.
```

```

% Create a Fitting Network
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)

```

### Αρχείο: nftool2.m

```

function [y1] = myNeuralNetworkFunction(x1)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 14-Jan-2024 13:02:32.
%
% [y1] = myNeuralNetworkFunction(x1) takes these arguments:
%   x = 14xQ matrix, input #1
% and returns:
%   y = 3xQ matrix, output #1
% where Q is the number of samples.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [0;0;0;0;0;0;0;-1;-1;0.026;0;0.00877358;0];
x1_step1.gain =
[2;2;2;2;2;2;2;1;1;2.0639834881321;2.25225225225225;2.0668813453644
8;2.13089899432222];
x1_step1.ymin = -1;

% Layer 1
b1 = [0.042830646482639678219;-
0.53151674299525120126;0.43669944030537216628;0.1415014134922687461;0
.27601823390516916357;-0.43862036475124155377;0.3590943543747276423;-

```

```

0.075478834346446491144; -0.35148959371053128953; -
0.19194437632192362653];
IW1_1 = [-0.052718658852218804045 -0.12725133800586033161 -
0.015490941922252449148 -0.039205923294848628313
0.01506590820007881748 0.040781156319187696069 -
0.035333434887407519365 0.031740767529286603721
0.069550497841968553225 0.020236942865760838839
0.58573775040377751289 -0.052676176097390454978
0.079713790598988087943 0.20840909767017445908; 0.36608781008512292221
0.47992233083282220818 0.37197582879846674553 0.42103460270915421537
0.45569730347360148937 0.34737732754908628996 0.21548851153051928886
0.41412238826060776553 1.0682332656667259396 1.1440055077284534413
6.4983911701555037155 -1.0264275104400222283 0.23789510070662367447
0.13861134140143879789; 2.8283870880626751365 -1.2805470726682894878 -
0.74537866118070250643 -0.64543188120331851376 -
0.075561267268419274323 0.14932638401641151749 -2.4142917912831429739
0.0091352286528839227175 0.46075361844359141728 -
0.081904150558614982613 -3.6753280489149946497 -2.5101323637202126449
0.37806638824974220237 2.0528935814504505331; -0.091162552960220771814
-0.12500726368431452906 -0.078993699165000083595 -
0.11131356121913381363 -0.09500703684119558845 -
0.089872575455388015175 -0.11615037821326106016
0.0094572372957089697459 -0.0084375034927992170752 -
0.038829718311234877326 -0.57630366692189438638 -
0.12235727451434402768 0.017006765023968341277
0.044518587399367794133; -2.0339789994602877066 1.1300254539082634775
-1.27967821713691432 1.6608881051860300282 0.54700771811968940117
0.070873020544527390308 -1.4752282506916474247 -
0.18104956254965681728 -0.45757081075896743894 1.2836257699193374116
8.0001521900883751925 4.7465465212740811651 -0.3252097236741804398 -
1.5735992988047027286; 0.41387082973171662736 -0.55845030138234008632
0.21048556749689889545 0.53426649083434529786 0.54763893851508804733
0.48652595291089723029 0.55876434563895527319 -2.859410438547795863
2.537324494149468368 0.32847371954445347964 -0.64723644149212589127
0.27463029654759585352 0.4560241502562630278 -
0.13598007899200156845; -0.39108564679177276746 0.51888471732880336873
0.0090173690318791268478 -0.49705457758045373806 -
0.49578301477636288475 -0.44785578876628007183 -
0.49159483032273765257 2.3004762541932088915 -1.9944828421857123679 -
0.27355717329634338109 0.18320605029762809801 -0.17915340585883521296
-0.2357705686210923024 0.057596862587850190807; 1.1317732769870418874
-2.4187673854350633107 4.0526036206017925778 -1.9587151050809439301 -
0.81648394896118203778 -0.32998756705027793013 0.71697128066996196338
-0.29215838323176190894 -0.17683241473051525272 -
0.70154030707831882552 -5.1164255545366703615 1.0647288220094028866 -
1.1390356603861842011 1.4086895679935436121; 2.3848825777702438167 -
1.4170471292393573837 1.4280790765227022732 -1.8886959891990557914 -
0.54443427350440798396 0.03823325772658783835 1.7564304484173520748
0.012198089745816634646 0.41068883481487711284 -1.5810021222115320239
-9.268352553763151036 -5.5239478895712341711 0.15385912473137816625
1.5181403141508811316; 0.28020548543585693801 0.26900251890214810224
0.14865491753113951878 0.12026758229763286068 0.020468416020243056924
-0.018997303257723679309 0.14012026465682017595 -
0.080051026795497248756 -0.15378662529553935312 -
0.047811555061492018448 -0.19029747383778314318
0.28908130838178308641 -0.18581786892032126079 -
0.43320888632376908234];

```

```

% Layer 2
b2 = [-0.7899694957591407718; 0.99312017281135711588; -
0.020756913354455629878];

```

```

LW2_1 = [2.0880230644622539415 0.068923233250362772595 -
0.15828235111591085205 2.9004306007693076985 -1.7803365555940533493 -
1.5239926344917502643 -2.0827073656170980165 -0.18988571081225052994
-1.7098430627055674069 1.3697051028307738463;0.37447998801662557566 -
0.099968544207471227159 0.041364701892042159137 -
1.6911512744792003282 -0.51840244910233368447 -0.15566219178048687488
-0.18491044068845913229 0.013544691224686577699 -
0.4831326906312530256 0.010290201586456765651;-2.4823276473677808873
-0.20452123633956156845 0.18228697256313416974 -1.7763994119250463566
0.58568171667562018978 0.080602598971849043208 0.11256617231763142717
0.15304806024497094596 0.57215675299981805235 -
1.8070550268705558583];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = [4.6518443399847;3.33333333333333;3.10405827683171];
y1_step1.xoffset = [0.259145;0.075;0.0818182];

% ===== SIMULATION =====

% Dimensions
Q = size(x1,2); % samples

% Input 1
xp1 = mapminmax_apply(x1,x1_step1);

% Layer 1
a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

% Layer 2
a2 = repmat(b2,1,Q) + LW2_1*a1;

% Output 1
y1 = mapminmax_reverse(a2,y1_step1);
end

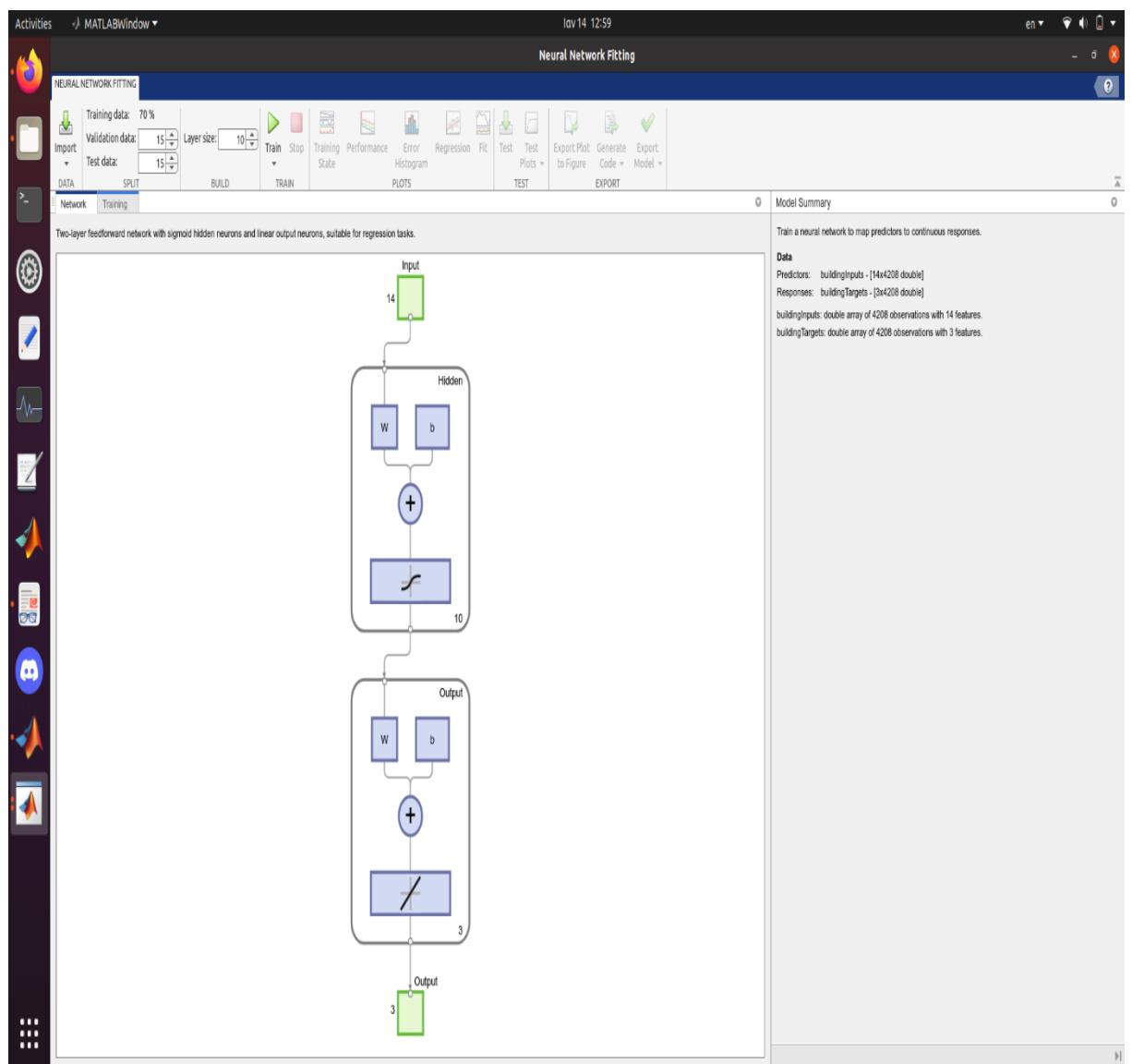
% ===== MODULE FUNCTIONS =====

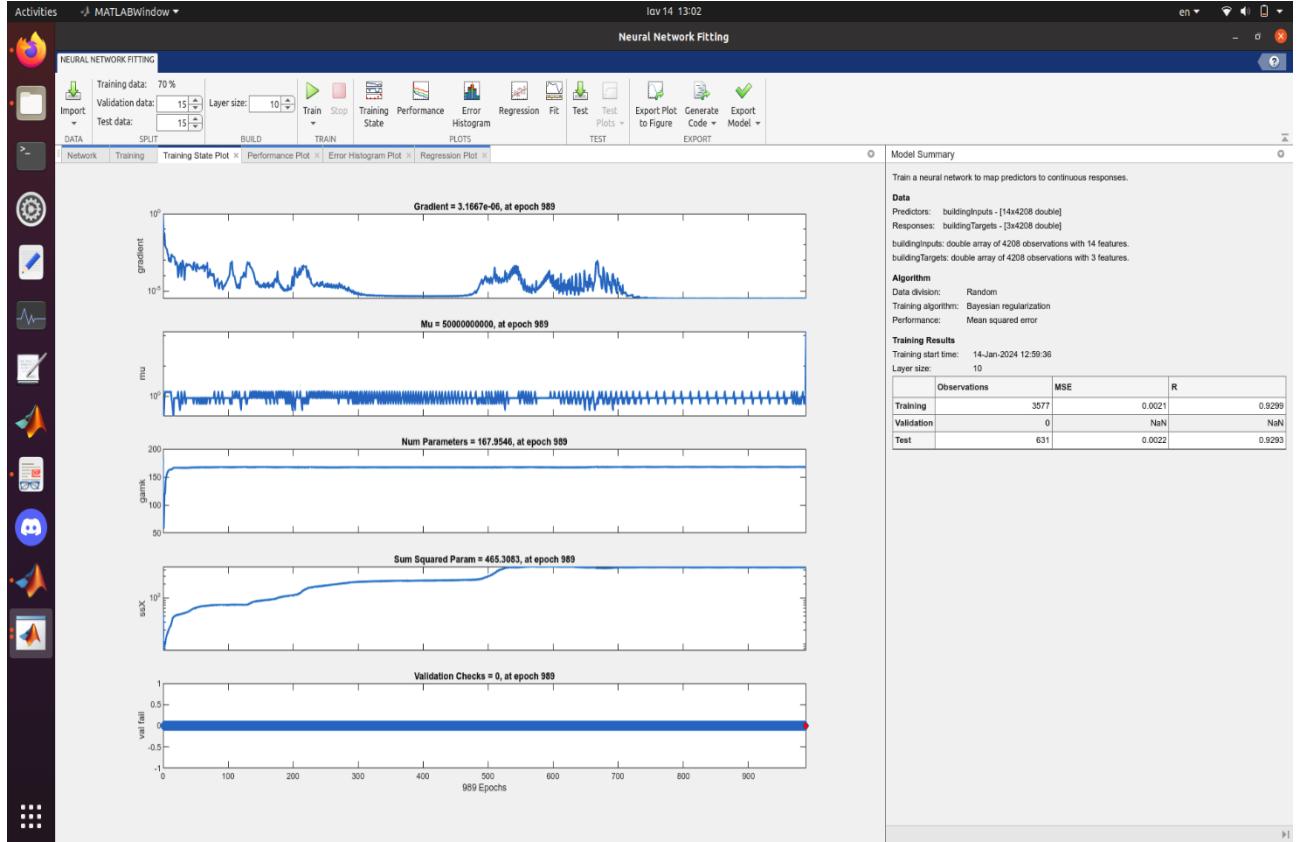
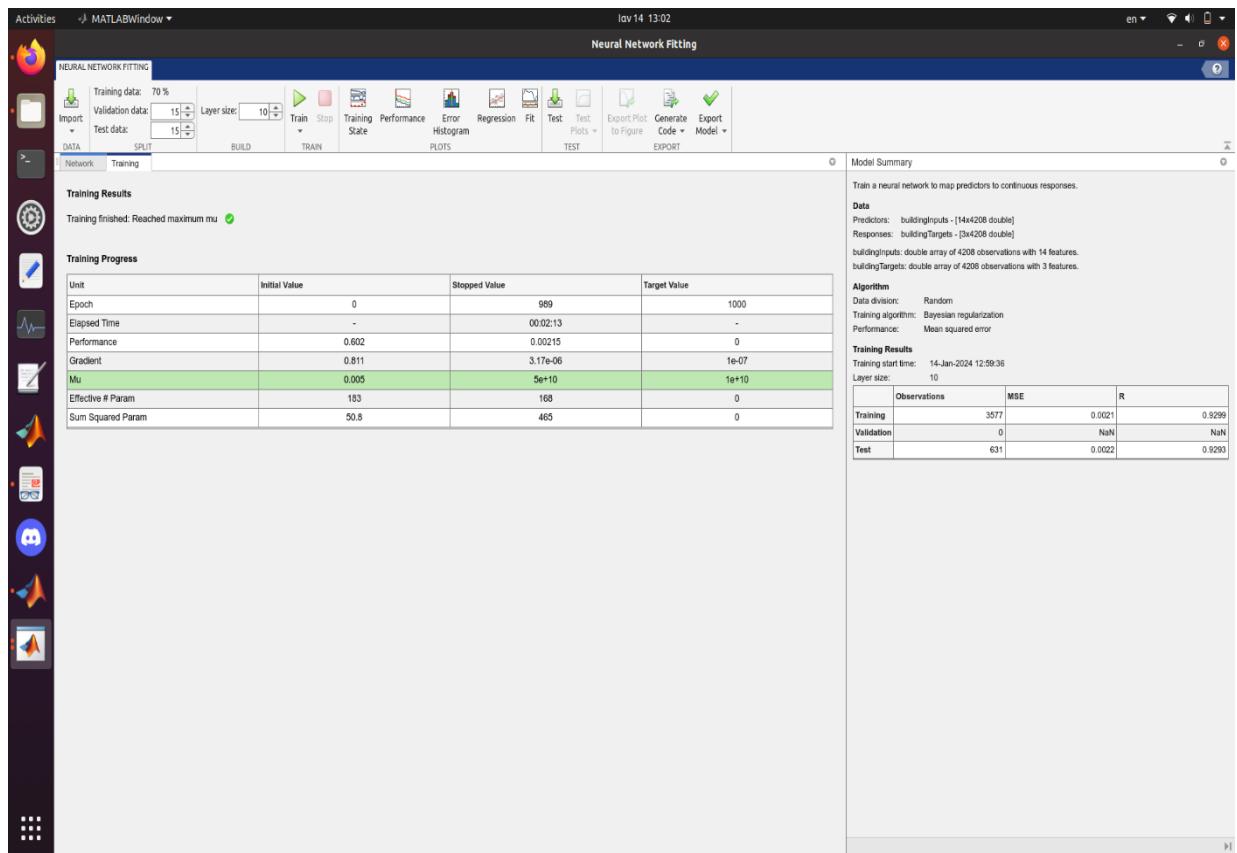
% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

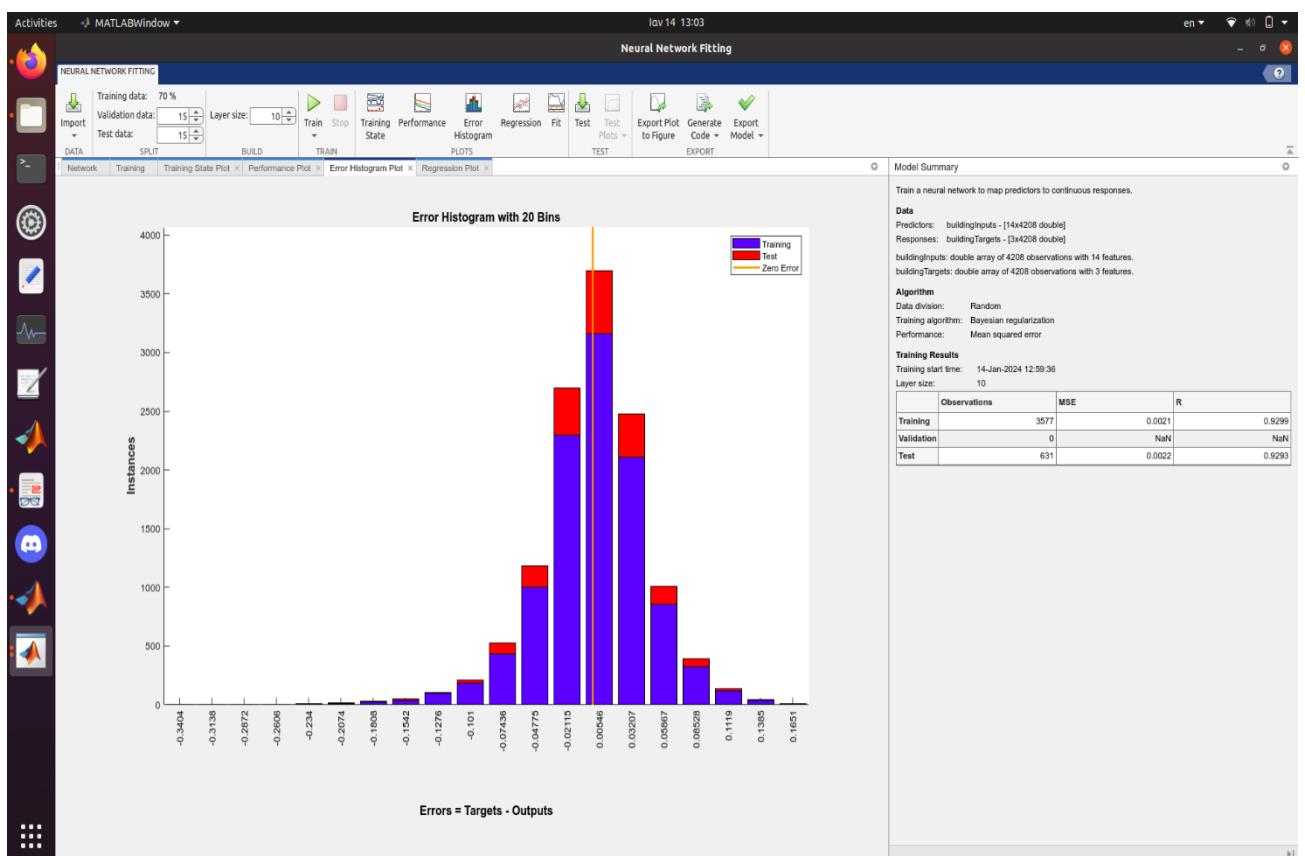
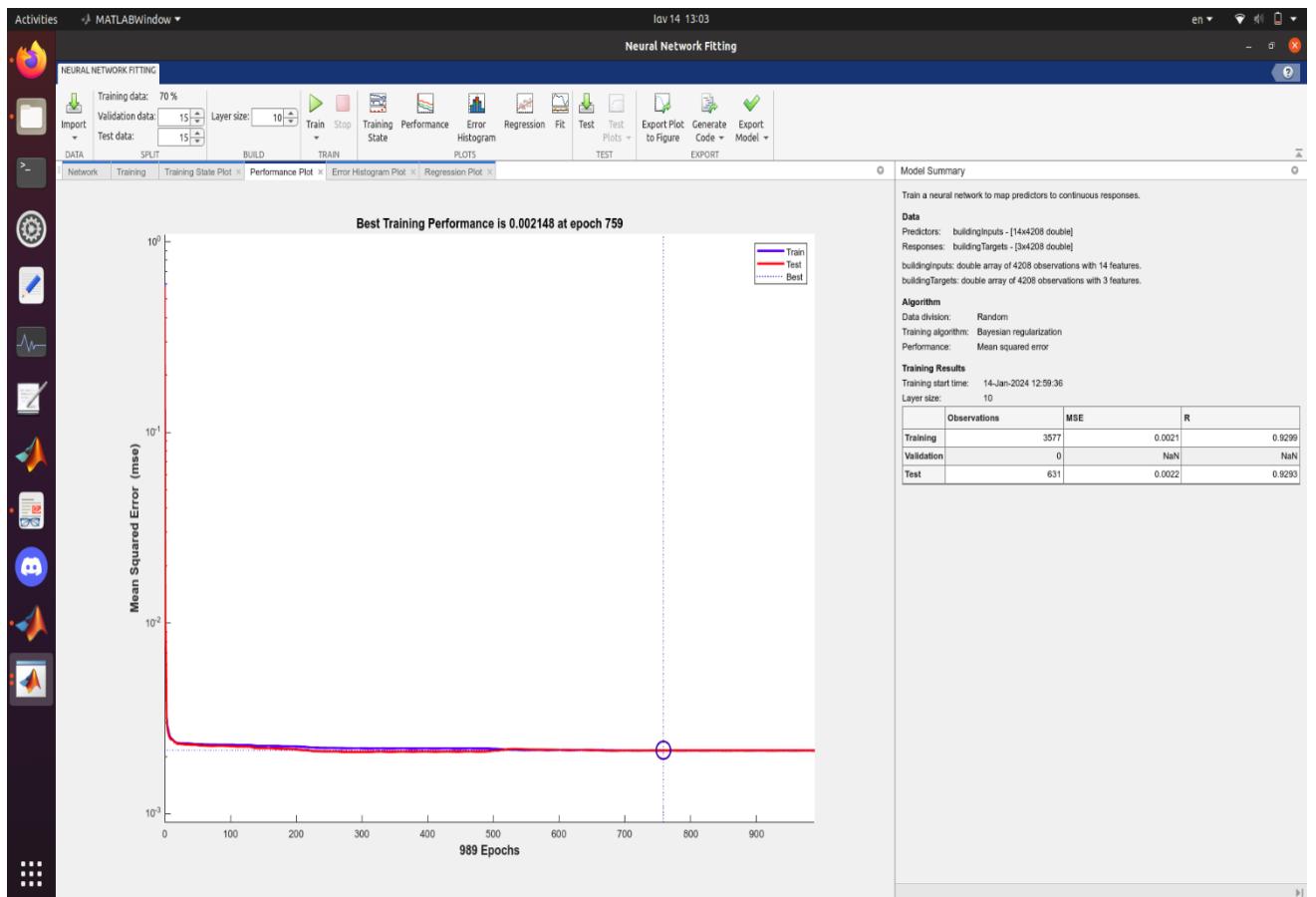
% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

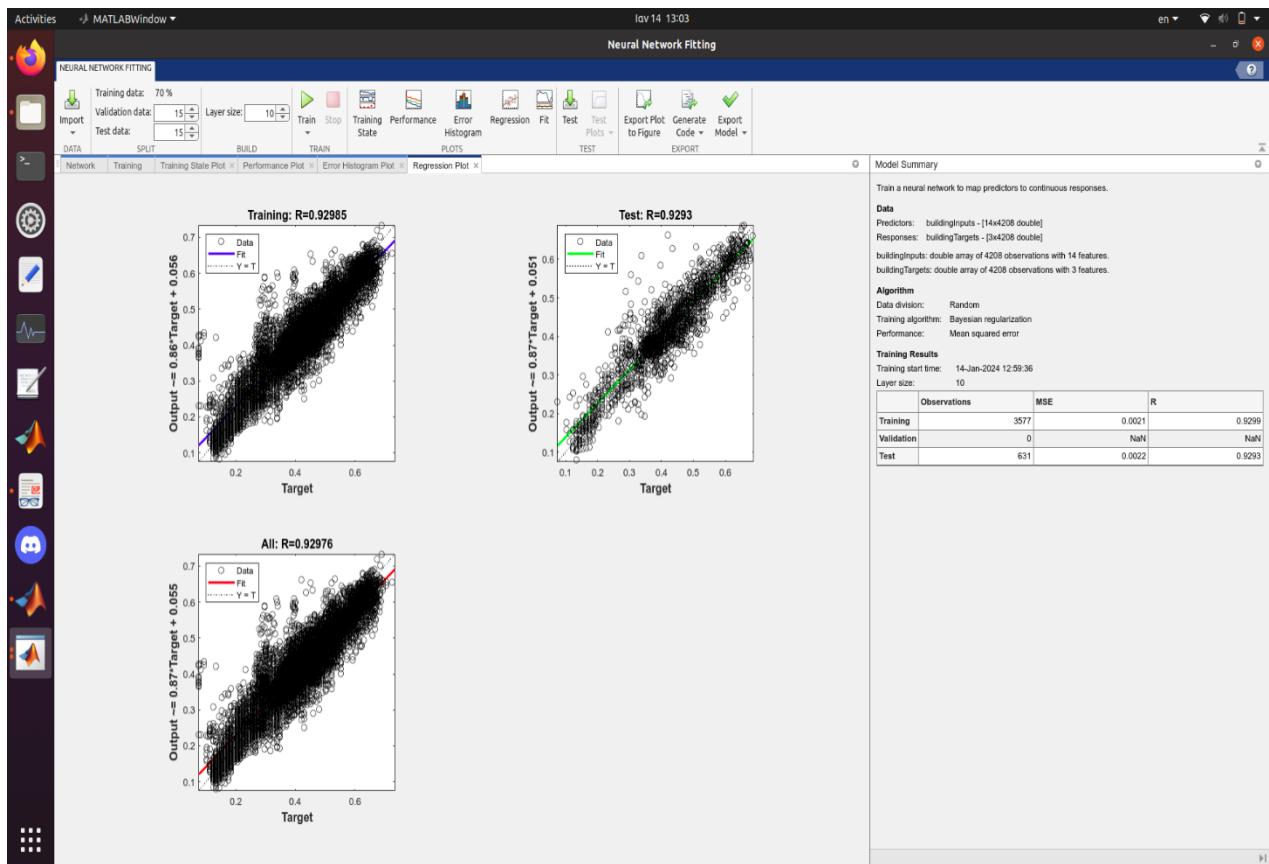
% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```









## 4. Pattern Recognition(nprtool)

**Dataset:** Wine Data Set

**Αρχείο:** nprtool1.m

```
% Solve a Pattern Recognition Problem with a Neural Network
% Script generated by Neural Pattern Recognition app
% Created 14-Jan-2024 13:05:28
%
% This script assumes these variables are defined:
%
% wineInputs - input data.
% wineTargets - target data.

x = wineInputs;
t = wineTargets;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainscg'; % Scaled conjugate gradient backpropagation.

% Create a Pattern Recognition Network
hiddenLayerSize = 10;
```

```

net = patternnet(hiddenLayerSize, trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotconfusion(t,y)
%figure, plotroc(t,y)

```

## Αρχείο: nprtool2.m

```

function [y1] = myNeuralNetworkFunction(x1)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 14-Jan-2024 13:05:30.
%
% [y1] = myNeuralNetworkFunction(x1) takes these arguments:
%   x = 13xQ matrix, input #1
% and returns:
%   y = 3xQ matrix, output #1
% where Q is the number of samples.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset =
[11.03;0.74;1.36;10.6;70;0.98;0.34;0.13;0.41;1.28;0.48;1.27;278];
x1_step1.gain =
[0.526315789473684;0.395256916996047;1.06951871657754;0.1030927835051
55;0.0217391304347826;0.689655172413793;0.421940928270042;3.773584905
66038;0.630914826498423;0.170648464163823;1.6260162601626;0.732600732
600733;0.0014265335235378];

```

```

x1_step1.ymin = -1;

% Layer 1
b1 = [1.6226192160865056113;-
1.2283874968223928992;1.2738752690879489027;0.35683337951829868784;0.
7125483664984280944;-1.3109262153066296719;0.8414679453067011039;-
0.76669427307651805137;-1.1710482603290672809;1.8184946018128624878];
IW1_1 = [-0.69249104692166429942 0.71440982970999411528 -
0.16260623533365867188 -0.16343797873689019928 -
0.40469009134789507565 -0.45228605703861896048 0.3675318051830376298
-0.20965131971077333173 -0.82386127404930609153
0.09831998575402493834 0.26675215641003940936 0.53610194534349508988
-0.37094905614845008479;0.2669973094231708477 0.23499935799268525938
0.077579011085013166449 0.94334216878842891418 -
0.49756195745788589502 -0.12018447694442081775 -1.0885324240418026154
-0.38343190927034304094 -1.1011068577272620228 1.5132973112606173949
-0.68870665196697244959 -1.15594378639681028 0.58646309643052774163;-
0.36080313516915912819 0.3513251686656774786 0.068350879577868783876
0.37473775796834812857 -0.27069121870013013176 0.49491120244248554272
-0.40645055758888148922 0.17649786974202397172 -0.5847072113026533513
-0.65076904934361901045 -0.45735381918098122478 -
0.44252086536027035457 0.00992505922324691571;-0.28709048693607364111
-0.63514318806177005872 -0.26377050480919073117
0.24132161965957049654 0.10694129156984268081 -0.55075808003508841537
1.4546546251791523385 0.57109392897687194512 0.21970311561995264915 -
1.4921157577862238508 1.4130373476196960159 0.61377401168271827903 -
0.75684766566596750614;2.3367546588640006888 0.54557927980119813594
1.8997342876009926105 -2.5287167473718028532 0.2125804407436896748 -
0.095636315168737509596 1.3038287343853138811 0.14119260938008681983
0.55147408483756044095 0.46588892458407460406 -0.30640515953291908025
1.4550006737674903157 2.4679008879541615151;0.52090696447997242302
0.73399363324559097066 1.4652578209058850778 0.15013269351824273801
0.44209401888768251609 0.90356292872382337755 -2.9296567298430060688
-0.51573090018842193683 -0.57005732730685798604 3.6204298088863962057
-1.7675716300537436254 -1.7280542820849404961
0.37429206321356445208;1.5345292451090382091 1.1302590955186346822
1.2743282962418440718 -1.9023629446553531608 0.47343044999965805397 -
0.015288100163414284488 1.4570622545687434979 0.046589305389605033259
0.15978064840179909134 -0.32367666651642168496 -
0.58539447958804835537 0.93371965963879355321 2.5773406763499093763;-
0.86664609245700596762 0.16837114984703221987 -0.82103547131977760998
0.73455613091016191962 -0.94137059744626094826 -
0.90875017460865192831 0.8214786218269728435 0.13362500725735990126 -
0.0080923824174770619955 -0.24399594796229745475 -
0.2246627522902106544 1.0183213523395644451 -0.53574405335181163323;-
0.78019014100508554677 -0.37071838256568034886
0.020445744495666690055 -0.073225535917385092821
0.86729863984870003701 0.41628166353917922526 -0.26196557783855123347
-0.75967353592942354901 0.81964345562933271783 0.31940559868336709792
-0.18701682399353686259 -0.11933466049130243836 -
0.4622670110052989334;0.87845050471212171583 0.4197444305152717714
0.38814094064056720024 -0.36273469485355008812 -
0.67021533412397515939 -0.8188900273033624666 -0.6161584341098810258
0.28275473921468985417 -0.10363168147853499268 0.61953688117830729709
-0.6914562950324419921 -0.6199021012816495757 -
0.16975970610631110902];

% Layer 2
b2 = [0.32995676754684455378;-0.69653563603048185282;-
0.4950137527382353908];
LW2_1 = [-0.45035364401609062623 -1.0393958708915422839 -

```

```

0.62575901897619923275 0.009375994584403073151 4.7939911982672818169
-1.1232908061650079823 3.7150739260230181138 -0.36825662930025515607
-0.46564126307082992495 1.1164861369714098593;-0.64166050043067379693
-1.2836532198282248896 -0.61797093641020972576 1.6714092495147512274
-3.7040291458067842179 -4.2713386153636445641 -3.2141606913101519183
1.3352352554572237686 0.68805022361562351296 -
0.80625476168913889907;-0.31803781769160743842 2.0999738739944899812
-0.03732337299818133669 -2.2461457086057445842 -1.0772487510831103563
5.4362743078123489937 -0.7488509378380293624 -1.157632220814427404 -
0.59646318303403311756 0.016673268571487452616];

% ===== SIMULATION =====

% Dimensions
Q = size(x1,2); % samples

% Input 1
xp1 = mapminmax_apply(x1,x1_step1);

% Layer 1
a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

% Layer 2
a2 = softmax_apply(repmat(b2,1,Q) + LW2_1*a1);

% Output 1
y1 = a2;
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Competitive Soft Transfer Function
function a = softmax_apply(n,~)
if isa(n,'gpuArray')
    a = iSoftmaxApplyGPU(n);
else
    a = iSoftmaxApplyCPU(n);
end
end
function a = iSoftmaxApplyCPU(n)
nmax = max(n,[],1);
n = bsxfun(@minus,n,nmax);
numerator = exp(n);
denominator = sum(numerator,1);
denominator(denominator == 0) = 1;
a = bsxfun(@rdivide,numerator,denominator);
end
function a = iSoftmaxApplyGPU(n)
nmax = max(n,[],1);
numerator = arrayfun(@iSoftmaxApplyGPUHelper1,n,nmax);
denominator = sum(numerator,1);
a = arrayfun(@iSoftmaxApplyGPUHelper2,numerator,denominator);
end
function numerator = iSoftmaxApplyGPUHelper1(n,nmax)

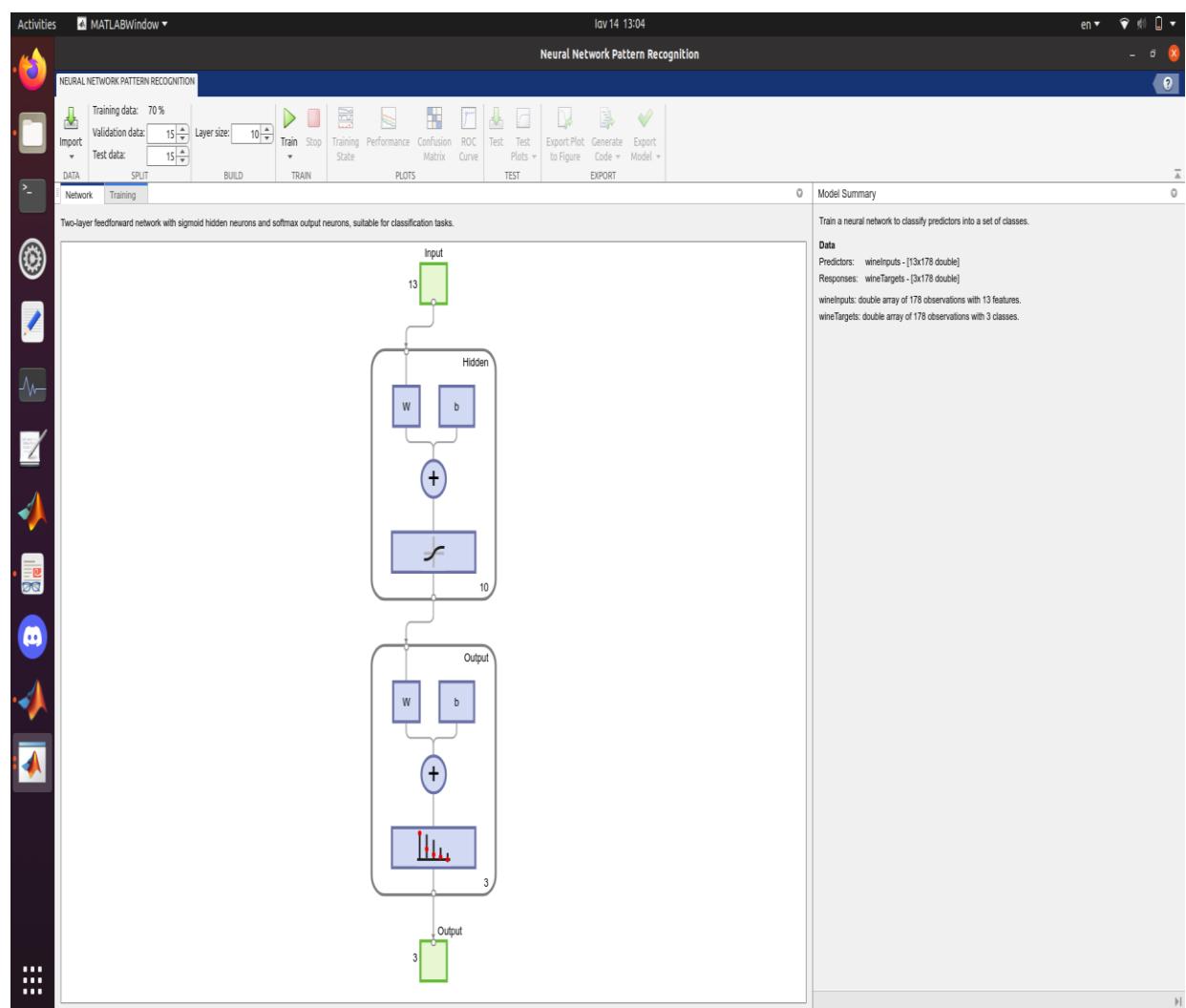
```

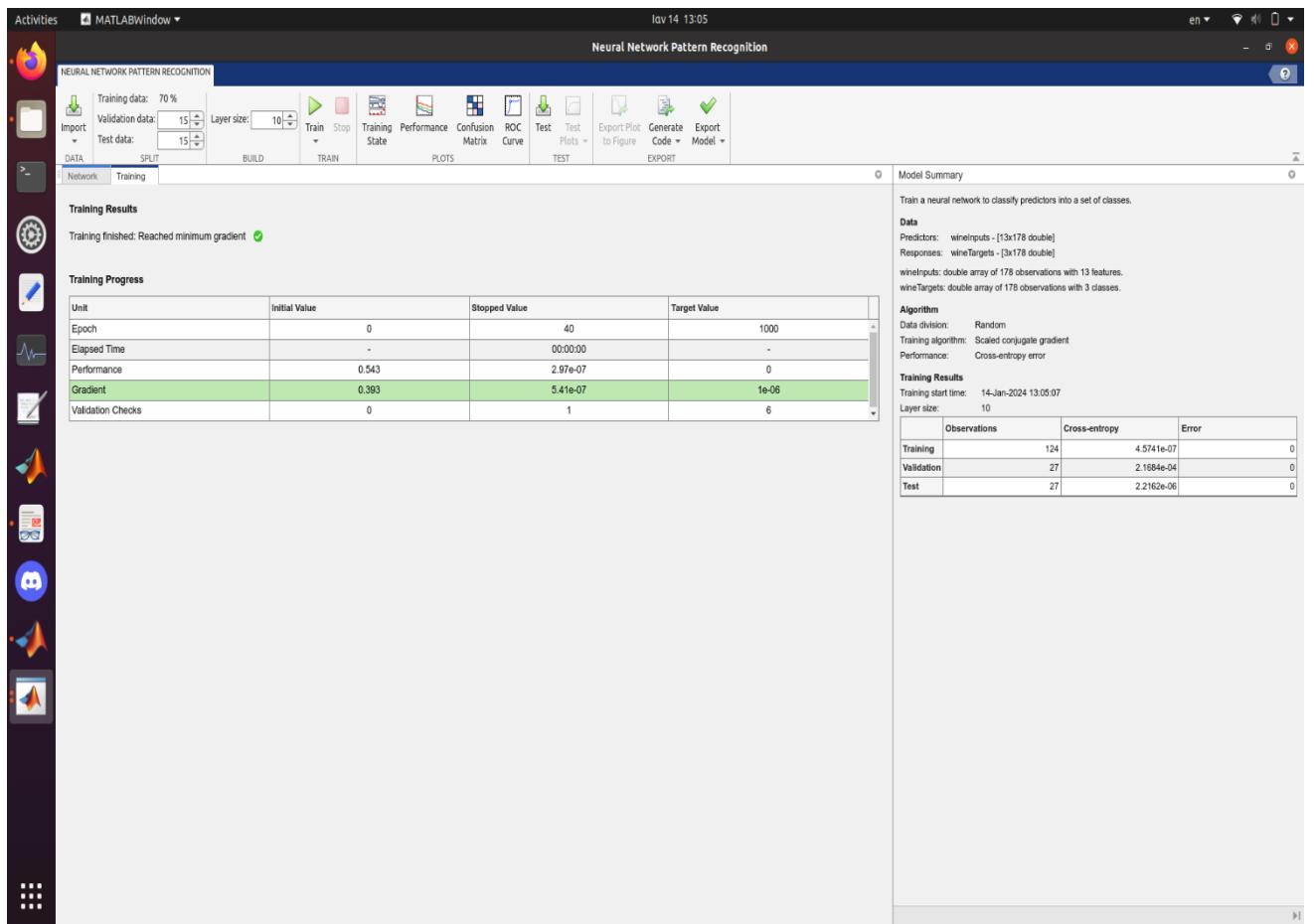
```

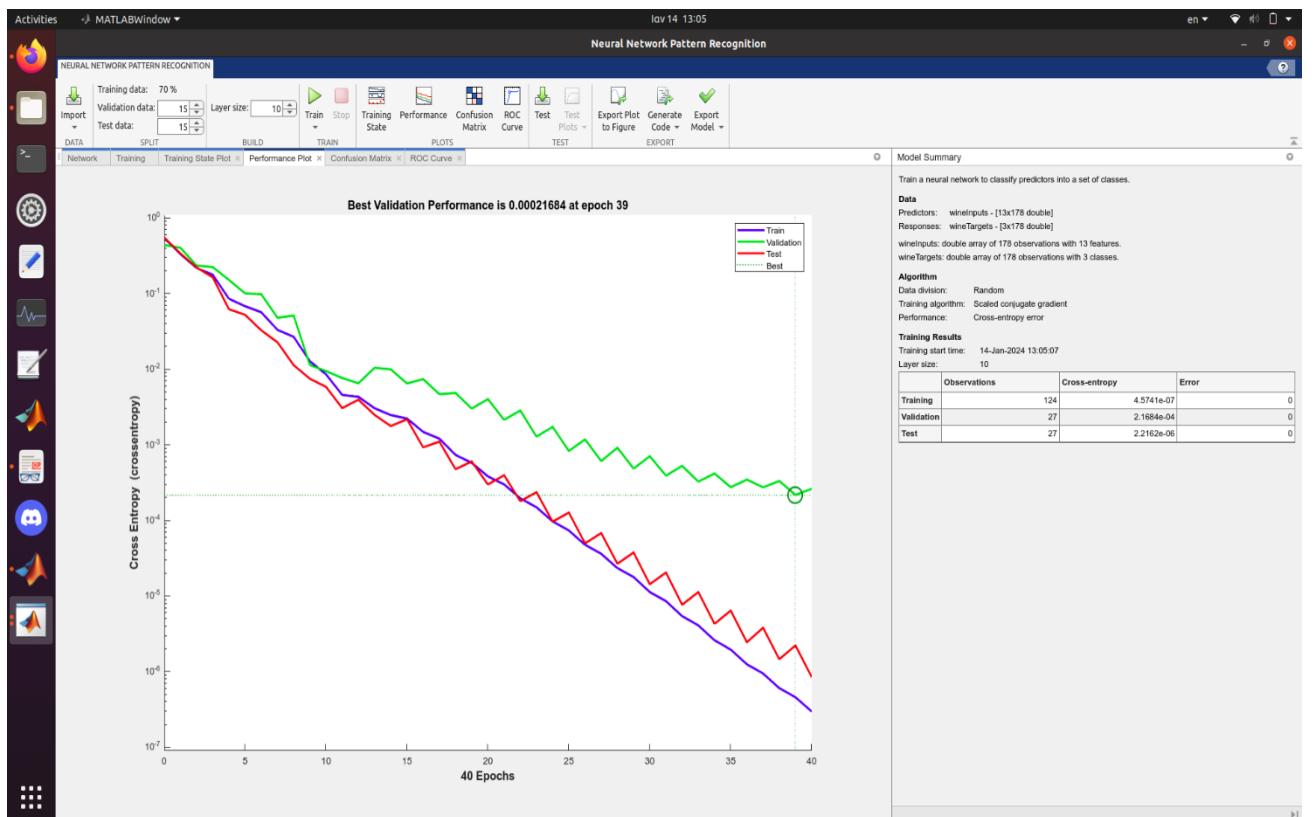
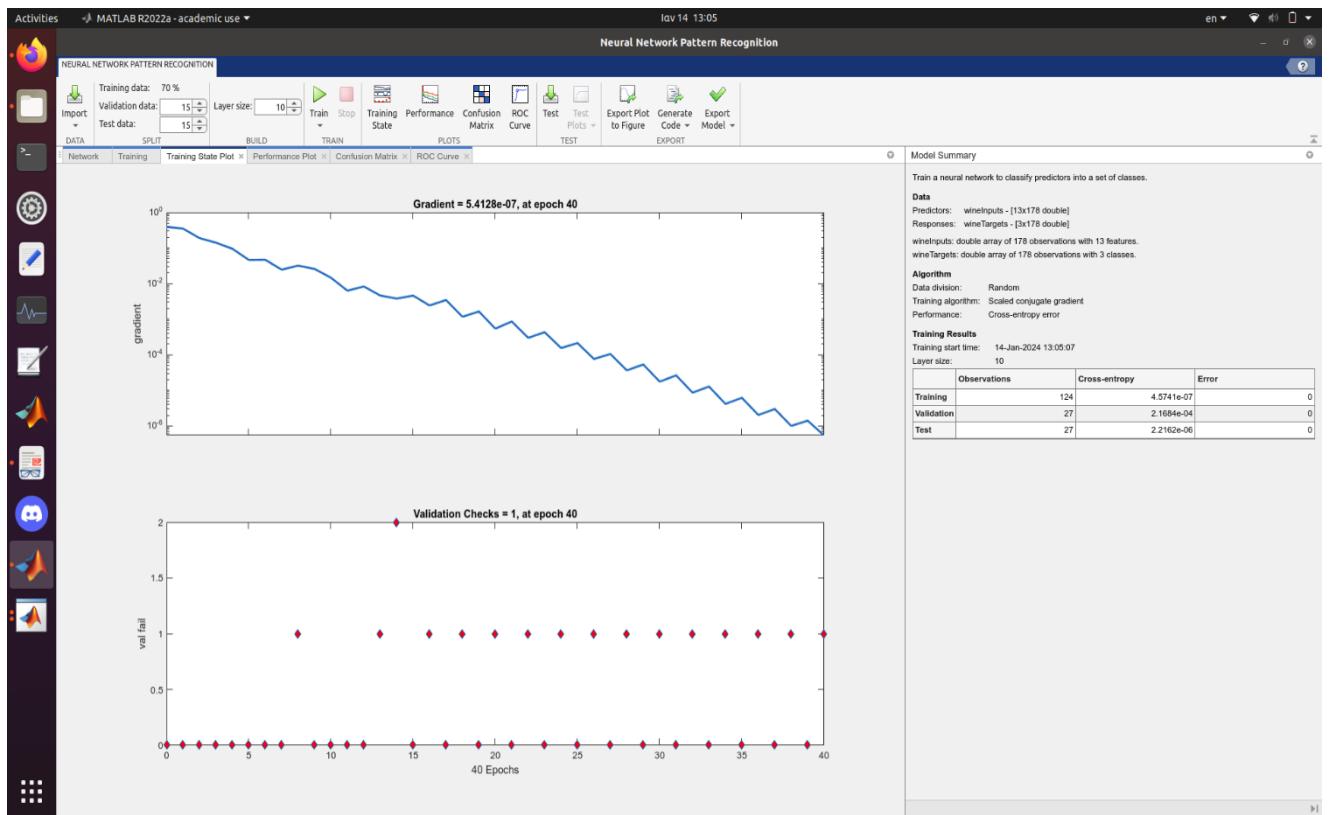
numerator = exp(n - nmax);
end
function a = iSoftmaxApplyGPUHelper2(numerator,denominator)
if (denominator == 0)
    a = numerator;
else
    a = numerator ./ denominator;
end
end

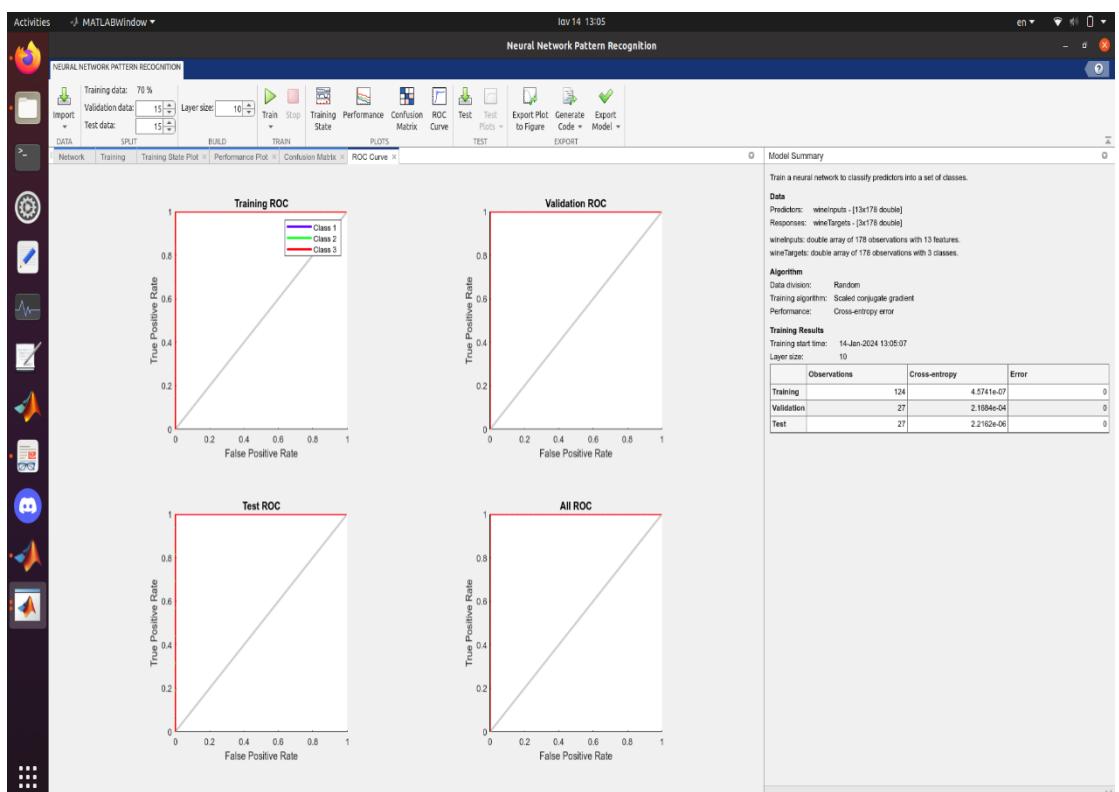
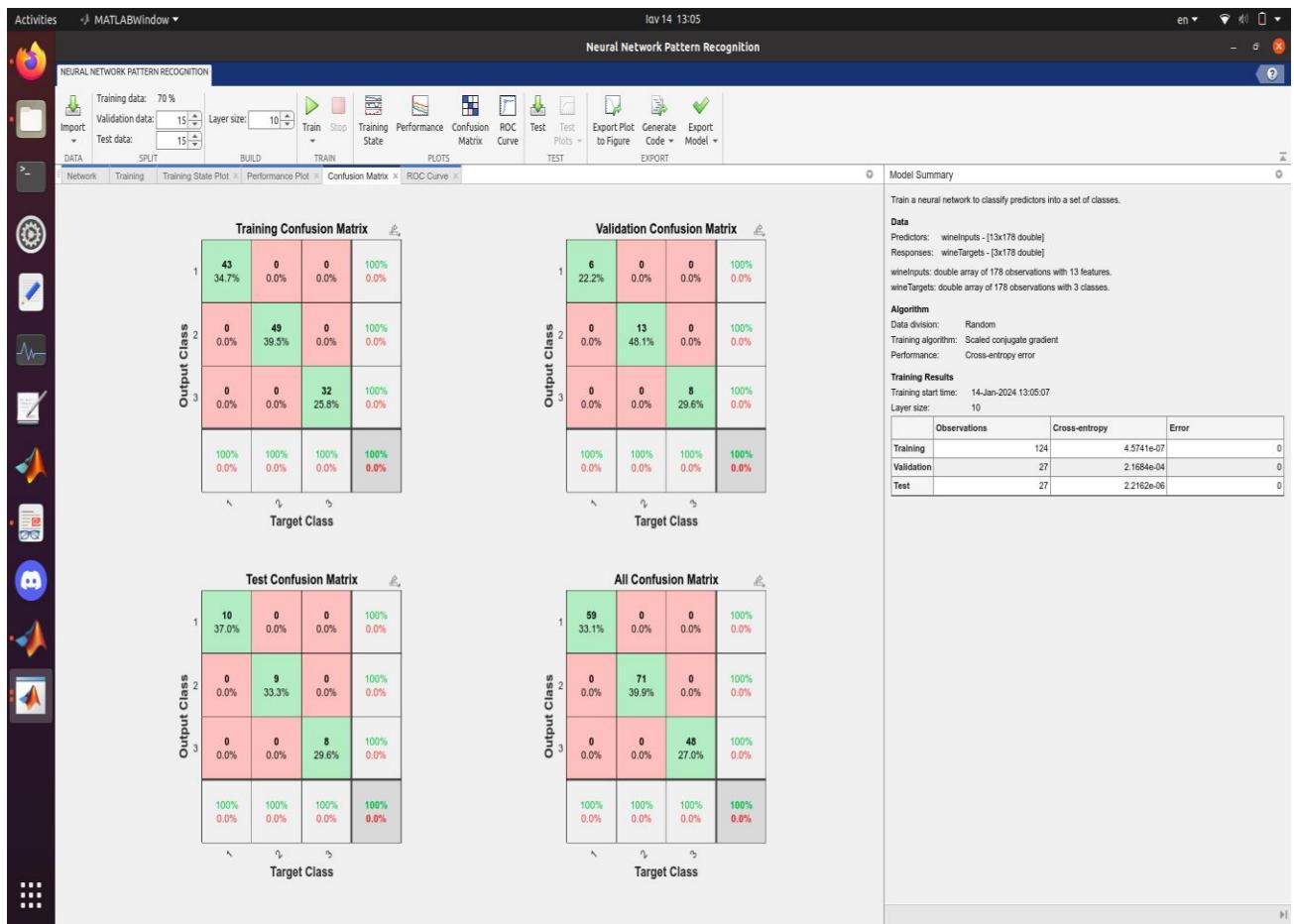
% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

```









### **Σχολιασμός αποτελεσμάτων:**

Στο ερώτημα A2 της εργασίας δημιουργούμε νευρωνικά δίκτυα με τη βοήθεια 4 εφαρμογών GUI του Matlab. Αναλυτικότερα, χρησιμοποιούμε τα εξής: nftool για Function Fitting, nprtool για Pattern Recognition, nctool για Clustering, ntstool για Time-series analysis. Αντίστοιχα, χρησιμοποιούμε τα εξής datasets που υπάρχουν έτοιμα στο Matlab: Iris flowers, Simple Nar, Building Energy και Wine Data Set. Μέσω των παραπάνω διεπαφών εκπαιδεύουμε τα δίκτυα που δημιουργούμε και οδηγούμαστε σε ενδιαφέροντα συμπεράσματα, μέσω των διαγραμμάτων που δημιουργούνται.

Ειδικότερα, στο 1<sup>o</sup> παράδειγμα παρουσιάζονται τα εξής διαγράμματα: SOM Neighbor Weight Distances, weights from every input, hits, SOM weight positions και SOM neighbor connections. Συνοπτικά, παρουσιάζονται δηλαδή οι θέσεις των βαρών των νευρώνων στο χάρτη SOM, το πόσο απέχουν τα βάρη στο χάρτη SOM, τα βάρη που έχουν ανατεθεί σε κάθε είσοδο και τις συνδέσεις γειτονίας μεταξύ νευρώνων στο χάρτη SOM.

Στο 2<sup>o</sup> παράδειγμα έχουμε τα εξής γραφήματα: Gradient, Mu, Validation Checks, best validation performance, error histogram. Τα συγκεκριμένα έχουν αναλυθεί για το A1 ερώτημα της εργασίας. Επίσης, βλέπουμε τη σχέση Error-Time, το Response of output element 1 for Time Series 1, το Autocorrelation of Error 1 και το Target 1 – Output 1. Απεικονίζεται δηλαδή η διαφορά μεταξύ της επιθυμητής και της πραγματικής εξόδου για την 1<sup>η</sup> χρονοσειρά, η αλλαγή της εξόδου 1 σε σχέση με την 1<sup>η</sup> χρονοσειρά και η συσχέτιση του σφάλματος για την 1<sup>η</sup> χρονοσειρά με τον εαυτό του.

Στο 3<sup>o</sup> παράδειγμα παρουσιάζονται διαγράμματα που έχουν όλα σχολιαστεί αναλυτικά για το A1 ερώτημα της εργασίας.

Τέλος, για το 4<sup>o</sup> και τελευταίο παράδειγμα, παρατηρούμε κάποια πρόσθετα διαγράμματα σε σχέση με το 3<sup>o</sup> παράδειγμα. Αναλυτικότερα, παρουσιάζονται 4 Πίνακες Σύγχυσης. Οπτικοποιούν τη σχέση Output Target και Class για την εκπαίδευση, τον έλεγχο, την επικύρωση, αλλά και το συνδυασμό τους. Ακόμη, έχουν δημιουργηθεί για τις 4 παραπάνω έννοιες ROC καμπύλες. Οι ROC καμπύλες παρουσιάζουν την απόδοση του μοντέλου σε σχέση με το ρυθμό των False Positive και True Positive σε διάφορα επίπεδα κατωφλίου για τα σετ εκπαίδευσης, επικύρωσης και ελέγχου. Παράλληλα, οι 4 Πίνακες Σύγχυσης παρουσιάζουν τον αριθμό των προβλέψεων σε κάθε κατηγορία και τον πραγματικό αριθμό παρατηρήσεων. Η οπτική αναπαράσταση της σχέσης εξόδου-στόχου βοηθάει στην κατανόηση του σφάλματος του μοντέλου.

Αξίζει να τονιστεί το γεγονός ότι σε κάθε ένα από τα 4 παραδείγματα του ερωτήματος A2, κατά το Export δημιουργούνται 2 αρχεία. Συγκεκριμένα, το 1<sup>o</sup> αρχείο δημιουργεί και εκπαιδεύει το νευρωνικό δίκτυο. Ενώ, το δεύτερο αρχείο υλοποιεί τη συνάρτηση myNeuralNetworkFunction, που επιτελεί διάφορες λειτουργίες και γενικότερα simulation.

## Μέρος Β

### Deep Network Designer

**Σημείωση:** Τα B1 και B2 παρουσιάζονται ως 1. Απλά, γίνεται παρουσίαση 2 διαφορετικών παραδειγμάτων.

#### Παράδειγμα 1

**Βήματα:**

1. Άνοιγμα του App Network Designer
2. New Blank Network
3. Διαλέγουμε από το πλαίσιο αριστερά της οθόνης τα κατάλληλα layers, και όπου κρίνεται απαραίτητο αλλάζουμε τις παραμέτρους του κάθε στρώματος μέσω του πλαισίου δεξιά της οθόνης.

**Στρώματα:**

imageInput (το παράδειγμα ασχολείται με εικόνες)

conv

relu

conv

relu

fc

fc

softmax (μέτρηση πιθανοτήτων)

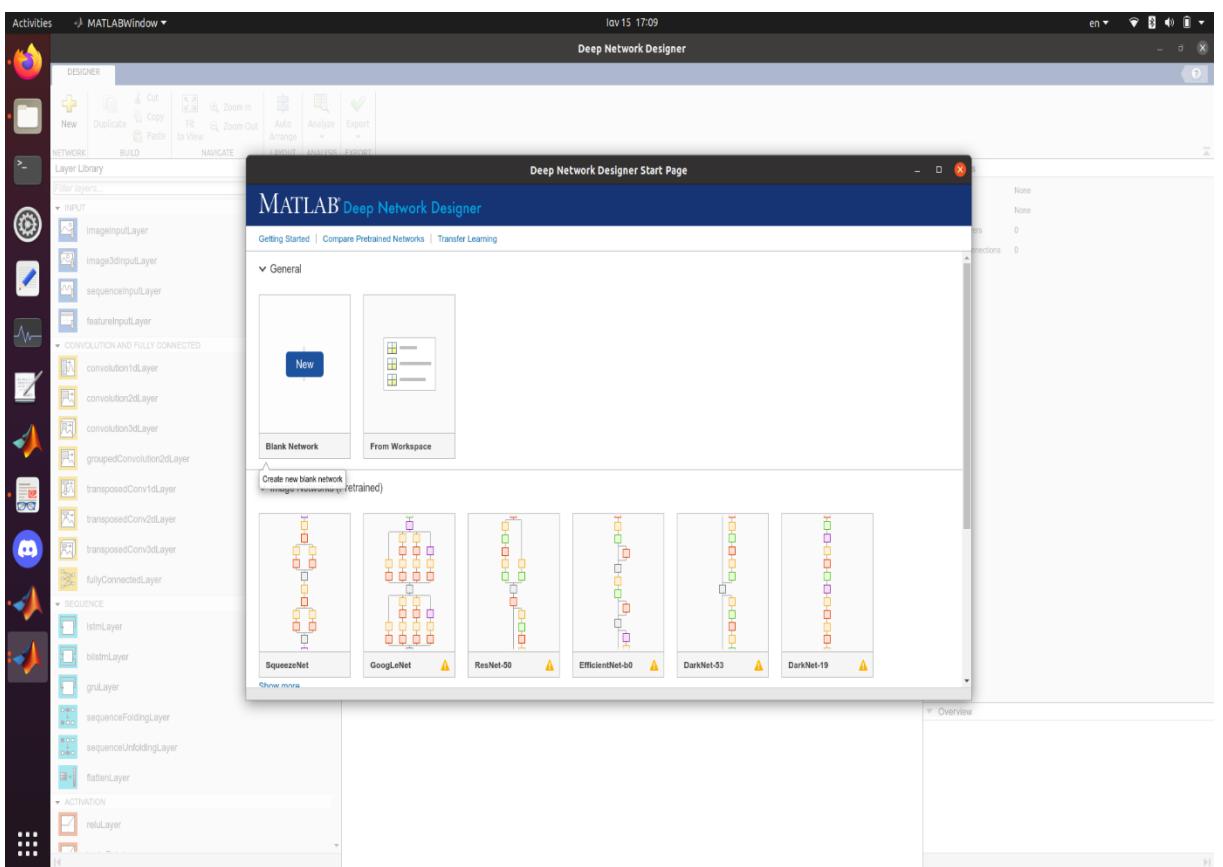
classoutput (πρόβλημα κατηγοριοποίησης)

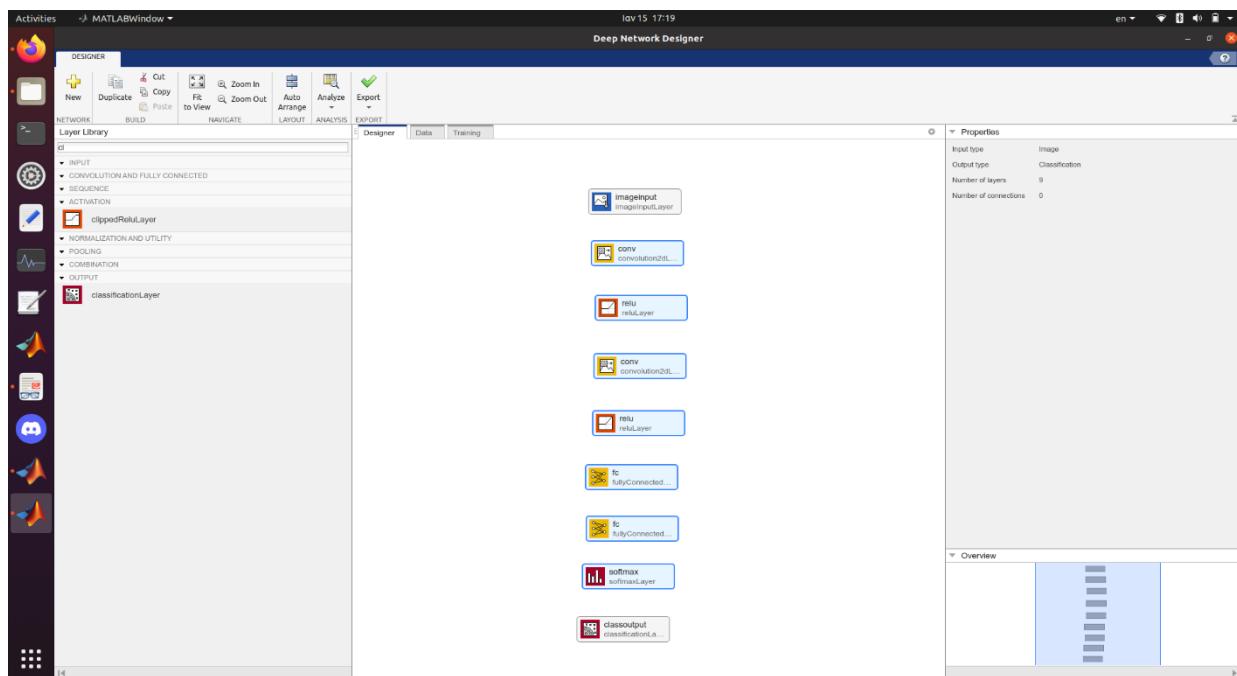
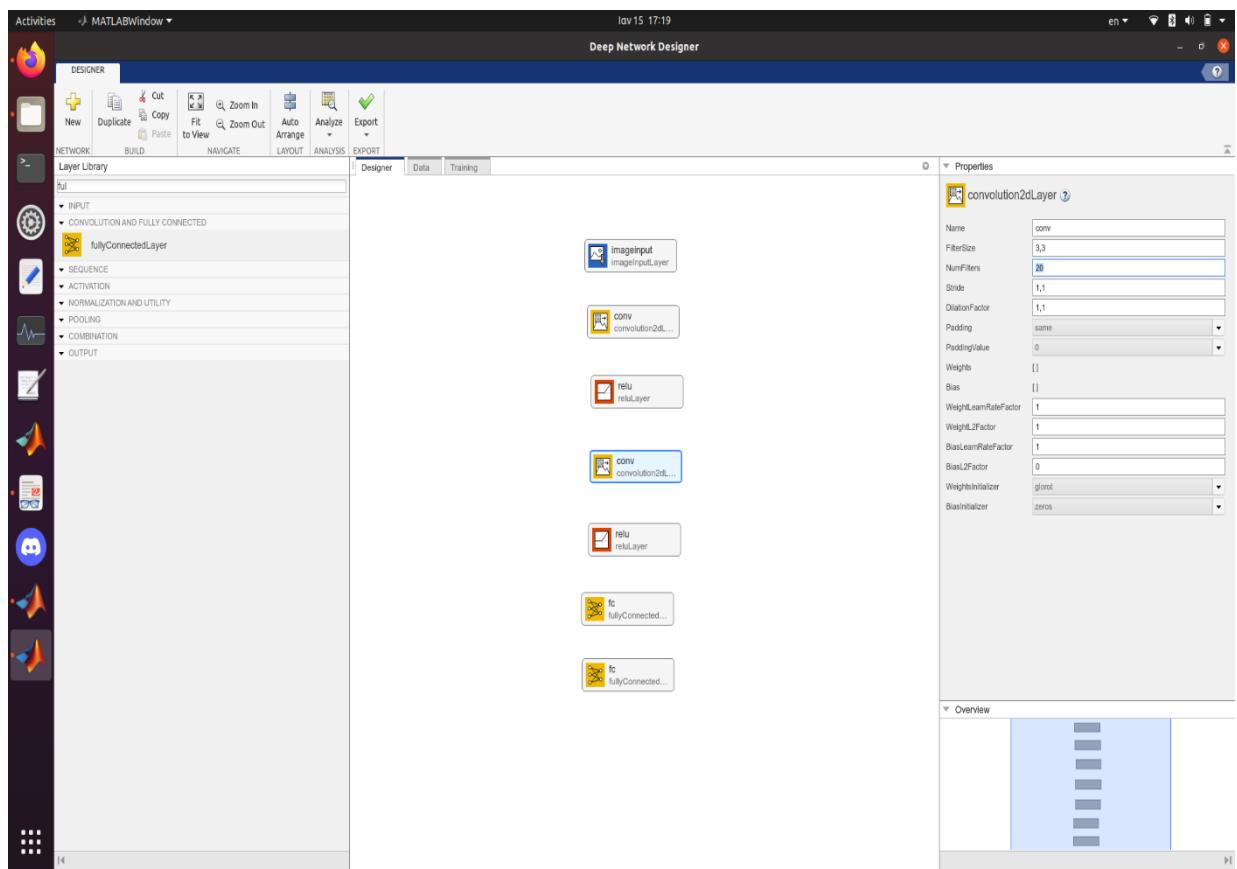
Μεταξύ του στρώματος εισόδου και του στρώματος εξόδου, θα μπορούσε να υπάρχει οποισδήποτε συνδυασμός στρωμάτων. Το στρώμα εξόδου: 10 έξοδοι, καθώς ασχολούμαστε με πρόβλημα αναγνώρισης μονοψήφιων αριθμών(0-9).

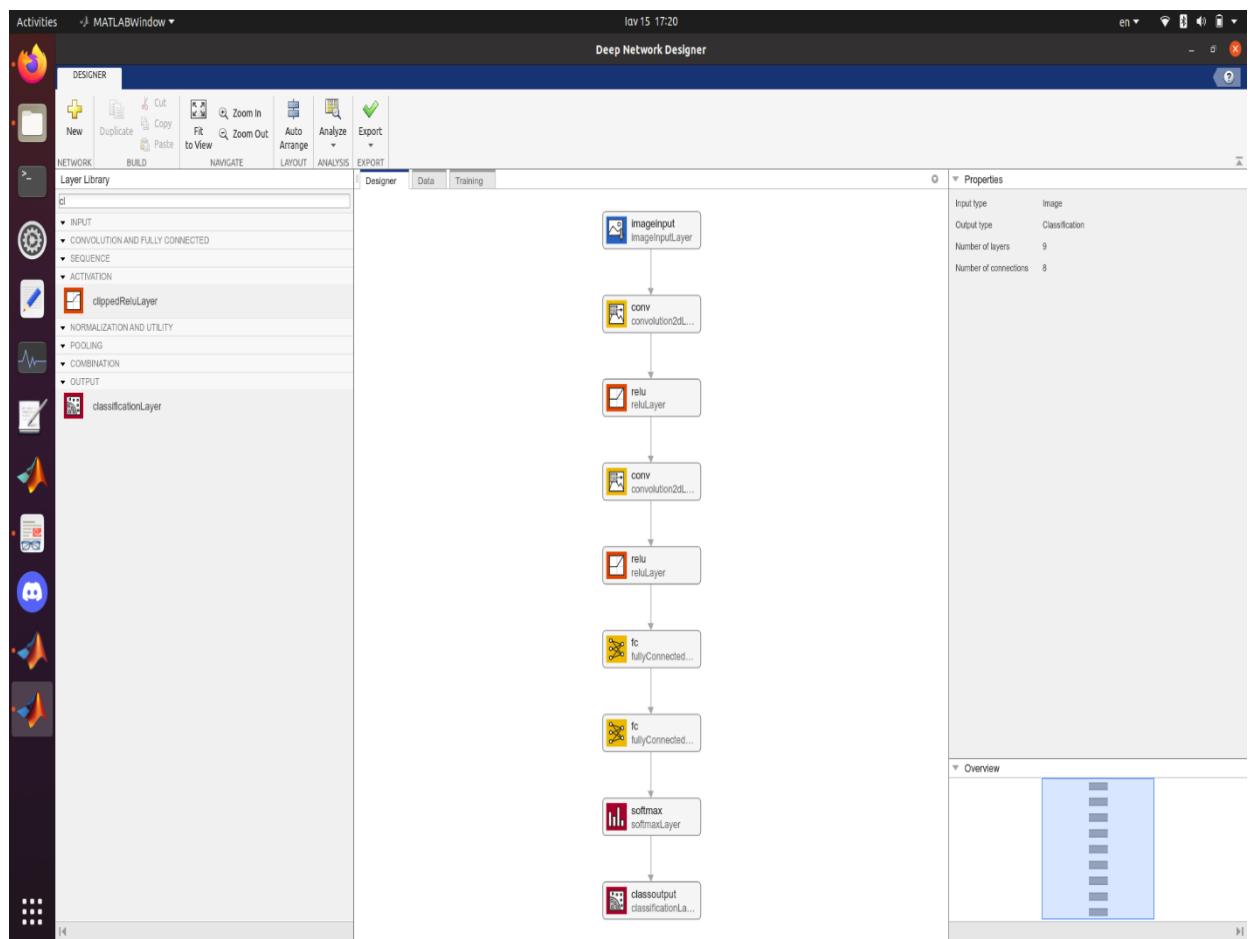
4. Σύνδεση των στρωμάτων μεταξύ τους
5. Κλικ στο Auto Arrange
6. Κλικ στο Analyze → Παρατηρούμε ότι ως αποτέλεσμα έχουμε 0 errors και 0 warnings. Άρα, η διαδικασία συνεχίζεται κανονικά.
7. Export
  - a. Network
  - b. Generate code
8. Επιλογή του Data tab → Import data:  
Browse→Home/Downloads/Matlab/R2022a/toolbox/nnet/nndemos/nndatasets/DigitDataset → Import
9. Επιλογή του Training tab → Train
10. Export

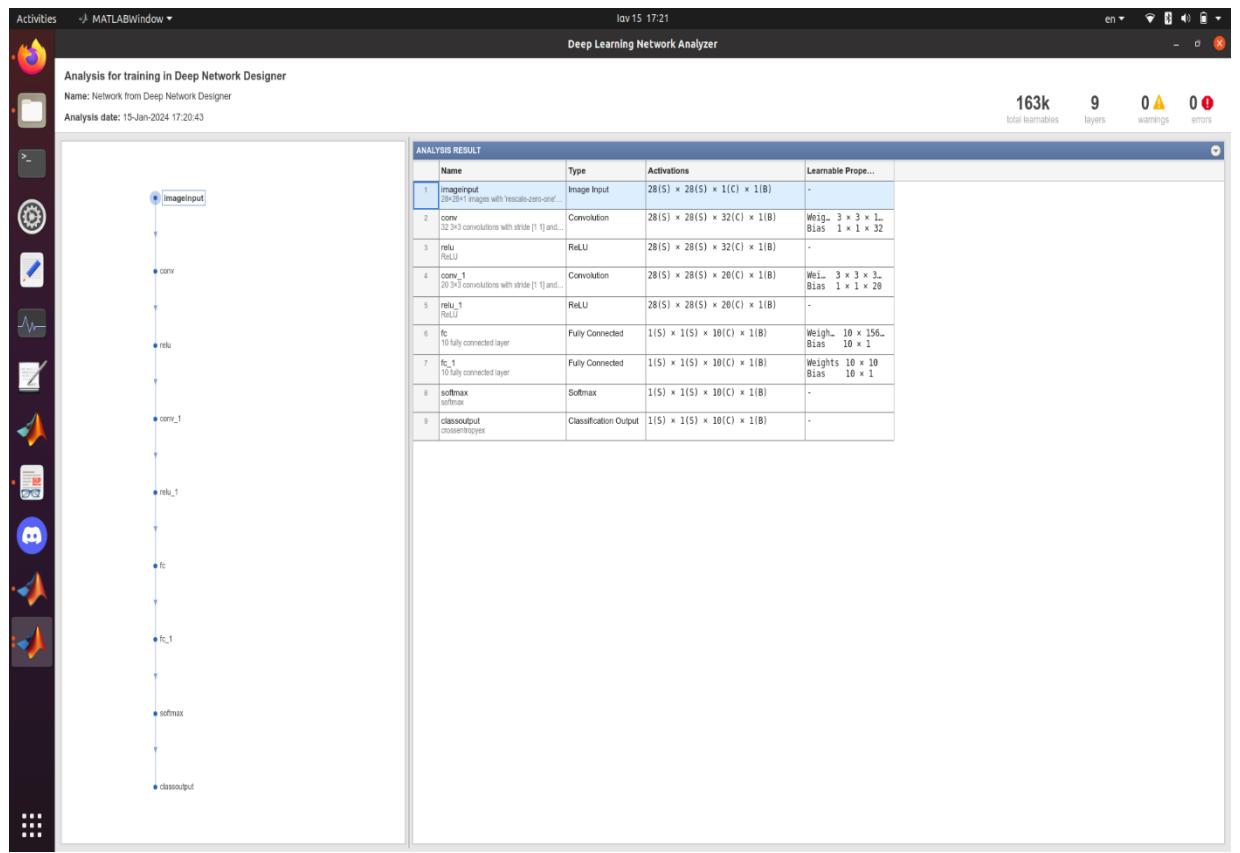
- a. Δημιουργία του trainedNetwork\_1
- b. Δημιουργία του trainInfoStruct\_1, που παρέχει στο χρήστη χρήσιμα στατιστικά που αφορούν τη διαδικασία της εκπαίδευσης.

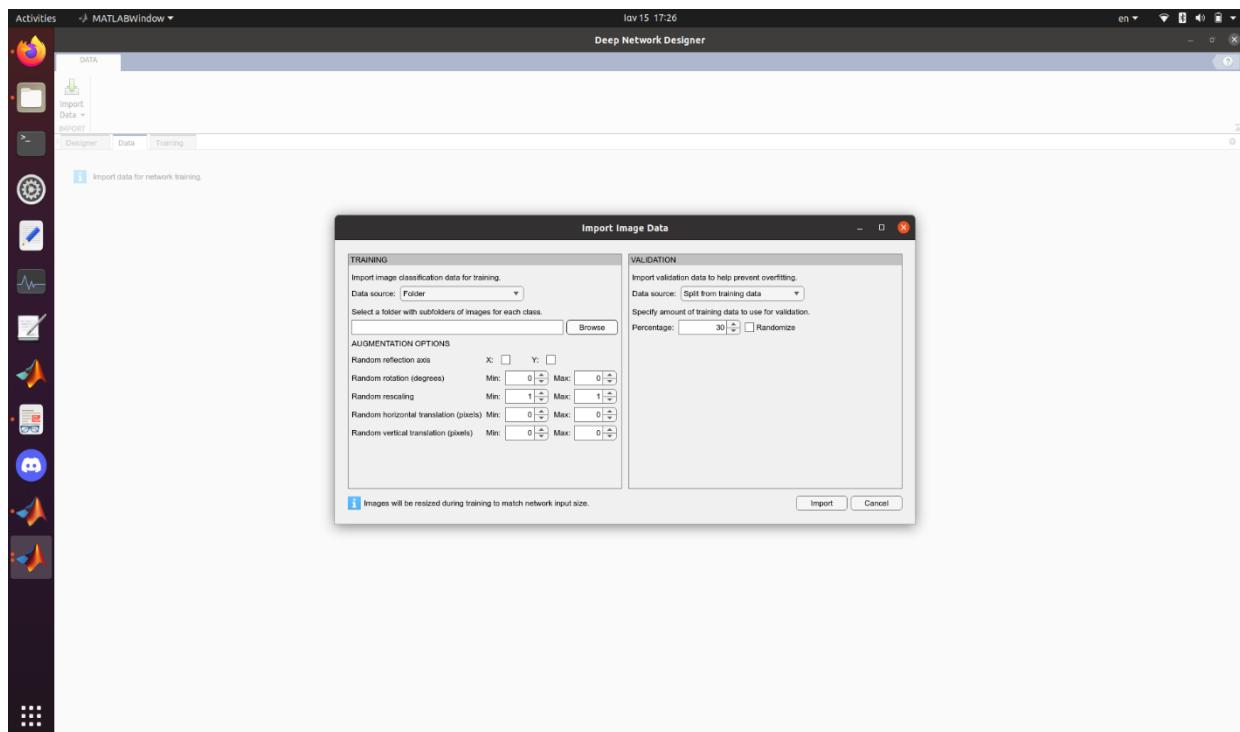
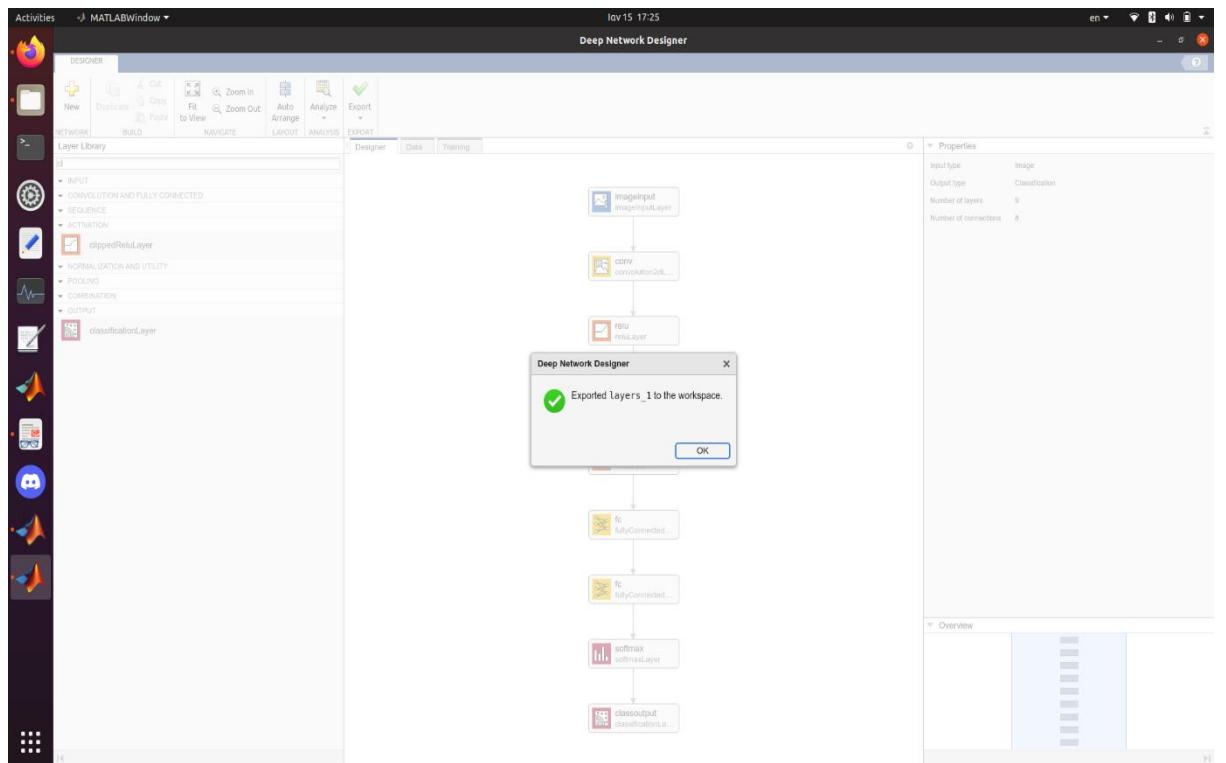
Ενώ, στο trainedNetwork\_1 μπορούμε να δούμε αναλυτικά πληροφορίες για το εκάστοτε στρώμα και τις τιμές των Weights, Bias και άλλων χρήσιμων παραμέτρων.

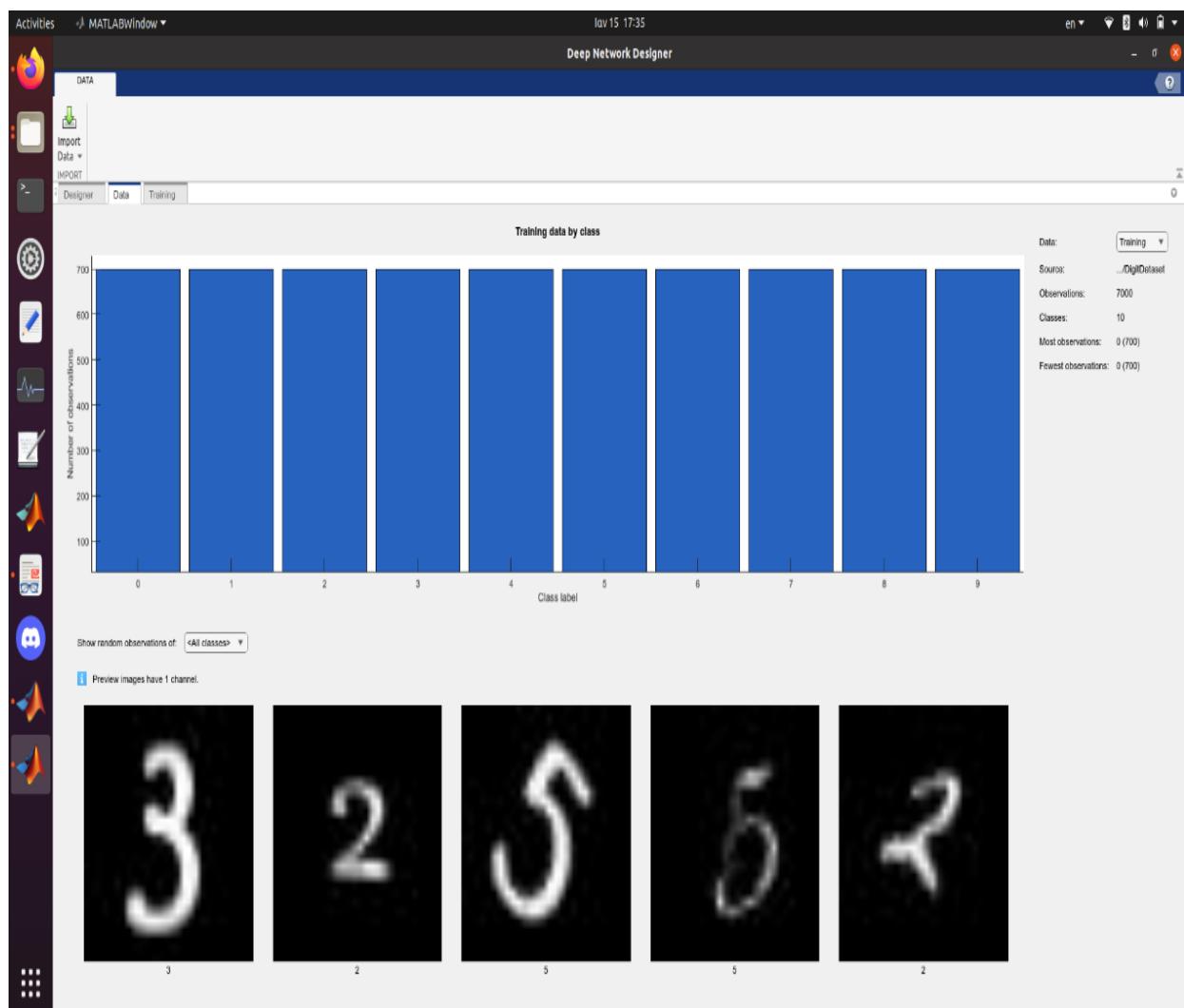
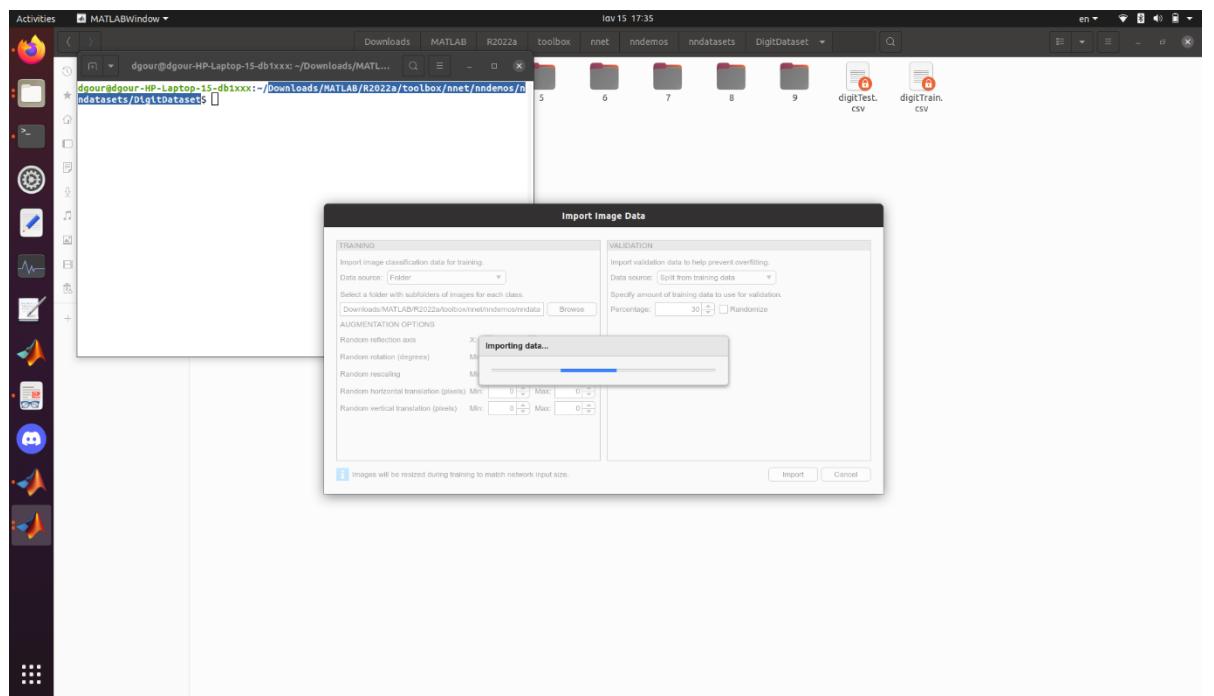


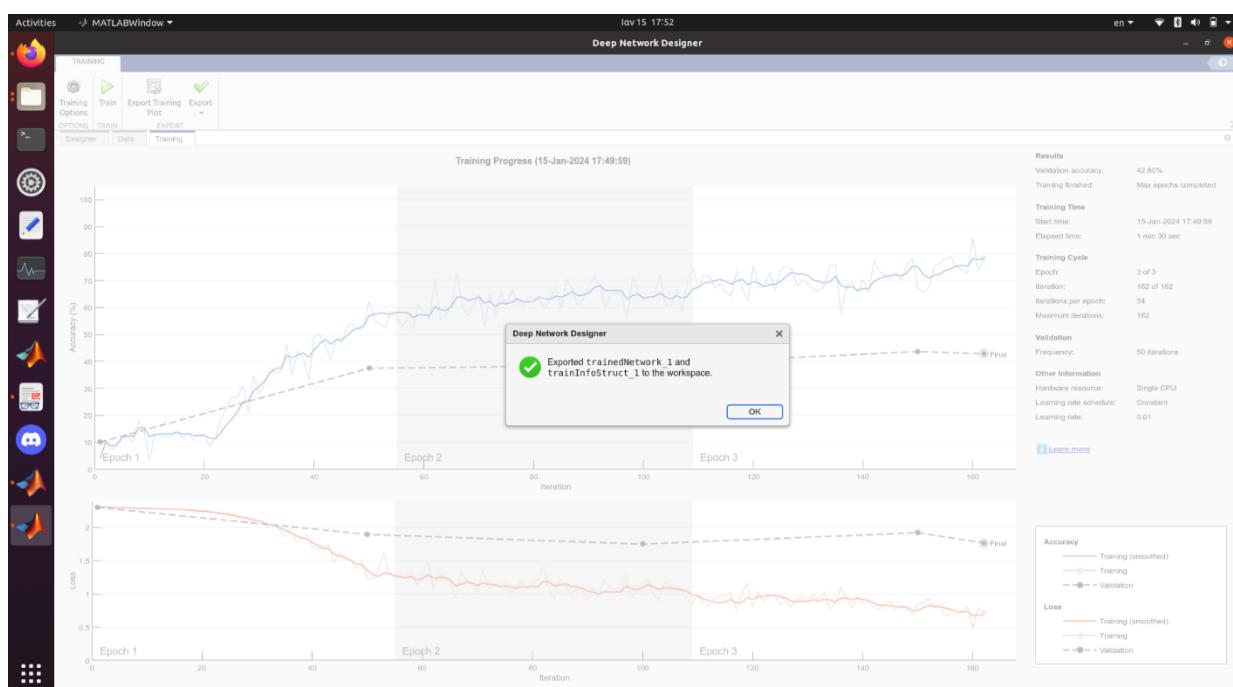












## Σχολιασμός αποτελεσμάτων:

Στο συγκεκριμένο παράδειγμα δημιουργούμε «από το 0» ένα νευρωνικό δίκτυο, επιλέγοντας κάθε στρώμα του, με στόχο να αναγνωρίσουμε χαρακτηριστικά από εικόνες. Όπως τονίσαμε και παραπάνω έγινε χρήση της εφαρμογής Deep Network Designer και θα γίνει εκπαίδευση, αλλά και έλεγχος της απόδοσης του δικτύου πάνω στο DigitDataset, που είναι προ εγκαταστημένο κατά την εγκατάσταση του Matlab, και περιέχει φωτογραφίες μονοψήφιων αριθμών(0-9). Σε κάθε ψηφίο αντιστοιχούν 700 εικόνες. Όσον αφορά τα στρώματα του δικτύου:

- **imageInput:** Το πρώτο στρώμα (imageInput) είναι υπεύθυνο για την εισαγωγή των εικόνων στο δίκτυο. Αυτό το στρώμα καθορίζει το μέγεθος των εικόνων εισόδου.
- **conv:** Το στρώμα συνέλιξης (conv) εφαρμόζει συνελίξεις στις εικόνες. Αυτό βοηθά στην εξαγωγή χαρακτηριστικών από τις εικόνες.
- **relu:** Το στρώμα ενεργοποίησης relu εφαρμόζει τη συνάρτηση ενεργοποίησης, η οποία προσθέτει μη-γραμμικότητα στο μοντέλο.
- **fc:** Το στρώμα πλήρως συνδεδεμένων νευρώνων (fc) εκτελεί πράξεις πλήρως συνδεδεμένου επιπέδου, συνδέοντας όλες τις εικόνες
- του προηγούμενου στρώματος με κάθε νευρώνα.
- **softmax:** Το στρώμα Softmax εφαρμόζει την συνάρτηση Softmax, μετατρέποντας τις εξόδους του προηγούμενου επιπέδου σε πιθανότητες.
- **classoutput:** Το στρώμα classoutput χρησιμοποιείται σε προβλήματα κατηγοριοποίησης. Εδώ γίνεται η τελική αναγνώριση της κλάσης.

Η συγκεκριμένη σειρά τοποθέτησης των στρωμάτων του νευρωνικού δικτύου χρησιμοποιείται σε προβλήματα εικόνας και κατηγοριοποίησης.

Σαν screenshot παρουσιάζεται το σύνολο των δεδομένων που αναμένεται να εκπαιδευτεί. Έπειτα, παρουσιάζεται η εκπαίδευση του νευρωνικού δικτύου σε κάθε εποχή. Αρχικά, η απόδοση του δικτύου είναι χαμηλή, ενώ τα σφάλματα πολλά, στη συνέχεια η κατάσταση διαφοροποιείται. Δηλαδή αυξάνεται αρκετά η απόδοση του δικτύου και μειώνεται συνεχώς ο αριθμός των λανθασμένων κατηγοριοποιήσεων. Είναι κατανοητό το γεγονός ότι αν επιλέγαμε περισσότερες εποχές εκπαίδευσης θα οδηγούμασταν σε καλύτερη απόδοση του δικτύου.

## Παράδειγμα 2

Βήματα:

1. openExample('nnet/TransferLearningWithDeepNetworkDesignerExample')
2. MerchData = unzip("MerchData.zip")
3. Άνοιγμα του App DeepNetworkDesigner
4. Άνοιγμα του προεκπαιδευμένου νευρωνικού δικτύου SqueezeNet
5. Import των MerchData, και αλλαγή των παραμέτρων
6. Άνοιγμα του Data tab
7. Προετοιμασία του δικτύου για εκπαίδευση:

Παρατηρούμε ότι τα MerchData αναγνωρίζουν 5 κλάσεις. Οπότε, πρέπει να επεξεργαστούμε το τελευταίο classification layer. Συγκεκριμένα, για το SqueezeNet το τελευταίο classification layer είναι το τελευταίο conv, όπου θα αλλάξουμε τις παραμέτρους του(NumFilters=5). Η συγκεκριμένη παράμετρος-ιδιότητα καθορίζει τον αριθμό των κλάσεων για προβλήματα ταξινόμησης.

Επίσης, αλλάζουμε και το τελευταίο layer του δικτύου, δηλαδή το classificatonLayer.

8. Analyze Network(για τσεκάρισμα λαθών στο δίκτυο)
9. Άνοιγμα του Training tab και αλλαγή ορισμένων Training Options.
10. Export
11. Classify new image

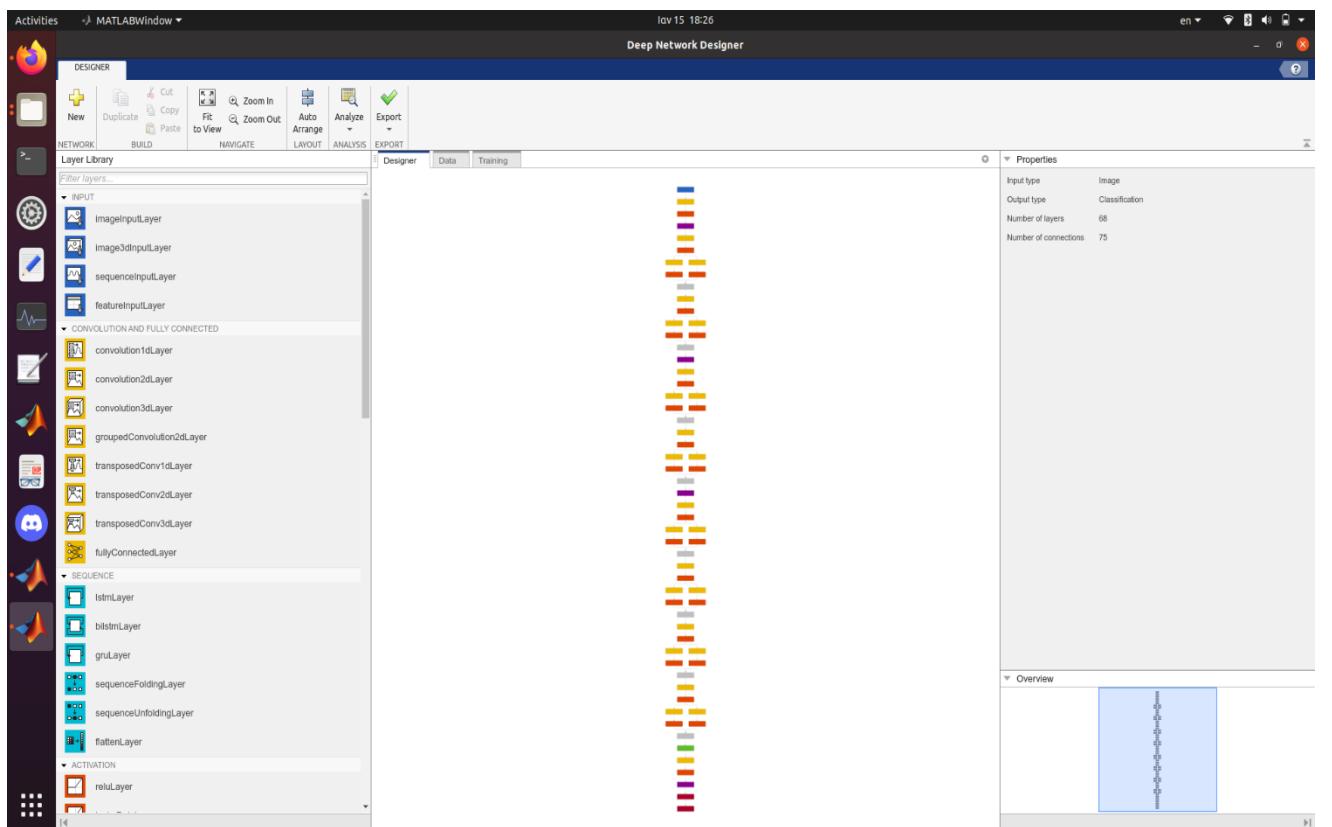
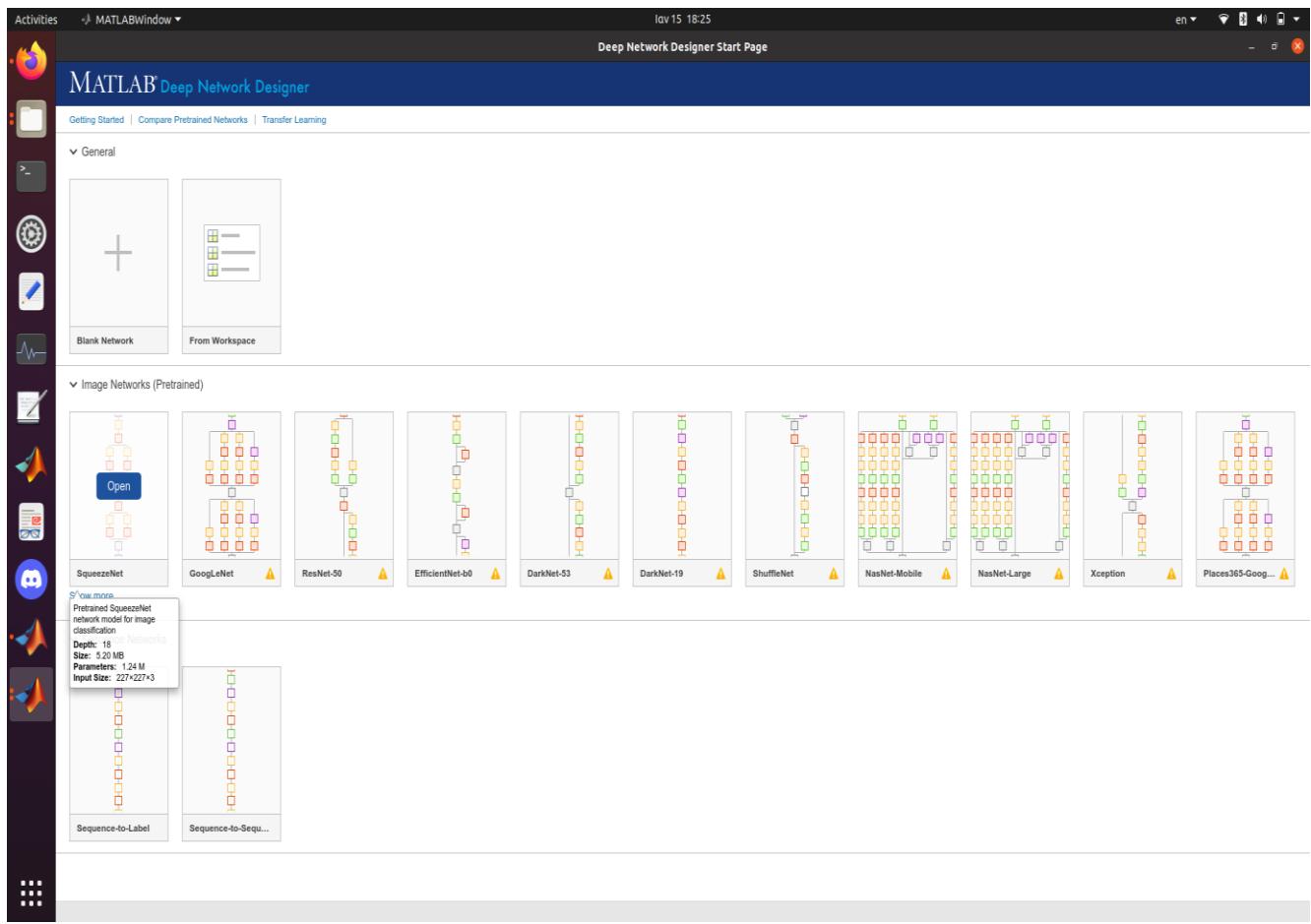
Αρχείο: b2\_code.m

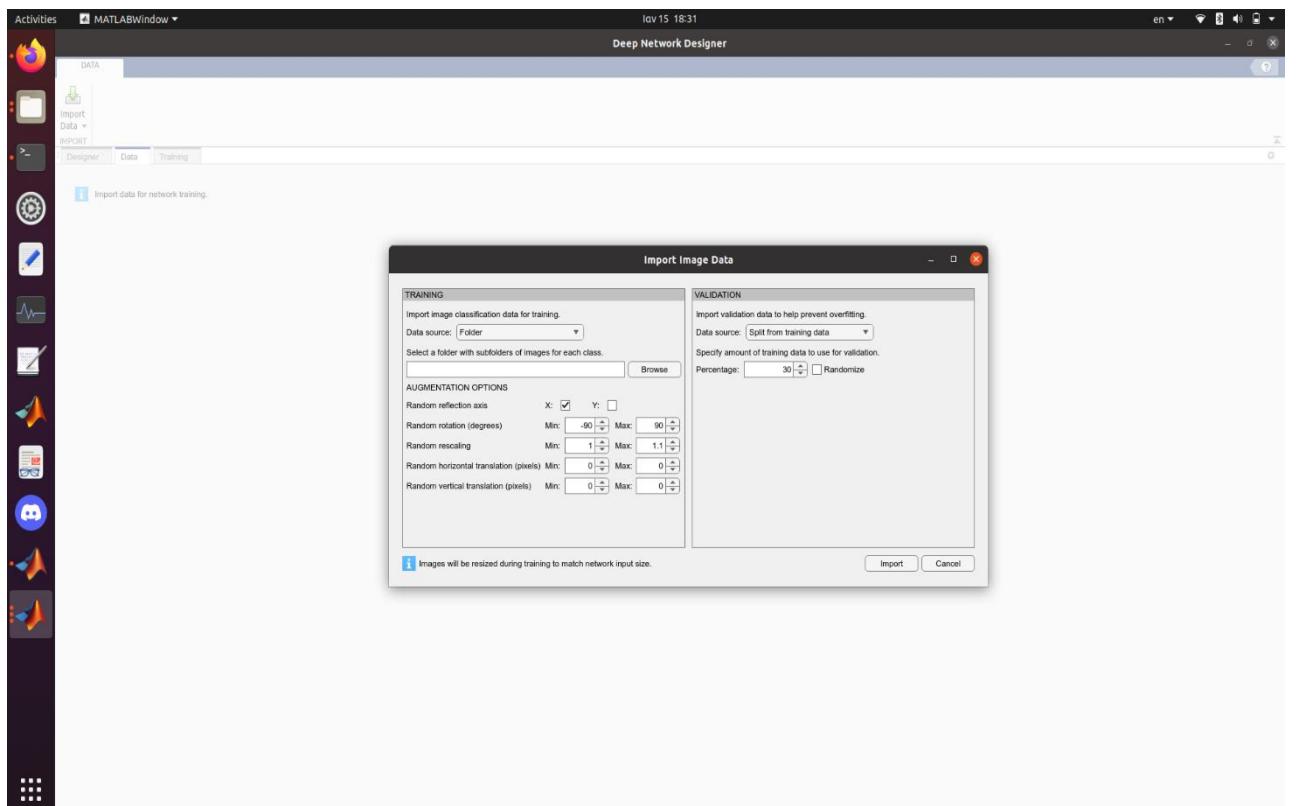
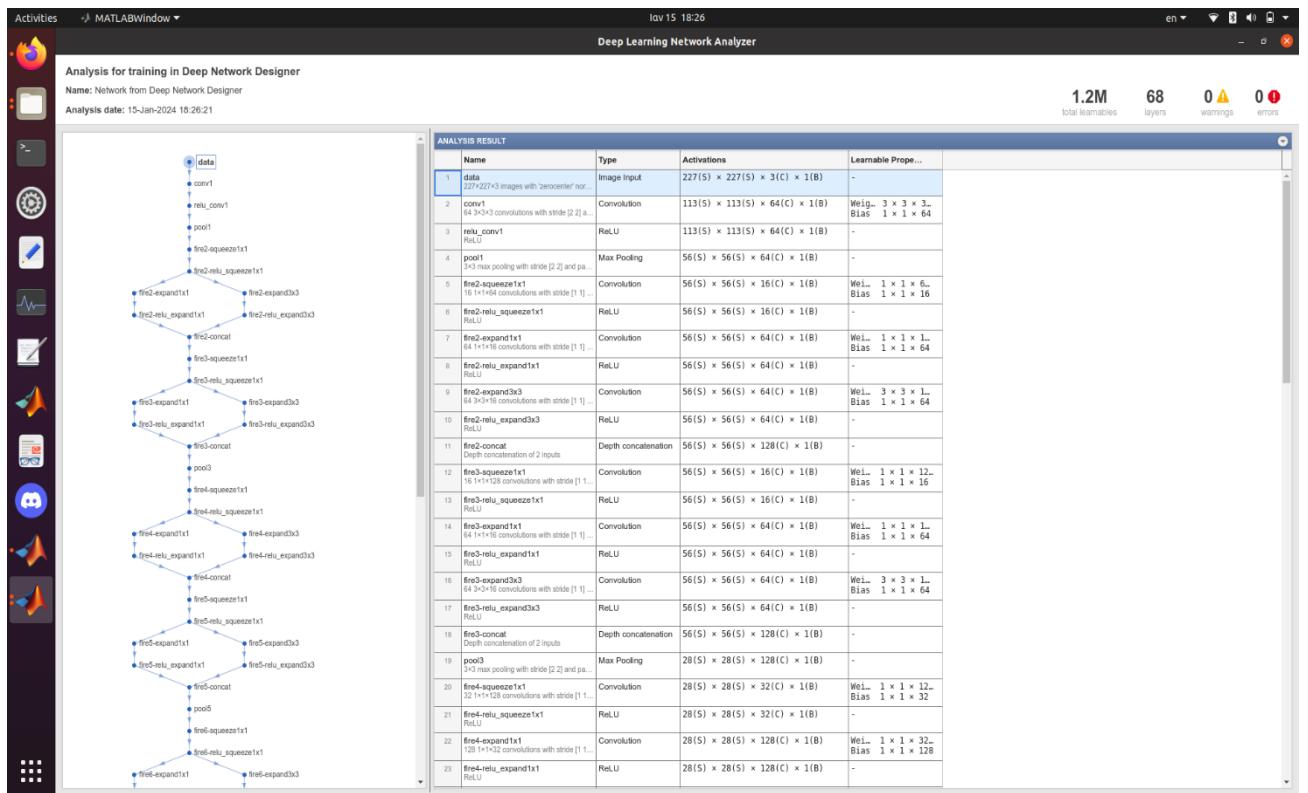
```
MerchData = unzip("MerchData.zip");

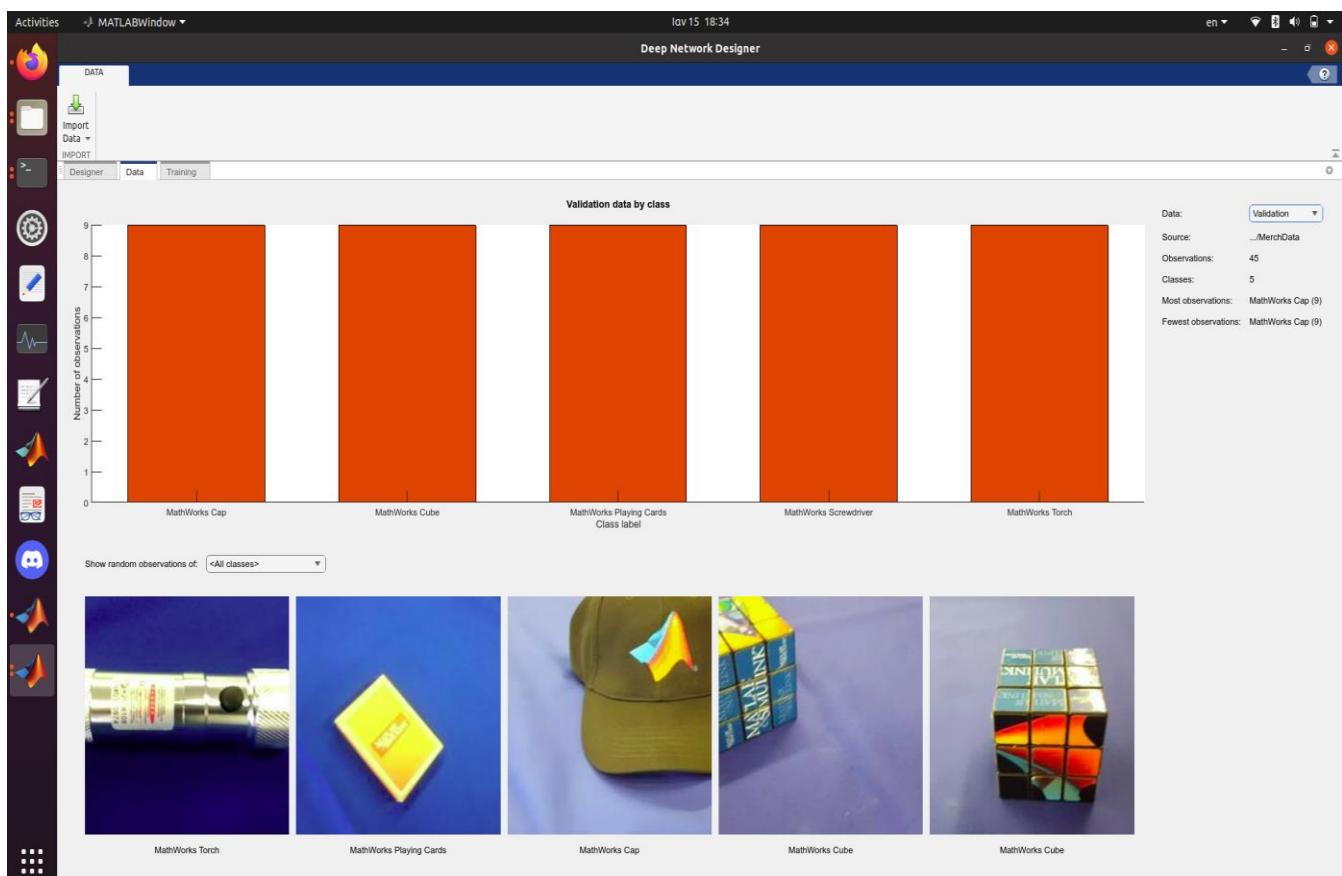
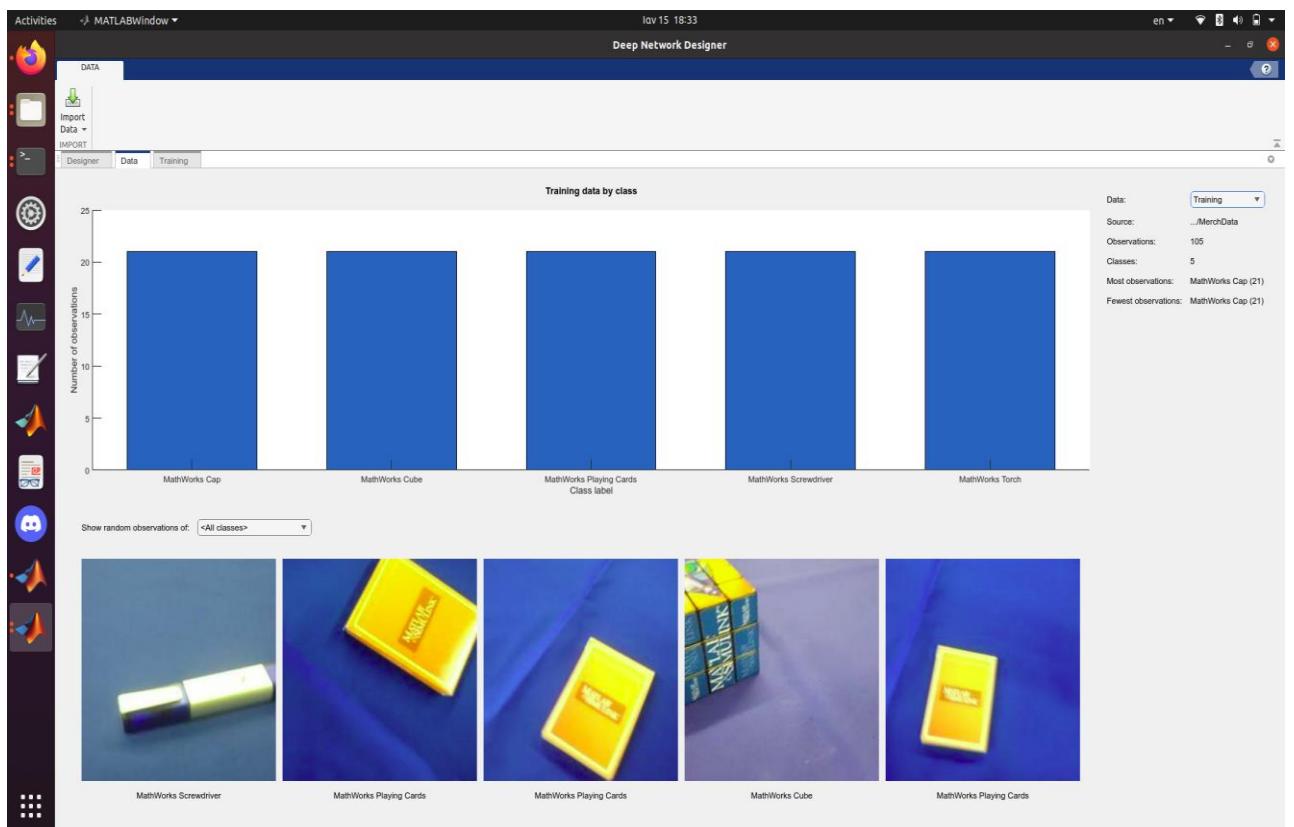
I = imread("MerchDataTest.jpg");
I = imresize(I, [227 227]);

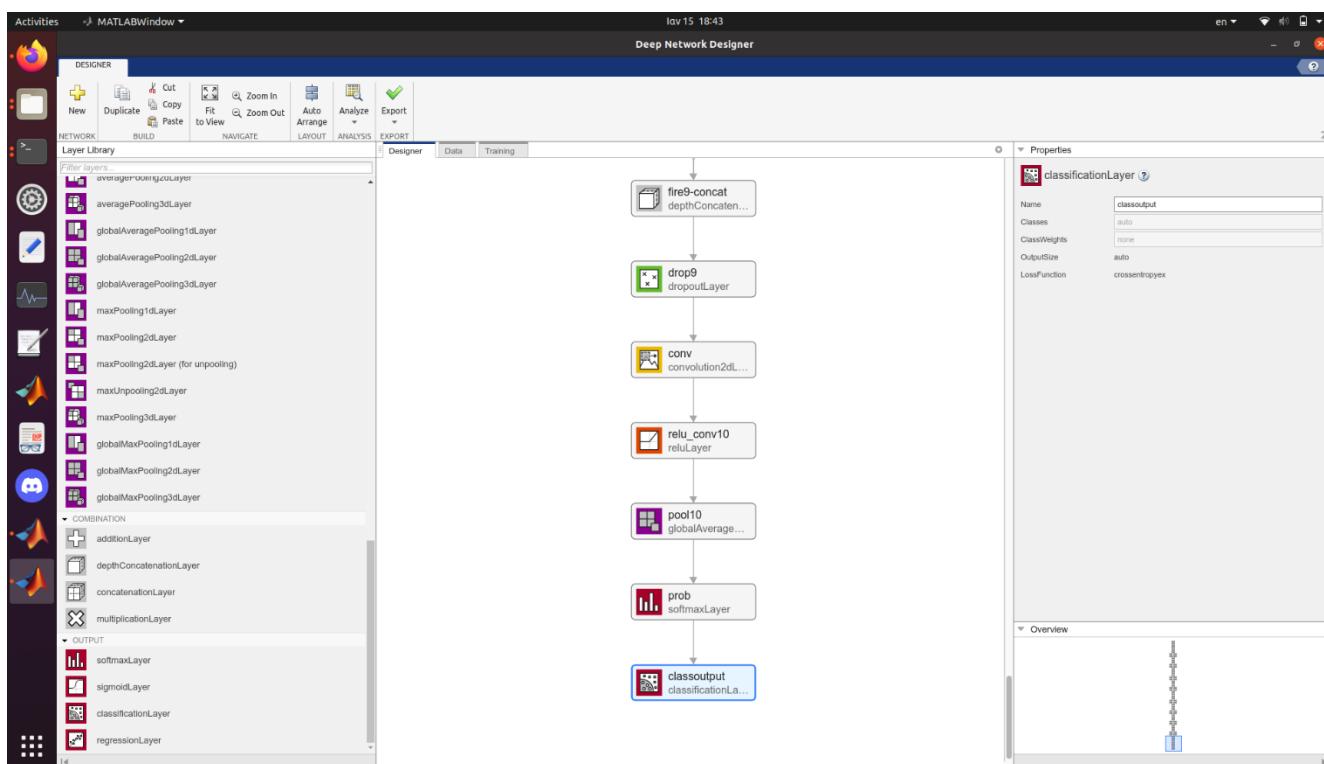
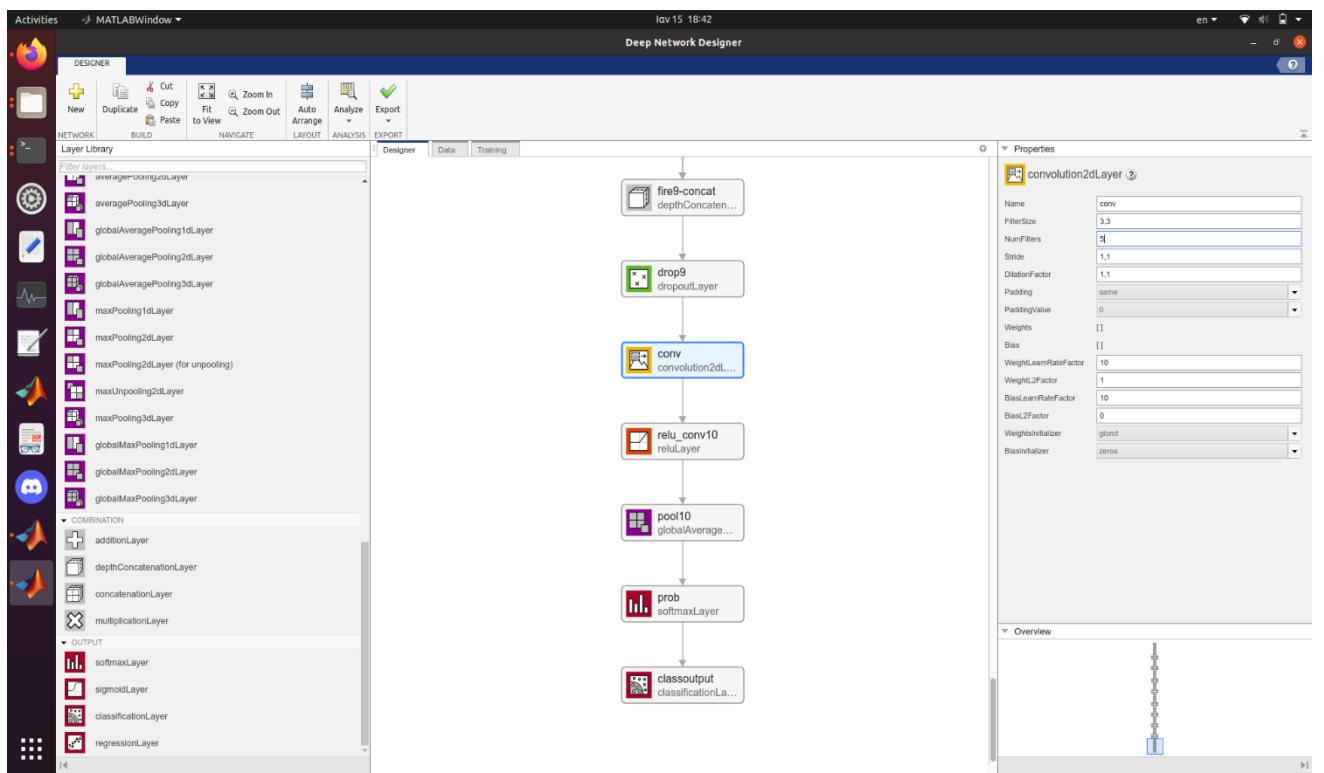
[YPred,probs] = classify(trainedNetwork_1,I);
imshow(I)
label = YPred;
title(string(label) + ", " + num2str(100*max(probs),3) + "%");
```

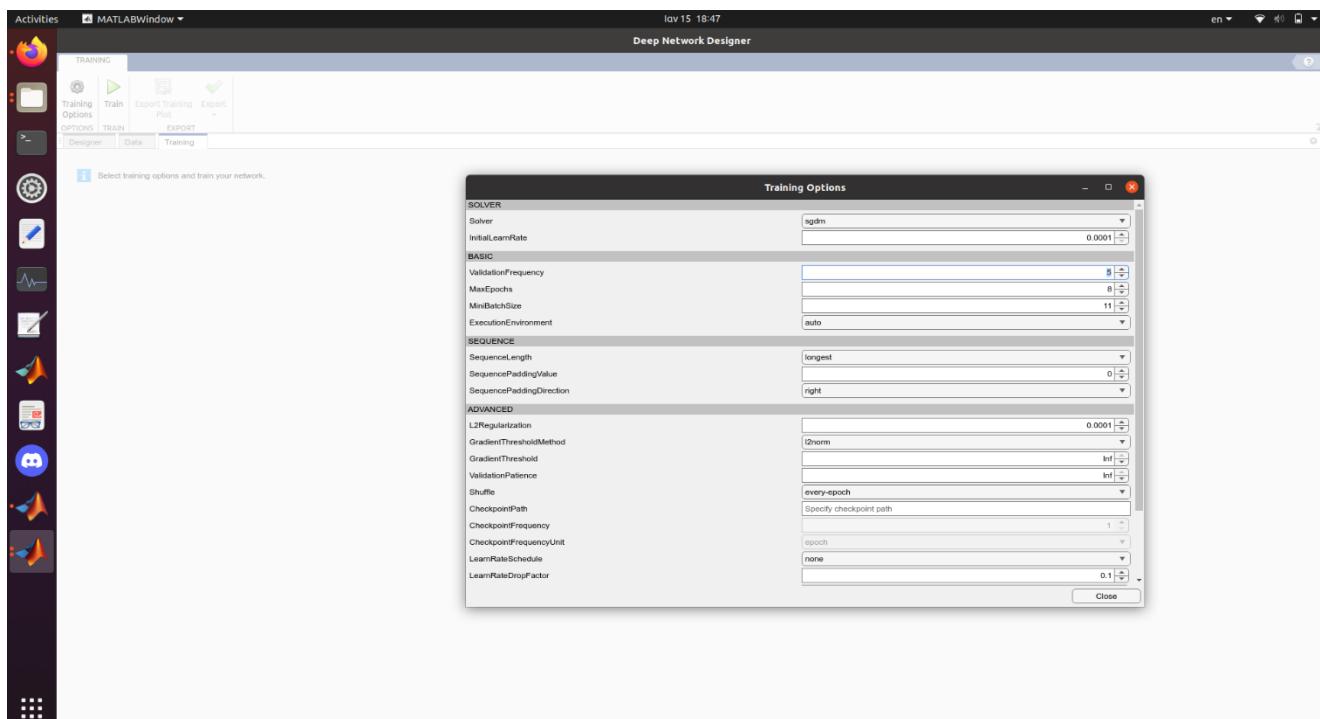
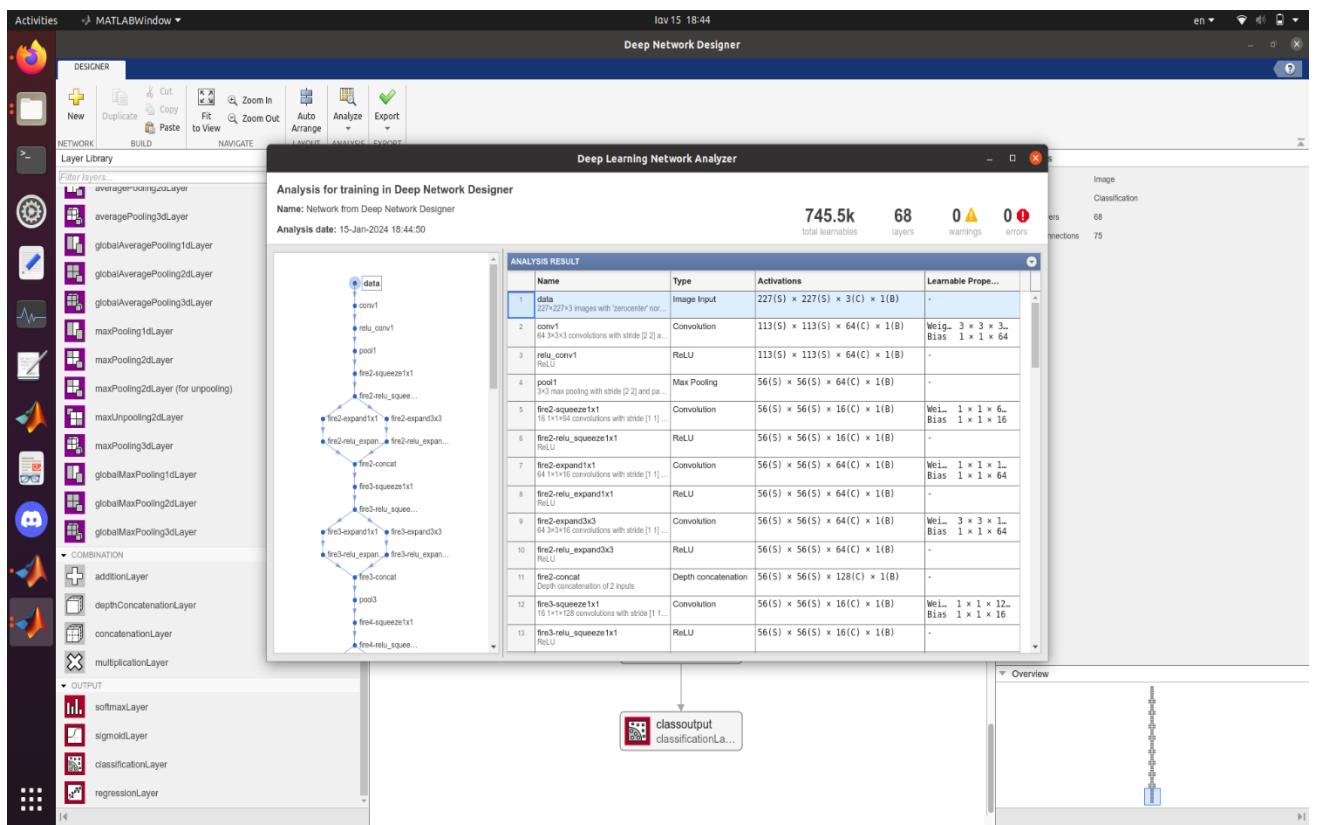
**Αποτέλεσμα:** Σωστή αναγνώριση(100%) της κλάσης στην οποία η εικόνα MerchDataTest.jpg, δηλαδή στην κλάση MathWorks Cube.

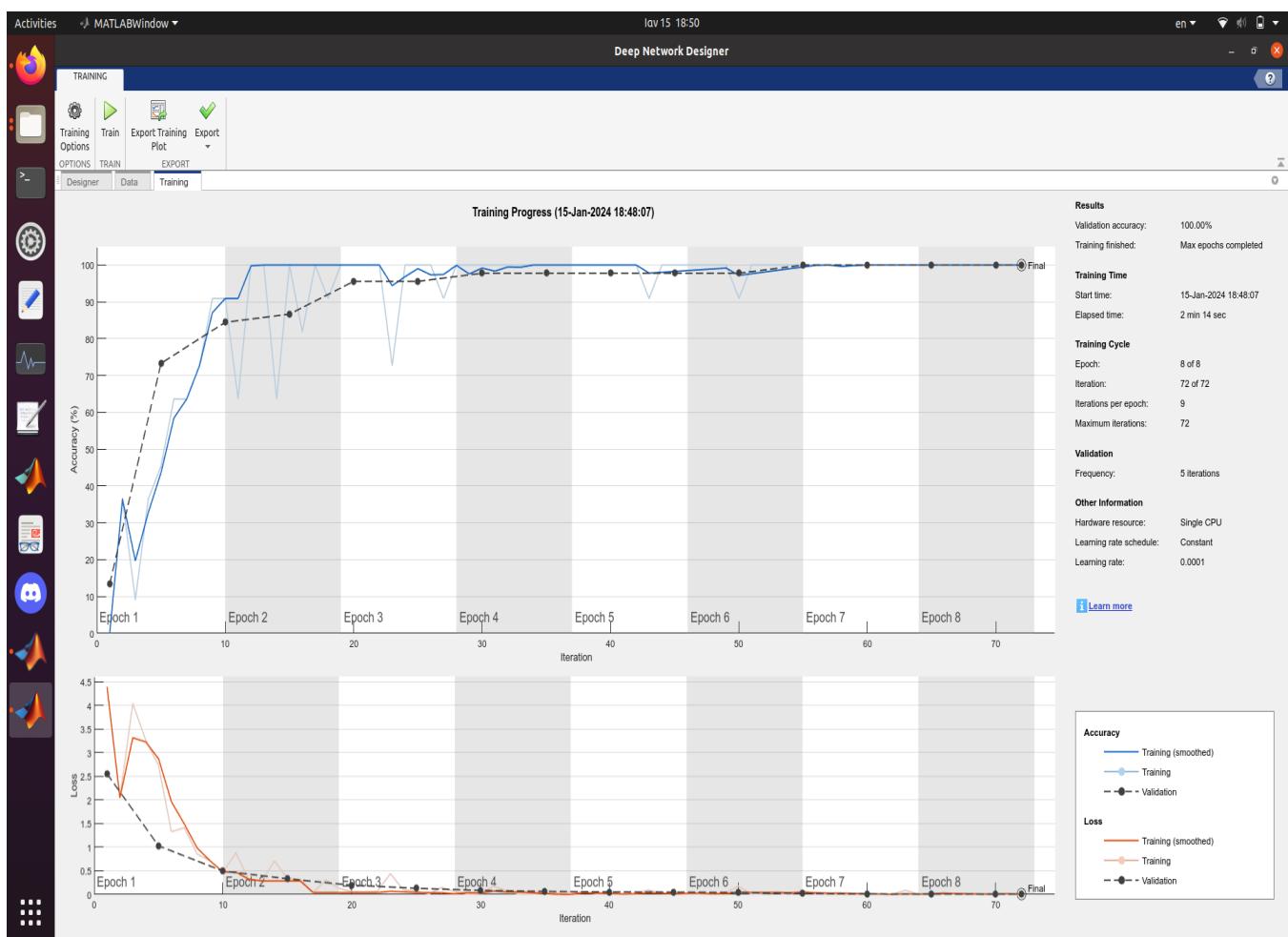


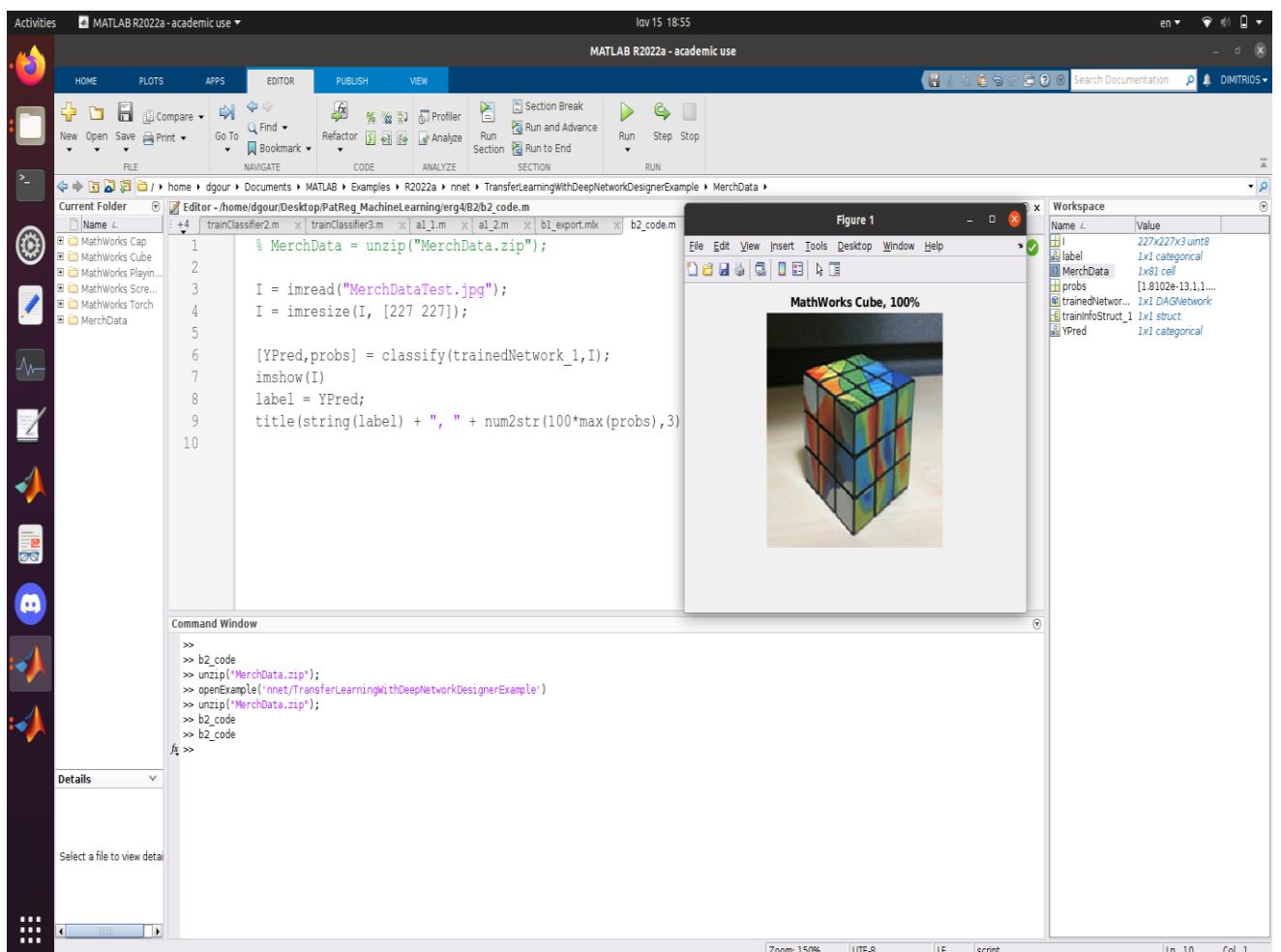
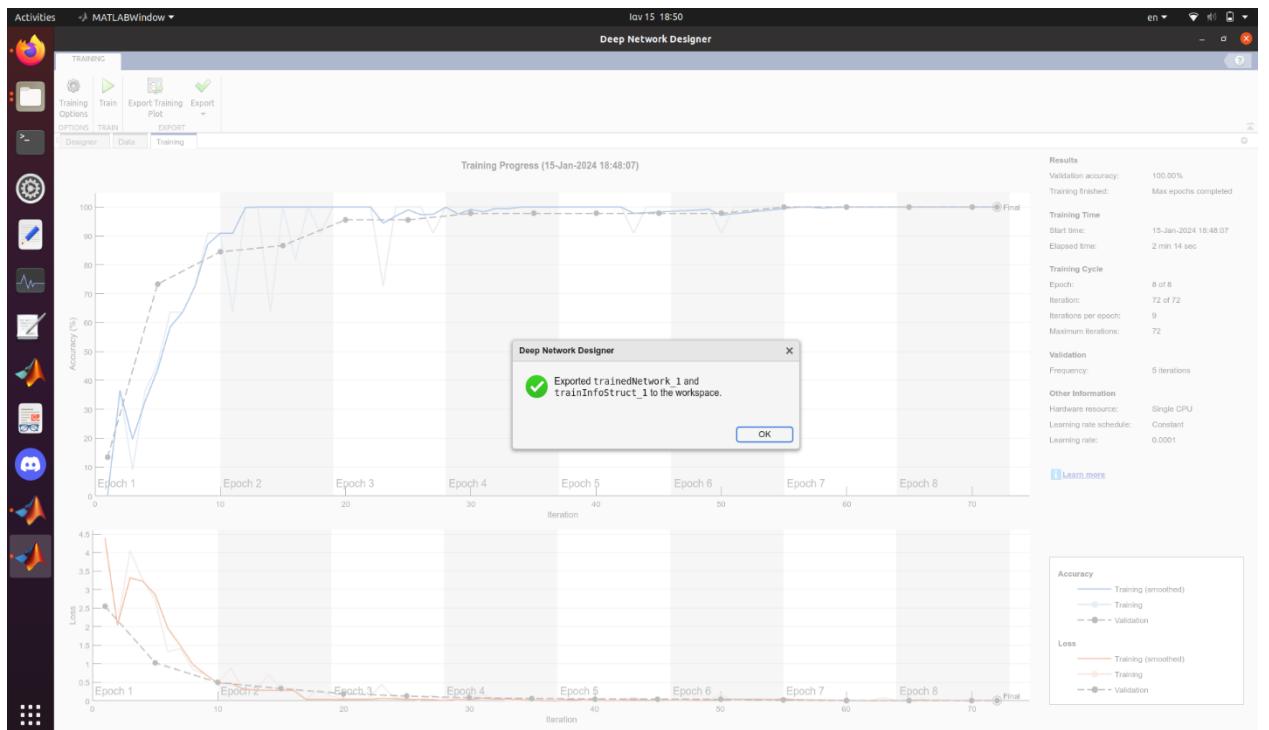












### **Σχολιασμός αποτελεσμάτων:**

Στο συγκεκριμένο παράδειγμα χρησιμοποιούμε το προεκπαιδευμένου νευρωνικό δίκτυο SqueezeNet, που αποτελεί ένα ελαφρύ νευρωνικό δίκτυο που ασχολείται με κατηγοριοποίηση εικόνων, και αλλάζουμε τα τελευταία στρώματα του.

Συγκεκριμένα, όπως παρουσιάζεται και στις παραπάνω εικόνες, αλλάζουμε το τελευταίο στρώμα του δικτύου, αλλά και το τελευταίο conv στρώμα. Οι κλάσεις που υπάρχουν σε αυτό το παράδειγμα είναι 5: Cap,Cube,Playing Cards,Screwdriver,Torch. Στο σύνολο δεδομένων υπάρχουν 105 παρατηρήσεις για εκπαίδευση, δηλαδή 21 από κάθε κλάση. Παράλληλα, για validation υπάρχουν 45 παρατηρήσεις.

Το δίκτυο εκπαιδεύεται, φτάνοντας 100% validation accuracy. Στο διάγραμμα παρατηρούμε την άνοδο της απόδοσης του δικτύου ανά εποχή, όπως και τη μείωση των λανθασμένων κατηγοριοποιήσεων. Τέλος, επιλέγουμε την εικόνα **MerchDataTest.jpg**, που ανήκει στην κλάση Cube. Με τη χρήση της συνάρτησης classify και ως όρισμα το νευρωνικό δίκτυο που δημιουργήσαμε (παραλλαγμένο SqueezeNet) παρατηρούμε ότι η εικόνα κατηγοριοποιήθηκε στην σωστή κλάση.