ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας

Εργαστηριακές Ασκήσεις - Μέρος Β - Θέμα Ι

Εαρινό Εξάμηνο 2024
Παράδοση: Σεπτέμβρης 2024

ΟΝΟΜ/ΜΟ:	**Ιωάννης Καζιξής**
ΕΤΟΣ:	**4ο**
ΑΡ. ΜΗΤΡΩΟΥ:	**1084515**
Email	**up1084515@ac.upatras.gr**

Θέμα Ι : Ταξινόμηση Εικόνων

Ερώτημα 1: Απεικόνιση Δείγματος από κάθε Κλάση του Mnist^2.

Όπως γράφω και μέσα στον κώδικα δεν κατάφερα να κατεβάσω το minst database από το http://yann.lecun.com/exdb/mnist/ καθώς έλεγε ότι δεν έχω την απαραίτητη άδεια, παρόλα αυτά βρήκα από αλλού το ίδιο αρχείο και το φορτώνω στην Matlab σαν ένα training και ένα test struct. Επισυνάπτω εδώ το σχετικό αρχείο…

mnist.mat

Μέρος Α:
Κατηγοριοποίηση Εικόνων με χρήση Συνελικτικών Νευρωνικών Δικτύων

Ερώτημα 2: Διαχωρισμός συνόλου σε Training και Validation Sets.

Η διαδικασία είναι σχετικά απλή, φτιάχνω ένα split index το οποίο το ορίζω να είναι το 80% του συνόλου = 60.000, από την αρχή έως αυτό έχω τα training data μου και μετά αυτού έως το τέλος τα validation. Χωρίζω καταλλήλως τα structs μου και δημιουργώ δύο καινούργιά structs με τα images και labels μεγέθους 48.000 και 12.000 για το training και το validation set αντίστοιχα.

```
%2.1.3
clear
load("mnist.mat");

%trainning data χωρισμένο σε training set και validation όπου το validation
%set δείχνει το λάθος στο νευρονικό δίκτυο
%μετά έχω το test set δεδομένα που δεν έχει ξαναδεί το δίκτυο για να
%παρατηρήσω την πραγματική του απόδοση σε γενικές περιπτώσεις δεδομένων
%εισόδου
%επομένος χωρίζω το training data σε 80-20% training-validation και έχω και το test set μου


splitIndex = 60000*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});    %48.000 εικόνες για το
trainingset
validationSet = struct('images',{training.images(:,:,splitIndex:end)},'labels',{training.labels(splitIndex:end)});  %12.000 εικόνες για
validationset

%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)

reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,48000);  % Μέγεθος: [28, 28, 1, 48000]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []);  % Μέγεθος: [28, 28, 1, 12000]

%τεστ μετατροπής των δεδομένων μου για την είσοδο

imagestart= size(trainingSet.images)
imagereshaped= size(reshapeTrainingSetimages)
imagelabels= size(trainingSet.labels)

x=2;

figure
    image (rescale(training.images(:,:,x),0,255));
    colormap("gray");
    colorbar
    axis square equal
    title(training.labels(x))

figure
    image (rescale(trainingSet.images(:,:,x),0,255));
    colormap("gray");
    colorbar
    axis square equal
    title(trainingSet.labels(x))
```

Ο κώδικας εμπεριέχει κάποια test τα οποία έχουνε γίνει commented για να ελέγξω ότι δεν έχω ανακατέψει ή φθείρει με οποιονδήποτε άλλο τρόπο τα δεδομένα μου μετά τον διαχωρισμό τους σε δύο νέα σύνολα και την μετατροπή τους για να είναι συμβατά ως είσοδο στο νευρωνικό μου δίκτυο. Στην ουσία δήλωσα ότι οι εικόνες είναι βάθους =1 δηλαδή είναι greyscale και όχι rgb.

Ερώτημα 3 : Κατασκευή και Εκπαίδευση του Νευρωνικού Δικτύου.

Σχόλια: Για την εκπαίδευση του δικτύου θα χρησιμοποιήσω την συνάρτηση της matlab trainnet παραμετροποιόντας την κατάλληλα όσον αφορά τα layers του Νευρωνικού δικτύου και τα options της συνάρτησης για την εκπαίδευση, από κάτω εξηγώ τις επιλογές μου αναλυτικά.

- Layers: Σύμφωνα με την εκφώνηση πάντα, έχουμε το image input layer το οποίο είναι η είσοδος του νευρωνικού δικτύου και είναι μεγέθους ανάλογο των εικόνων.

```
inputSize =[28 28 1];
```

Επίσης έχουμε το convolution layer με 6 φίλτρα ή kernels μεγέθους [5χ5], αυτό παράγει ένα feature map βάθους 6, (δηλαδή κάθε kernel παράγει ένα feature map και μετά παραθέτονται όλα σε ξεχωριστό βάθος το καθένα).

```
convolution2dLayer([5,5],6)
```

Ακολουθείτε από ένα ReluLayer όπου η λειτουργεία του είναι να αφήνει ανέγγιχτα τα στοιχεία τις εισόδου με τιμές >=0 ενώ τα στοιχεία που είναι <0 τα κανονικοποιεί όλα σε =0.
Το pooling layer στην συνέχεια μειώνει το μέγεθος της εισόδου κατά το ήμισυ εφόσον είναι [2χ2]. Όσον αφορά το pooling layer επειδή δεν διευκρινίζεται τα χαρατκηριστικά του αποφάσησα να είναι averaging δηλαδή να παίρνει τον μέσον όρον από τα τέσσερα στοιχεία στην γειτονιά που δημιουργεί και επίσης οι γειτονιές μεταξύ τους να μην επικαλύπτονται άρα έθεσα το stride και αυτό [2χ2].

```
reluLayer
averagePooling2dLayer([2,2],"Stride",[2,2])
```

Με την ίδια λογική επαναλαμβάνεται ακόμη μια φορά η ακολουθία των επιπέδων αυτών (conv, relu ,pooling)…
Η έξοδος είναι το fully connected layer το οποίο έχει 10 νευρώνες όσες και οι κλάσεις μάς που συνδέονται όλοι με αυτούς του προηγούμενου επιπέδου, εδώ εισόδου, (δέκα ψηφία από το 0-9). Αυτό παράγει ένα σκορ σε κάθε νευρώνα του για την πιθανότητα της δοθείσας εικόνας να ανήκει σε κάθε έναν από αυτούς.

```
classes = numel(unique(training.labels));  %10 κλάσεις
```

Τέλος έχω το softmax layer το οποίο μετατρέπει σε πιθανότητες τα αποτελέσματα (σκορ) του fully connected layer , στην ουσία κανονικοποιεί όλα τα σκορ σε τιμές από το 0-1.

```
layers = [
  imageInputLayer(inputSize)
  convolution2dLayer([5,5],6)
  reluLayer
  averagePooling2dLayer([2,2],"Stride",[2,2])
  convolution2dLayer([5,5],12)
  reluLayer
  averagePooling2dLayer([2,2],"Stride",[2,2])
  fullyConnectedLayer(classes)
  softmaxLayer
 ];
```

- Options:a) Ορίζω την συνάρτηση ανανέωσης των παραμέτρων του δικτύου να είναι η Stochastic Gradient Descent.
  b) Ορίζω το MiniBatchSize = 20 , δηλαδή θα ανανεώνει τα βάρη κάθε 20 εικόνες. Οι εποχές θα είναι 4 επομένος το δίκτυο μας θα περάσει (trainingSet/ MiniBatchSize)*epochs => (48.000/20)*4 = 9600 iterations.
  c) Για την παρακολούθηση της εξέλιξης της διαδικασίας εκπαίδευσης έχουμε την crossentropy Loss μετρική (όπως μας είχατε υποδείξει στην διάλεξη) ο οποία δείχνει πόσο μακριά είναι κάθε φορά η μαντεψιά του δικτύου από την πραγματικότητα, αυτήν ακριβώς την μετρική χρησιμοποιεί ο βελτιστοποιητής Stochastic Gradient Descent για να ανανεώσει τα βάρη του fully connected layer μας. Μπορείτε να δείτε ότι έχω σημειώσει και άλλες μετρικές αλλά αυτές δεν τις πήρα υπόψην διότι δεν ζητούνται και διότι δεν τις γνωρίζω τόσο καλά. Συμπεριλήφθηκαν λόγω δικών μου πειραματισμών και περιέργειας. Έχω επίσης ενεργοποιήση την λειτουργεία verbose η οποία στο command window περιοδικά τυπώνει τις τιμές των εν λόγω συναρτήσεων καθώς και άλλα στοιχεία και το training progress για να βλέπω live την εξέλιξη των γραφημάτων.
  d) Ορίζω τα Validation Data μου , το σύνολο δεδομένων δηλαδή που μας αποκαλύπτει πόσο καλά γενικεύει το δίκτυο μας και το Validation Frequency ίδιο με το MiniBatchSize.

e) Ορίζω το περιβάλλον υλοποίησης ώς cpu διότι μετά από δοκιμές συμπέρανα η εκτέλεση στον cpu είναι σχεδόν 2X γρηγορότερη από την gpu καθώς ούτε στην cpu ούτε στην gpu κατάφερα να χρησιμοποιήσω την παράλληλη αρχιτεκτονική τους και έτρεχε μόνο σε ένα core (cpu ή gpu) και τα μεμονωμένα cores του cpu ήταν ταχύτερα.
f) Τέλος ορίζω στο δίκτυο μετά το πέρας κάθε εποχής να αποθηκεύετε για να συγκρίνω έπιτα χρησιμοποιώντας το TestSet ποιας εποχής δίκτυο είναι το αποδοτικότερο.
g) Το output network το τελικό δίκτυο δηλαδή θα είναι αυτό με την καλύτερη (μικρότερη) τιμή της μετρικής Validation Loss , για αυτό θα κάνω λόγω παρακάτω.

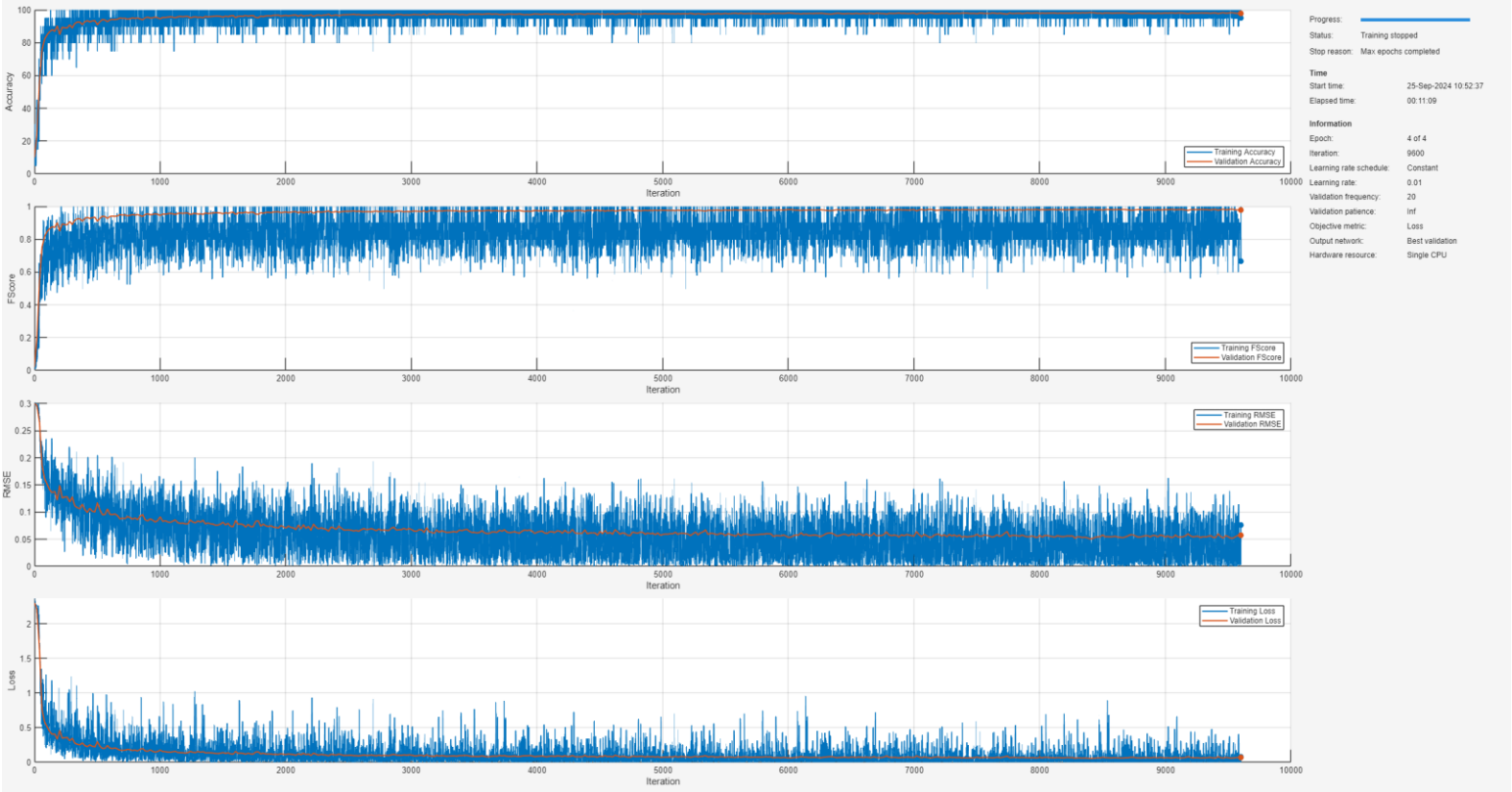
Εδώ φαντάζομαι είναι απαραίτητο να ορίσετε εσείς το δικό σας path εάν επιθυμείτε να δοκιμάσετε τον κώδικα.

```
epoch =4;   %εποχές
b =20;  %minibatchsize

options = trainingOptions("sgdm", ...
   MiniBatchSize=b, ...
   MaxEpochs=epoch, ...
   Metrics=["accuracy","fscore","rmse"],  ...
   Verbose=true, ...
   VerboseFrequency=20,...
   ValidationData={reshapeValidationSetimages, categorical(validationSet.labels)}, ...
   ValidationFrequency=b, ...
   OutputNetwork="best-validation", ...
   ExecutionEnvironment="cpu",...
   Plots="training-progress", ...
   CheckpointPath="E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και
Ανάλυση Εικόνας\ΜέροςΒ", ...
   CheckPointFrequency=1 ...
   );
```

Συνάρτηση trainnet:

```
[diktyo,info] = trainnet(reshapeTrainingSetimages,categorical(trainingSet.labels),layers,'crossentropy',options);
```

Εικόνα διεκπεραίωσης εκπαίδευσης.



Παρατηρήσεις: Εδώ βλέπουμε την πορεία εκπαίδευσης του δικτύου για 4 εποχές, φαίνεται ήδη από την πρώτη πως και το Validation Loss έχει κατασταλάξει γύρω στο 0.1 και μειώνεται αργά ,στην ουσία μετά από την πρώτη εποχή έχει επέλθει η σύγκλιση καθώς τρεις εποχές μετά έχει μειωθεί μόνο κατά ~0.03. Ενώ από την άλλην το Training Loss έχει κατέβει μεν αλλά παρουσιάζει τεράστια διασπορά, δεν μπορώ να πω ότι συγκλίνει κάπου. Για αυτόν τον λόγω πιστεύω είναι προτιμότερη η Μετρική Validation Loss για την ανάδειξη της απόδοσης του δικτύου. Εξάλλου δεν μας ενδιαφέρει το δίκτυο να βελτιστοποιηθεί για τα training data δηλαδή να αρχίσει να κάνει overfitting (ενώνοντας τα σημεία στο γράφημα λόγου χάρη) αλλά ζητάμε από το δίκτυο να γενικεύσει και να βρει την γενικότερη κατανομή των δεδομένων γενικού σκοπού που θέλουμε να ξεχωρίζει σε κλάσεις το δίκτυο μας και αυτήν την δυνατότητα είναι που ελέγχει το Validation Loss.

Μπορείτε να δείτε την επίδοση του δικτύου στο τέλος κάθε εποχής αναλυτικότερα σε αυτόν εδώ τον πίνακα.

Iteration	Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
2400	1	0.10432	95	0.10325	96.992
4800	2	0.11767	95	0.082364	97.525
7200	3	0.10034	95	0.076058	97.767
9600	4	0.079969	95	0.072258	97.807

Τα δεδομένα για τον πίνακα αντλούνται από αυτόν εδώ τον κώδικα μετά την περάτωση της εκπαίδευσης του δικτύου.

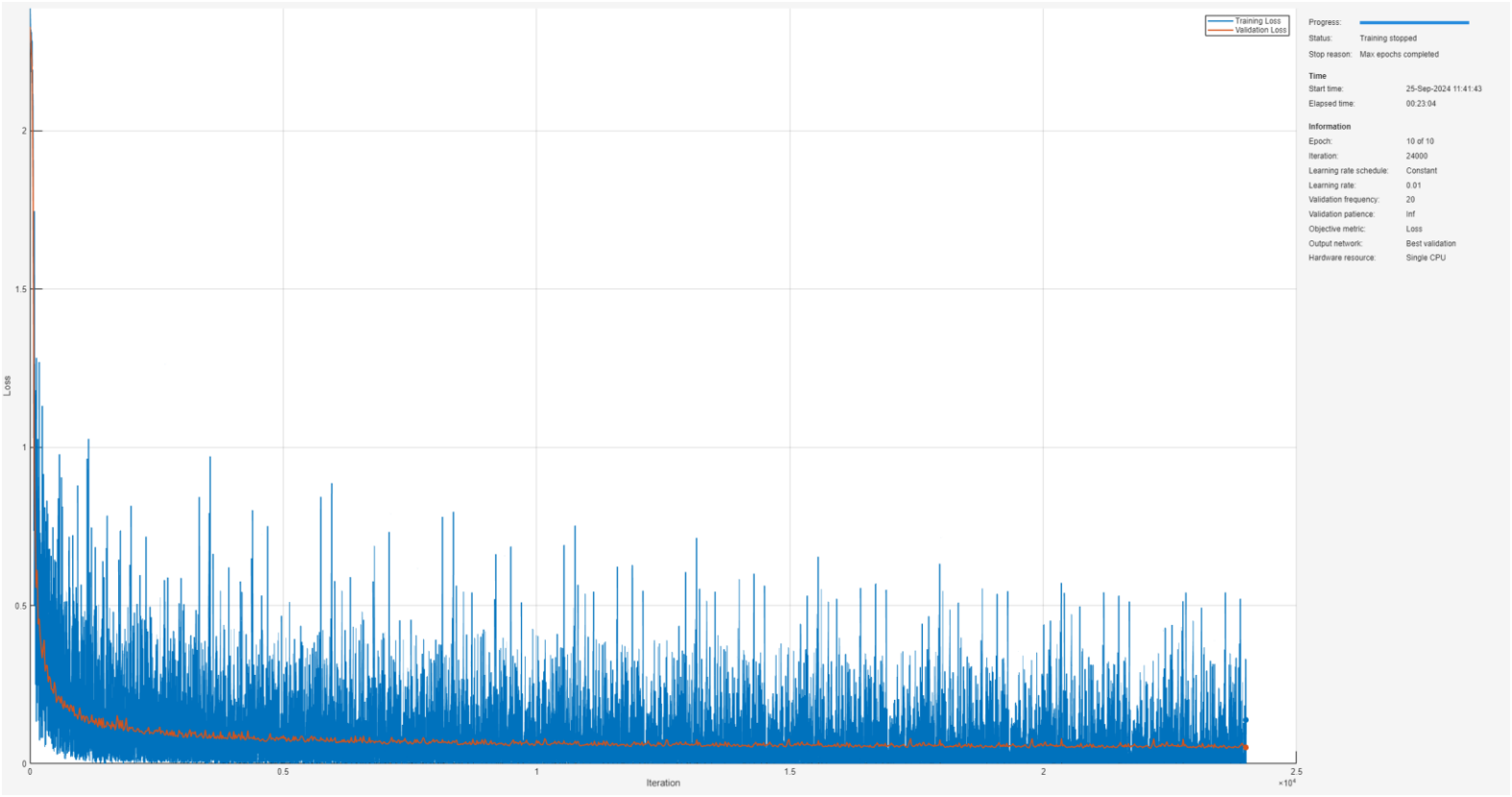
```
%δείχνω τις τιμές των μετρικών στο τέλος κάθε εποχής
for i=1:4
   blabla=["metrics from epoch",i];
   disp(blabla)
   TrainingHistory=info.TrainingHistory(i*epochsize,:)
   ValidationHistory=info.ValidationHistory(i*epochsize/20 +1,:)
end
```

```
%info.TrainingHistory
%info.ValidationHistory
```

Από όλα τα παραπάνω θεωρώ είναι σκόπιμο το τελικό δίκτυο να είναι βελτιστοποιημένο με βάση το Validation Loss. Άρα και για αυτό έβαλα στα options το OutputNetwork="best-validation" και για το συγκεκριμένο πείραμα αυτό είναι το Δίκτυο του iteration 8040 (δεν είναι σταθερά το 8040 , διότι στις επαναλήψεις του πειράματος υπάρχει shuffling των δεδομένων, στον προκειμένη είναι το 8040).

```
BestNetworkIterration = info.OutputNetworkIteration
```

Εδώ παραθέτω άλλο ένα πείραμα πάνω στο ίδιο δίκτυο για 10 εποχές χωρίς όμως τις παραπάνω extra μετρικές. Δεν το επανέλαβα πολλές φορές και εν τέλη δεν το προτίμησα διότι έπαιρνε απαγορευτικά πολύ χρόνο χωρίς την εκμετάλλευση της παραλληλίας όπως προανέφερα.
Νομίζω πάντως ότι φαίνεται πιο ξεκάθαρα η διαφορά στην ποιότητα του Trainning VS Validation…



Ερώτημα 4 : Έλεγχος του Νευρωνικού Δικτύου.

Χρησιμοποιώντας την συνάρτηση testnet μπορούμε να ελέγξουμε την απόδοση του δικτύου όσον αφορά την ποιότητα της γενίκευσης σε νέα άγνωστα για αυτό δεδομένα. Παίρνουμε ώς όρισμα το OutputNetwork και τα TestSet data, με το ίδιο minibatchsize και ελέγχουμε την Ακρίβεια, την τετραγωνική ρίζα του Μέσου Τετραγωνικού Σφάλματος ,το fscore και τέλος το crossentropy Loss.
Για να δούμε πως το BestNetwork (όσον αφορά Validation crossentropy Loss) ανταπεξέρχεται έναντι των iterations από τις άλλες εποχές φρόντισα να δοκιμάσω χρησιμοποιόντας την testnet και δίκτυα από το πέρας κάθε μιας.

Δίκτυα	Iteration	Accuracy	RMSE	Fscore	CrossEntropy
Best	8660	98.5600	0.0468	0.9855	0.0449
Epoch 4	9600	98.1900	0.0524	0.9817	0.0571
Epoch 3	7200	98.0700	0.0534	0.9805	0.0600
Epoch 2	4800	98.0800	0.0546	0.9807	0.0638
Epoch 1	2400	97.4200	0.0628	0.9740	0.0835

Φαίνεται ξεκάθαρα και από τις άλλες μετρικές ότι πράγματι σε όλους τους τομείς είναι καλήτερο από τα άλλα iterations με εξαιρετική ακρίβεια κατηγοριοποιεί κάθε αριθμό στην αντίστοιχη κλάση ψηφίο του.

Εάν θέλετε να δοκιμάσετε αυτό το κομμάτι του κώδικα θα πρέπει να αλλάξετε τα absolute paths , με συγχωρείται αλλά δεν κατάφερε να το γενικεύσω για να τρέχει αμέσως σε άλλους υπολογιστές.

```
%2.1.4
reshapeTestSetImages = reshape(test.images, 28, 28, 1, []); %το testSet μου Μέγεθους: [28, 28, 1, 10000]
diktyoepoch4 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ\net_checkpoint__9600__2024_09_25__11_03_46.mat");
diktyoepoch3 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ\net_checkpoint__7200__2024_09_25__11_01_01.mat");
diktyoepoch2 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ\net_checkpoint__4800__2024_09_25__10_58_16.mat");
diktyoepoch1 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ\net_checkpoint__2400__2024_09_25__10_55_37.mat");
```

Δεν κατάφερα να υλοποιήσω το Confusion Matrix

Παραθέτω εδώ τον κώδικα των προσπάθειων μου, (το διάνυσμα των predicted labels ήτανε ασύμβατης μορφής με την είσοδο του ConfusionChart και δεν κατάφερα να το φέρω στην σώστη).

```
%%
% classes2 = numel(unique(training.labels));  %10 κλάσεις
%
%
% predictedLabels = minibatchpredict(diktyo, reshapeTestSetImages,MiniBatchSize=b);
% predictedLabelsDecoded = onehotdecode(predictedLabels,classes2,2,"double");
%
% whos predictedLabelsDecoded
% reshapedPredictedLabelsDecoded = reshape(predictedLabelsDecoded,[],1);
% whos reshapedPredictedLabelsDecoded
%
% trueLabels =test.labels;
%
%
% % Display the confusion matrix as a chart
% confusionchart(trueLabels, PredictedLabelsDecoded);
```

Ερώτημα 5 : Επανάληψη χρησιμοποιώντας το 5%-10%-50% των Δεδομένων Εκπαίδευσης

Ο κώδικας χρειάστηκε να αλλάξει αντικαθιστώντας με μεταβλητές τις καρφωτές τιμές που είχα βάλει για τον ορισμό των μεγεθών.

Το 2 ερώτημα άλλαξε κυρίως.

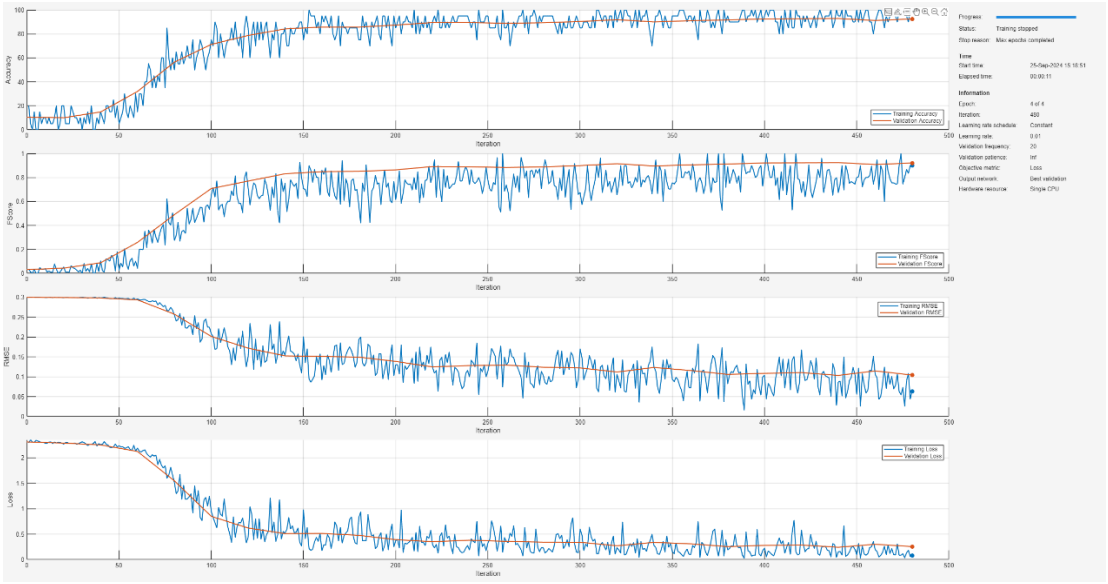
```
%2.1.5
%Τα παρακάτω βήματα είναι τα βήματα 2-4 χρησιμοποιόντας το 5% του συνόλου

%2.1.2
clear
load("mnist.mat");

percent = 5/100;
wholeSize = 60000;
newSize = wholeSize*percent
splitIndex = newSize*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});
validationSet =
struct('images',{training.images(:,:,splitIndex:newSize)},'labels',{training.labels(splitIndex:newSize)});

%Μεγέθη του Training και Validation Set
T = size(trainingSet.labels)
V = size(validationSet.labels)
%%
%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)
reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,[]); % Μέγεθος: [28, 28, 1, T(1,1)]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []); % Μέγεθος: [28, 28, 1, V(1,1)]

%Επικύρωση ακεραιότητας
size(reshapeTrainingSetimages)
size(reshapeValidationSetimages)
```

Και στο 3ο ερώτημα

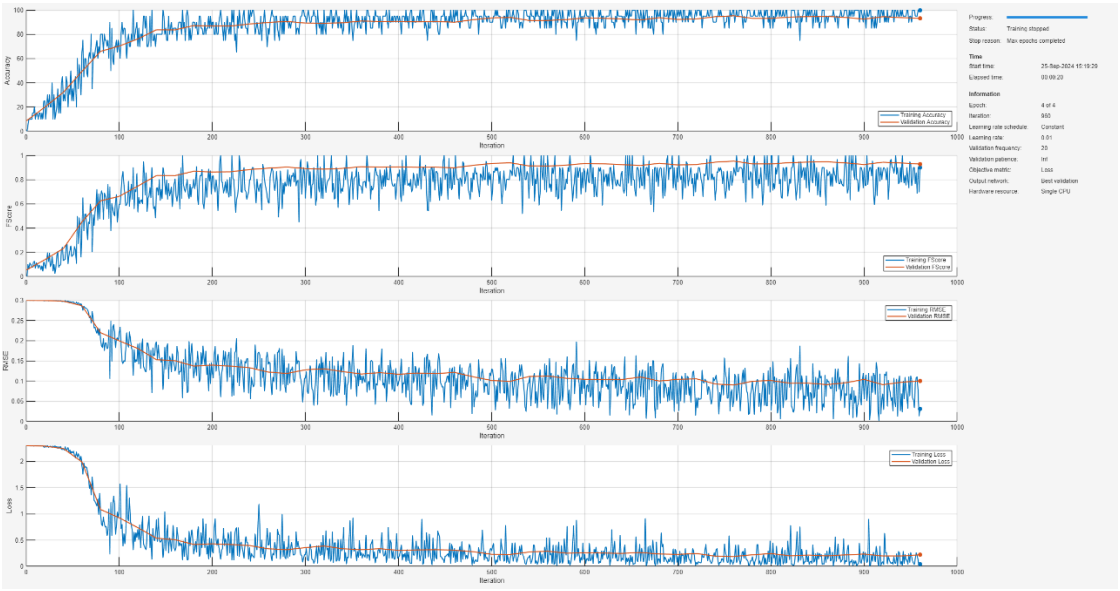
```
epochsize=T(1,1)/b;
```

Αντιστοίχως έγιναν και τα 10%-50% , παρακάτω θα παραθέσω σε πίνακες όπου θα φαίνονται συγκεντρωτικά τις διαφορές στην απόδοση και τις περεταίρω μετρικές.
Πρώτα οι Εικόνες Εκπαίδευσης

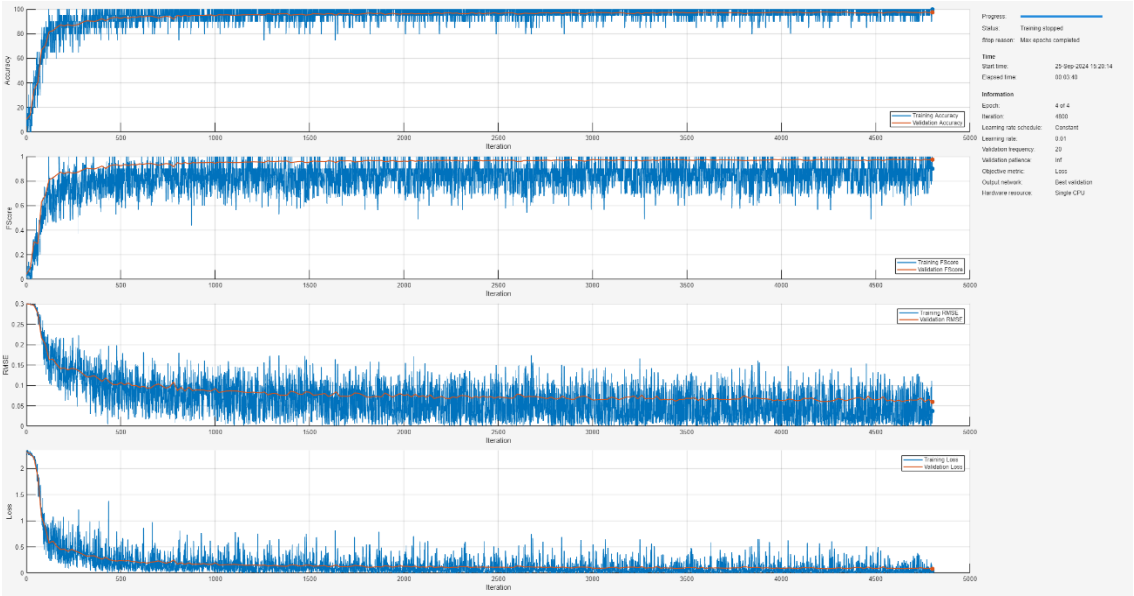
5%


Accuracy
Iteration
Training Accuracy
Validation Accuracy
FScore
Training FScore
Validation FScore
RMSE
Training RMSE
Validation RMSE
Loss
Training Loss
Validation Loss
Progress
Status: Training stopped
Stop reason: Max epochs completed
Time
Information

10%


Accuracy
Iteration
Training Accuracy
Validation Accuracy
FScore
Training FScore
Validation FScore
RMSE
Training RMSE
Validation RMSE
Loss
Training Loss
Validation Loss
Progress
Status: Training stopped
Stop reason: Max epochs completed
Time
Information

50%


Accuracy
Iteration
Training Accuracy
Validation Accuracy
FScore
Training FScore
Validation FScore
RMSE
Training RMSE
Validation RMSE
Loss
Training Loss
Validation Loss
Progress
Status: Training stopped
Stop reason: Max epochs completed
Time
Information

Πίνακας Σύγκρισής Αποδόσεων

Δίκτυα	Iteration	Accuracy	RMSE	Fscore	CrossEntropy
Best 100%	8660	98.5600	0.0468	0.9855	0.0449
Best 50%	4240	98.1000	0.0546	0.9810	0.0621
Best 10%	940	95.3400	0.0829	0.9531	0.1466
Best 5%	460	92.5200	0.1060	0.9245	0.2363

Παρατηρήσεις: Όλος περιέργως χρησιμοποιώντας ακόμη και το 5% των δεδομένων για την εκπαίδευση του δικτύου η πτώση στην επίδοση είναι αξιόλογα μικρή. Θεωρούσα δηλαδή πριν το πείραμα ότι η μείωση στην επίδοση θα ήταν ανάλογη με την μείωση στο μέγεθος του συνόλου (Training- Validation) αλλά διαψεύσθηκα. Το συμπέρασμα που μπορώ να συνάγω από αυτό είναι πως η αρχιτεκτονική του δικτύου μας ή και ίσως γενικότερα τα Συνελικτικά Δίκτυα δεν απαιτούν καθόλου μεγάλο όγκο δεδομένων για την εκπαίδευση τους ,έτσι ώστε να επιφέρουν ακρίβεια της τάξης του 90% (τουλάχιστον για προβλήματα κατηγοριοποίησης τέτοιου είδους και τέτοιας πολυπλοκότητας).

Μέρος Β:

Κατηγοριοποίηση Εικόνων μέσω εξαγωγής χαρακτηριστικών και νευρωνικού ταξινομητή.

Ερώτημα 1: Υπολογισμός χαρακτηριστικών για το σύνολο των δεδομένων.

Προβλήματα & Παρατηρήσεις: Όσον αφορά τα features που θα τροφοδοτούσα ως είσοδο στο Νευρωνικό Δίκτυο επέλεξα το Histogram of Oriented Gradients (HoG), διαβάζοντας τις σχετικές πληροφορίες για την συνάρτηση αυτή στην matlab έφτασα σε μια παραμετροποίηση η οποία ήταν κατάλληλη θεωρώ για την εξαγωγή των χαρακτηριστικών της εικόνας , έχοντας υπόψιν ότι η εικόνα είναι πολύ μικρή 28χ28 μόλις, προσπάθησα να το βελτιστοποιήσω ως προς την εξαγωγή λεπτομέρειας. Παρόλα αυτά προχωρώντας και αφού είχα δημιουργήσει τα δύο νέα structs με τα features του HoG (Validation και Training αντίστοιχα) συνειδητοποίησα πώς ο όγκος των δεδομένων ήταν τεράστιος με αποτέλεσμα αρκετή ώρα αφού προσπαθούσε η matlab να φορτώσει όλα αυτά τα δεδομένα στην είσοδο της συνάρτησης trainnet (εκπαίδευση δικτύου) , εν τέλη να εμφανίζει μήνυμα λάθους OUT OF MEMORY.

```
Out of memory.

Error in featureextract (line 101)
    ValidationData={ValidationHOGset.Features, categorical(validationSet.labels)}, ...

Related documentation
```

Έχοντας υπόψιν αυτό προχώρησα στην αγνόηση των λεπτομερειών στην εικόνα.

	Size	Bytes
Αρχικό	1x1584	6336
Νέο Default παράμετροι	1x144	576
Ελαφρύ	1x48	192



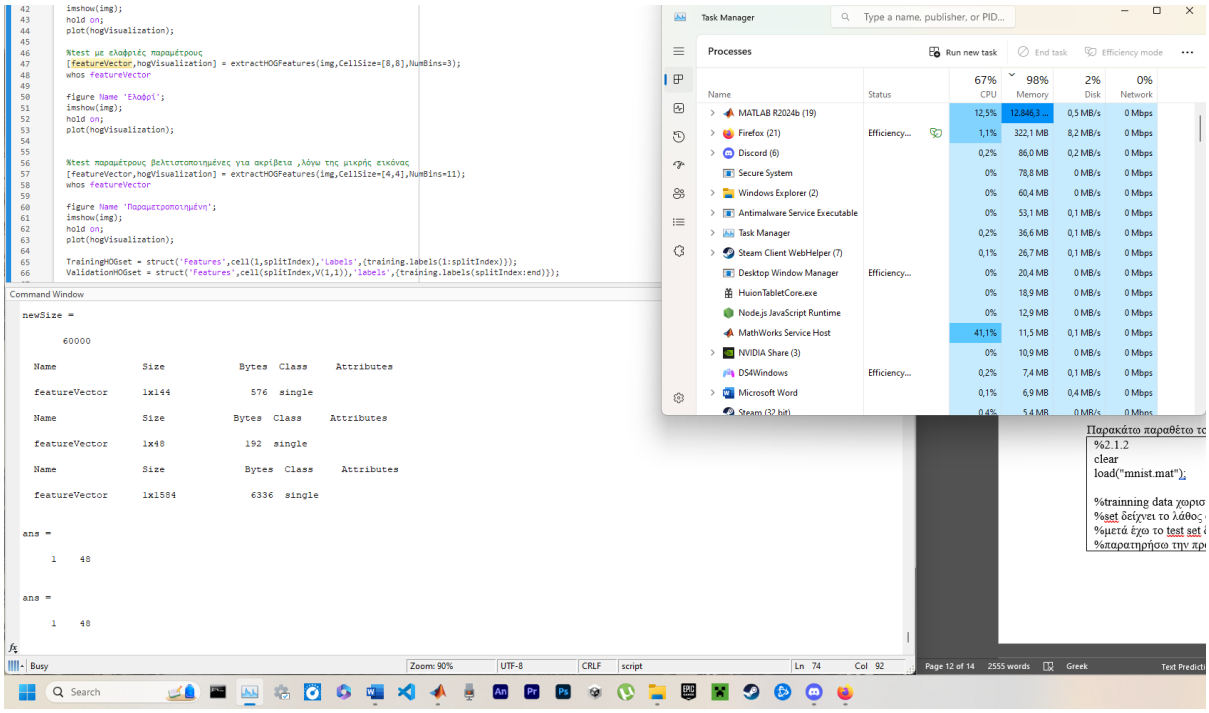
Παρόλα αυτά ακόμα και με το 10% της πληροφορίας (εάν συγκρίνεται το default με το αρχικό Θεωρητικά Βέλτιστο) πάλι ο όγκος ήτανε τεράστιος και πάλι οδηγούμουν στο ίδιο λάθος μην μπορώντας να εκπαιδεύσω το δίκτυό μου.

```
ans =

     1   144


ans =

     1   144

Out of memory.

Error in featureextract (line 99)
    ValidationData={ValidationHOGset.Features, categorical(validationSet.labels)}, ...
                    ^^^^^^^^^^^^^^^^^^^^^^^^^

Related documentation
```

Συνέχισα λοιπόν με όσο πιο ελαφριά παραμετροποίηση μπορούσα η οποία θεωρώ θα είχε κάκιστη επίδοση το δίκτυο αφού δεν περιέχει σχεδόν καθόλου πληροφορία. Δεν μπόρεσα να το επιβεβαιώσω όμως διότι και πάλι ο όγκος ήτανε απαγορευτικός.

```
ans =

     1    48

ans =

     1    48

Out of memory.

Error in elafri (line 109)
    ValidationData={ValidationHOGset.Features, categorical(validationSet.labels)}, ...
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^
Related documentation
```

Εδώ ένα screenshot κατά την διάρκεια της εκτέλεσης του προγράμματος στην Ελαφριά παραμετροποίηση…



Παρακάτω παραθέτω τον αντίστοιχο κώδικα για την υλοποίηση του αποτυχούς ΜέρουςΒ όσον αφορά την δημιουργία των feature και την κατηγοριοποίηση τους σε struct καθώς και τα τεστ. Ολόκληρος ο κώδικας θα υπάρχει στο παράρτημα:

```
%2.1.2
clear
load("mnist.mat");

%trainning data χωρισμένο σε training set και validation όπου το validation
%set δείχνει το λάθος στο νευρονικό δίκτυο
%μετά έχω το test set δεδομένα που δεν έχει ξαναδεί το δίκτυο για να
%παρατηρήσω την πραγματική του απόδοση σε γενικές περιπτώσεις δεδομένων
%εισόδου
%επομένος χωρίζω το training data σε 80-20% training-validation και έχω και το test set μου


percent = 100/100;
wholeSize = 60000;
newSize = wholeSize*percent
```

```
splitIndex = newSize*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet =
struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});   %48.000 εικόνες
για το trainingset
validationSet =
struct('images',{training.images(:,:,splitIndex:newSize)},'labels',{training.labels(splitIndex:newSize)}); %12.
000 εικόνες για validationset

%Μεγέθη του Training και Validation Set
T = size(trainingSet.labels);
V = size(validationSet.labels);

%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)

reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,48000); % Μέγεθος: [28, 28, 1, 48000]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []); % Μέγεθος: [28, 28, 1, 12000]

%τεστ μετατροπής των δεδομένων μου για την είσοδο

img = reshapeTrainingSetimages(:,:,:,1);

%test με default παραμέτρους
[featureVector,hogVisualization] = extractHOGFeatures(img);
whos featureVector

figure Name 'Default';
imshow(img);
hold on;
plot(hogVisualization);

%test με ελαφριές παραμέτρους
[featureVector,hogVisualization] = extractHOGFeatures(img,CellSize=[8,8],NumBins=3);
whos featureVector

figure Name 'Ελαφρί';
imshow(img);
hold on;
plot(hogVisualization);


%test παραμέτρους βελτιστοποιημένες για ακρίβεια ,λόγω της μικρής εικόνας
[featureVector,hogVisualization] = extractHOGFeatures(img,CellSize=[4,4],NumBins=11);
whos featureVector

figure Name 'Παραμετροποιημένη';
imshow(img);
hold on;
plot(hogVisualization);

TrainingHOGset = struct('Features',cell(1,splitIndex),'Labels',{training.labels(1:splitIndex)});
ValidationHOGset = struct('Features',cell(splitIndex,V(1,1)),'labels',{training.labels(splitIndex:end)});

parfor i=1:splitIndex
   feature = extractHOGFeatures(reshapeTrainingSetimages(:,:,:,i),CellSize=[8,8],NumBins=3);
   TrainingHOGset(i).Features = feature;
end
```

```
parfor i=splitIndex:V(1,1)
   feature = extractHOGFeatures(reshapeTrainingSetimages(:,:,:,i),CellSize=[8,8],NumBins=3);
   ValidationHOGset(i).Features = feature;
end

%test μορφής
size(TrainingHOGset(:,1).Features)
size(TrainingHOGset(1).Features)
```

Συμπέρασμα: Επειδή αναλώθηκα πολύ στην προσπάθεια να καταφέρω να χρησιμοποιήσω τα HoG ως feature για την είσοδο του δικτύου μου και εν τέλη η προσπάθεια αυτή απόβηκε άκαρπη , δεν είχα χρόνο να πειραματιστώ με άλλων ειδών features όπως μας προτείνετε και εσείς. Έχοντας παραδεχτεί αυτό το λάθος μου ,πιστεύω πως όσον αφορά τον κώδικα η υλοποίηση του Μέρους Β είναι σωστή, με το πρόβλημα να βρίσκεται στο υλισμικό μου. Βέβαια και εγώ ο ίδιος γνωρίζω όπως θα παρατηρούσατε και εσείς πως εάν το HoG έχει απαγορευτικό υπολογιστικό κόστος για το σύστημα μου θα έπρεπε να διαλέξω ένα ή παραπάνω τέλος πάντων feature/s όπου να ξεπερνούν αυτό το πρόβλημα.
Εφόσον δεν είχα χρόνο να ψάξω και να υλοποιήσω το δίκτυο με άλλα features επέλεξα να συνεχίσω το ΜέροςΒ συγκρίνοντας το συνελικτικό Δίκτυο του Μέρους Α με ένα δίκτυο χωρίς συνελικτικό μέρος και ως είσοδο τις αρχικές εικόνες χωρίς κάποια περεταίρω τροποποίηση.

Ερώτημα 2: Επανάληψη των προηγούμενων ερωτημάτων του Μέρους Α.

Ερώτημα 2.2)
Όσον αφορά το ερώτημα 2 η διαδικασία παραμένει ίδια και δεν υπάρχει λόγος να αναλωθούμε στην επανάληψη της εξήγησης.

Ερώτημα 2.3)
Εδώ είναι απαραίτητο να αφαιρέσουμε τα συνελικτικά επίπεδα από την δήλωση της αρχιτεκτονικής του δικτύου μας επομένως θα μοιάζει κάπως έτσι:

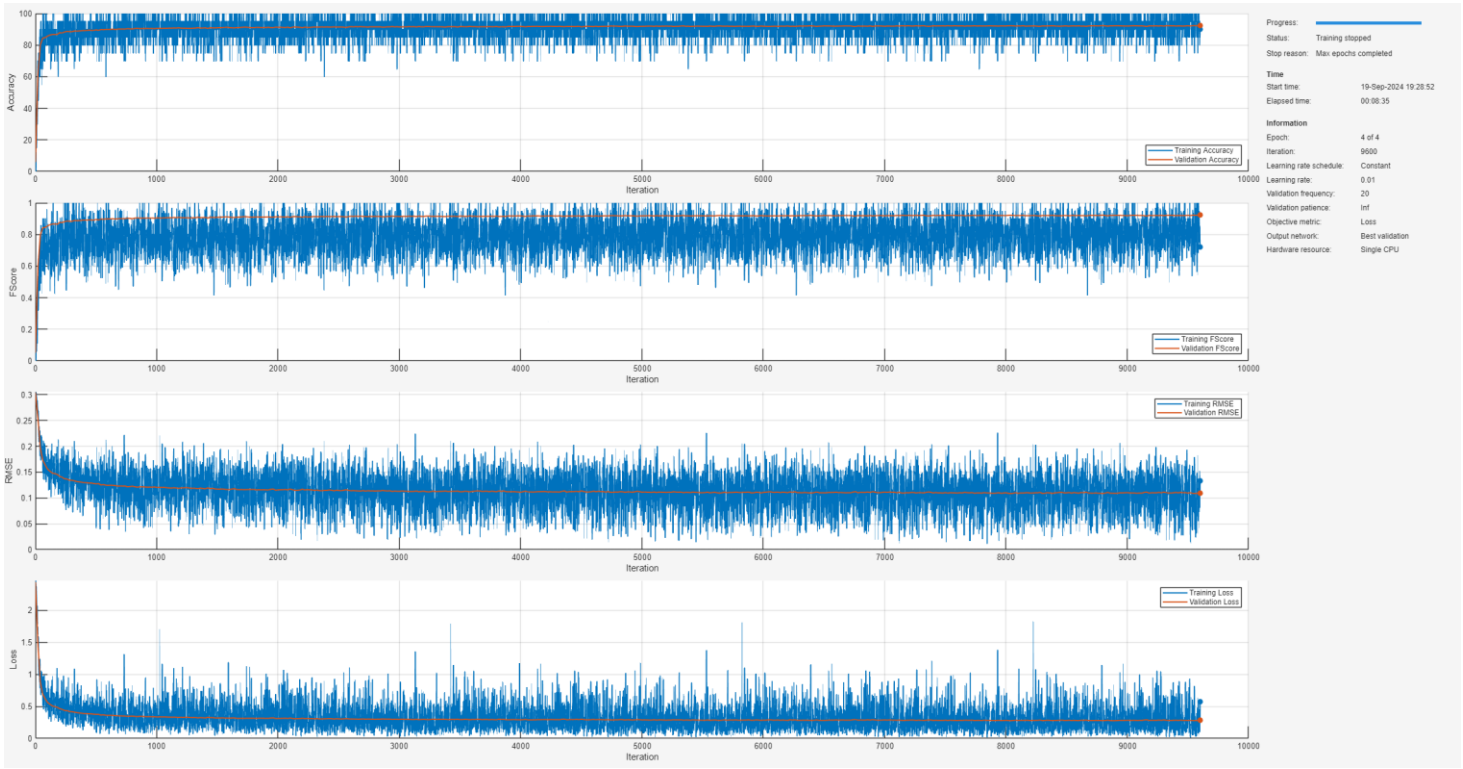
```
layers = [
   imageInputLayer(inputSize)
   fullyConnectedLayer(classes)
   softmaxLayer
  ];
```

Δεν απαιτούνται αλλαγές στην συνέχεια, ο κώδικας παντού είναι της ίδιας λογικής.

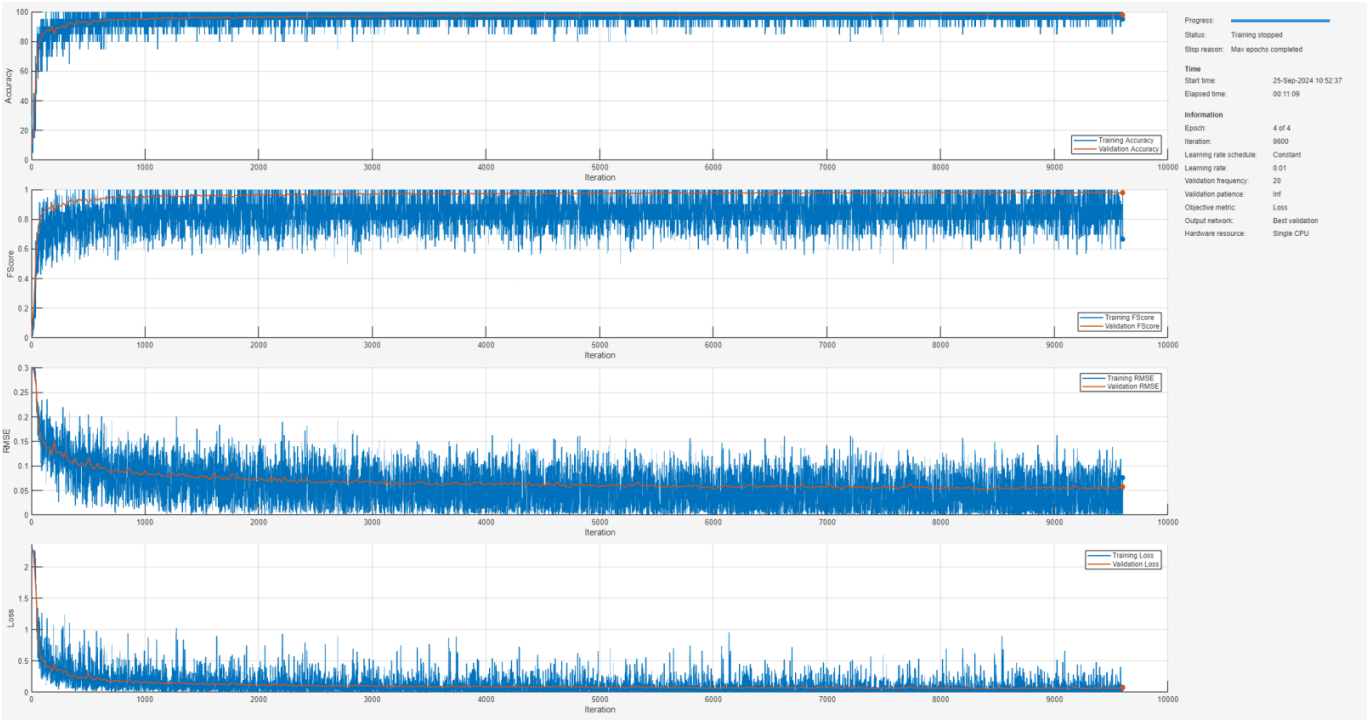
Ερώτημα 3: A vs B.

- Διαδικασία Εκπαίδευσης

Παρακολούθηση της εκπαίδευσης του B:



Σε σύγκριση με αυτή της εκπαίδευσης του A:

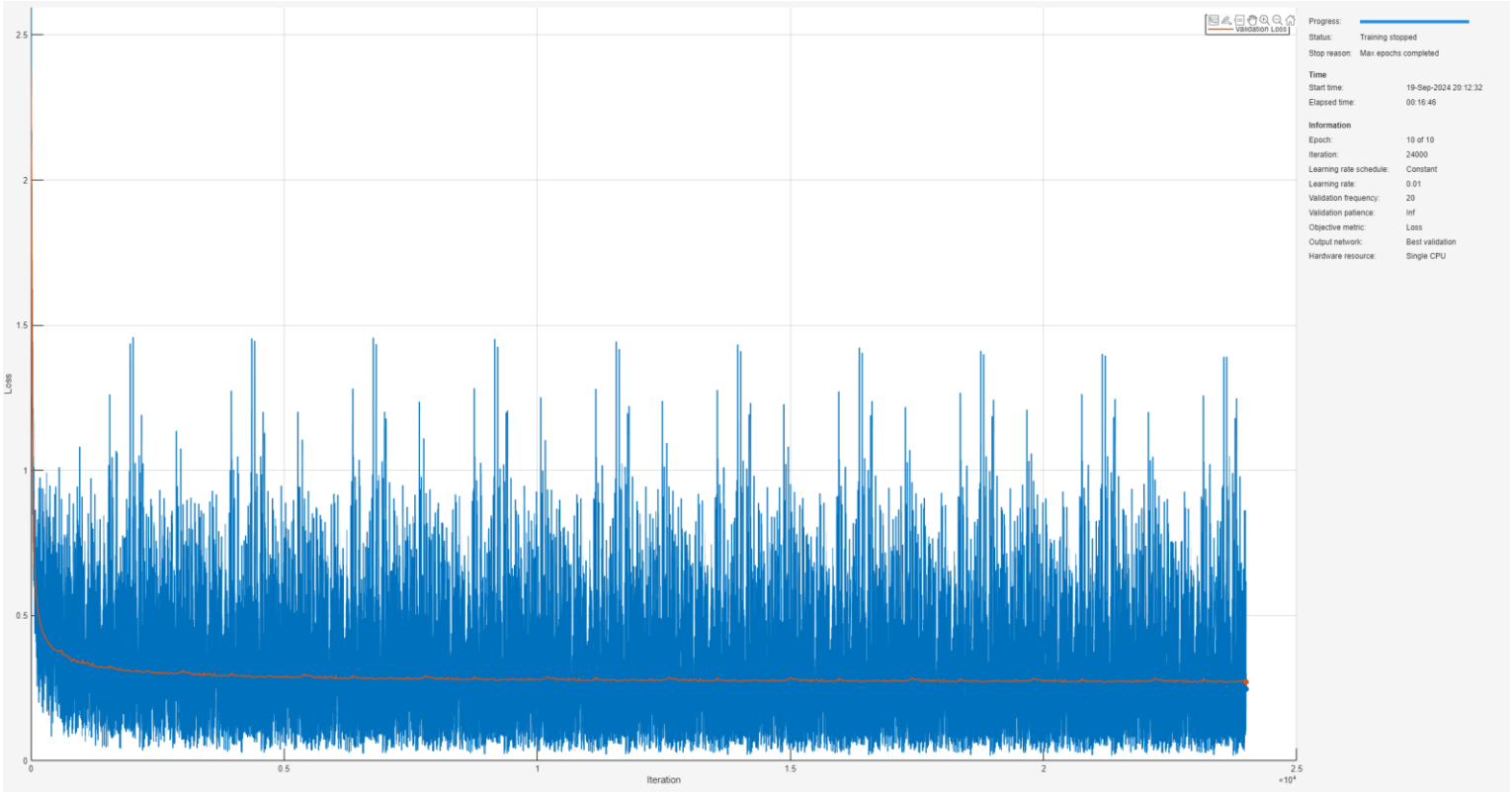


Παρατηρήσεις: Η συμπεριφορά του Validation Loss ως το χρησιμότερο και πιο αξιόπιστο τρόπο για να ελέγξουμε την ικανότητα γενίκευσης του δικτύου και άρα το πραγματικό του σφάλμα , είναι ανεξάρτητη του συνεκτικού ή όχι χαρακτήρα του δικτύου. Ισχύουν αυτά που επώθηκαν και αρχικώς. Η σύκλιση παρόλα αυτά του δικτύου αυτού υστερεί σε σύγκριση με το συνελικτικό, καθώς πρακτικά και τα δύο δίκτυα συγκλίνουν από την πρώτη εποχή με τελική μείωση μόνο ~0.03,~0.025, εκεί όμως που το Α δίκτυο προσέγγιζε το 0.072258 Validation Loss στην 4η εποχή το Β φτάνει μόνο στην τιμή του 0.28066.

Αναλυτικότερη σύγκριση στον πίνακα, με κόκκινο σημειώνονται η τιμές του Β.

Iteration	Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
2400	1	0.62769	90	0.30609	91.501
2400	1	0.10432	95	0.10325	96.992
4800	2	0.59798	90	0.29058	91.934
4800	2	0.11767	95	0.082364	97.525
7200	3	0.58131	90	0.28419	92.242
7200	3	0.10034	95	0.076058	97.767
9600	4	0.56972	90	0.28066	92.359
9600	4	0.079969	95	0.072258	0.97807

Το Β συγκλίνει αρκετά χειρότερα από το Α ανεξαρτήτως τον αριθμό των εποχών για την οποία θα διεξαχθεί η εκπαίδευση, όπως φαίνεται και από την σχετική εικόνα.
(Εάν ανατρέξετε ανωτέρω για να δείτε την αντίστοιχη του Α είναι εμφανές.)



Σύγκλιση για 10 εποχές του Β

- Έλεγχος

Παρατηρώ ότι όσον αφορά την ακρίβεια υστερεί έναντι του Epoch4 , παρόλα αυτά το Loss του πράγματι, είναι μικρότερο. Διαβάζοντας σχετικά με τις δύο μετρικές αυτές κατέληξα ότι είναι προτιμότερη η βελτιστοποίηση ώς προς Loss καθώς μπορεί να κάνει λάθος ελάχιστα λίγο πιο συχνά αλλά κατά μέσον όρο η λανθασμένες αποφάσεις του Best θα απέχουν λιγότερο από την πραγματική απάντηση από αυτές του Epoch4. Άρα θεωρώ το Best ως το βέλτιστο δίκτυο.

Δίκτυα	Iteration	Accuracy	RMSE	Fscore	CrossEntropy
Best	8040	91.9100	0.1099	0.9179	0.2820
Best	8660	98.5600	0.0468	0.9855	0.0449
Epoch 4	9600	92.1600	0.1104	0.9205	0.2853
Epoch 4	9600	98.1900	0.0524	0.9817	0.0571
Epoch 3	7200	92.0100	0.1112	0.9190	0.2884
Epoch 3	7200	98.0700	0.0534	0.9805	0.0600
Epoch 2	4800	91.8800	0.1124	0.9176	0.2939
Epoch 2	4800	98.0800	0.0546	0.9807	0.0638
Epoch 1	2400	91.3900	0.1153	0.9127	0.3081
Epoch 1	2400	97.4200	0.0628	0.9740	0.0835

Στην αναμεταξύ τους σύγκριση (A vs B) είναι λογικό το A να υπερτερεί όπως προοικονομούσαν και τα δεδομένα που είχαμε από την σύγκριση στην εκπαίδευση. Προφανώς το A γενικεύει καλύτερα σύμφωνα με κριτήρια που αυτό το ίδιο οργανικά έχει επιλέξει σε αντίθεση με το B το οποίο έχει στην διάθεση του μόνο τα raw data.

- Απόδοση με υποσύνολο των Δεδομένων Εκπαίδευσης

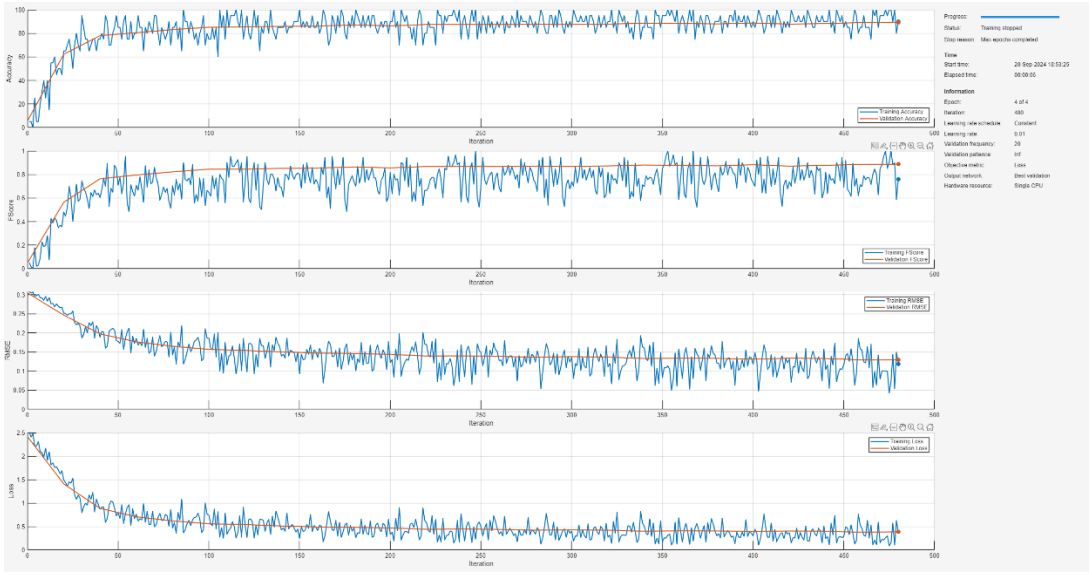
Πίνακας Σύγκρισής Αποδόσεων

Δίκτυα	Iteration	Accuracy	RMSE	Fscore	CrossEntropy
Best 100%	8040	91.9100	0.1099	0.9179	0.2820
Best 50%	4500	91.6900	0.1125	0.9158	0.2924
Best 50%	4240	98.1000	0.0546	0.9810	0.0621
Best 10%	960	89.9000	0.1249	0.8975	0.3539
Best 10%	940	95.3400	0.0829	0.9531	0.1466
Best 5%	460	88.0500	0.1347	0.8788	0.4105
Best 5%	460	92.5200	0.1060	0.9245	0.2363

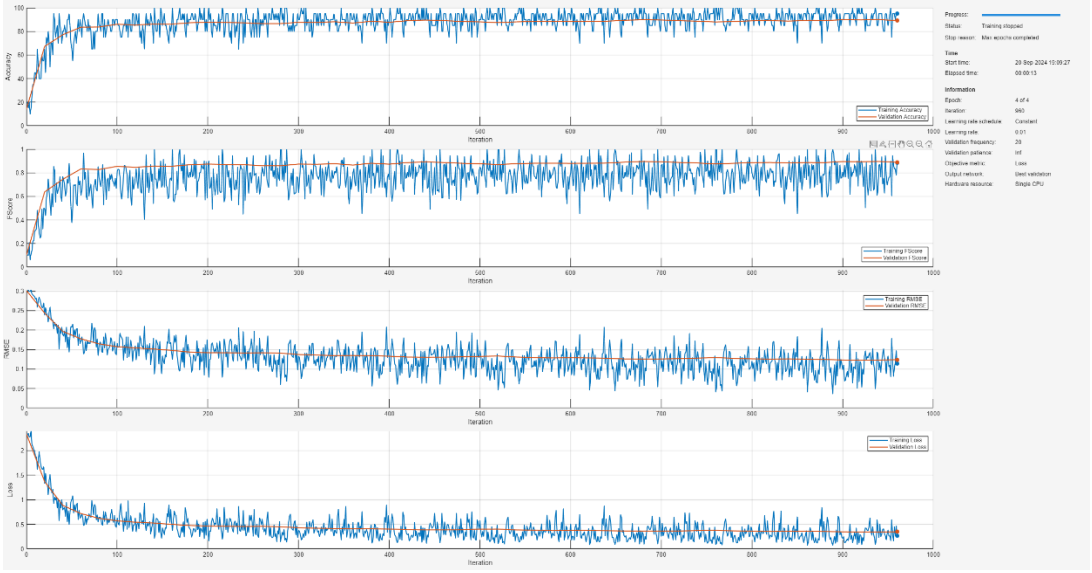
Σχόλια: Παρομοίως το B φαίνεται να μην επηρεάζεται δραστικά από την τεράστια μείωση στο μέγεθος του συνόλου εκπαίδευσης εάν συλλογιστούμε ότι με μείωση 95% η ακρίβεια ελαττώνεται κατά 4.1998% μόνο.
Το απρόσμενο είναι πως φαίνεται το B να επηρεάζεται λιγότερο από το A, αφού η μείωση στην ακρίβεια του A είναι 6.1282% αλλά αυτό θα μπορούσε και να οφείλεται στην επίτευξη εκ μέρους του A μεγαλύτερης συνολικής ακρίβειας.

Ακολουθούν εικόνες του B ως προς τα υποσύνολα:

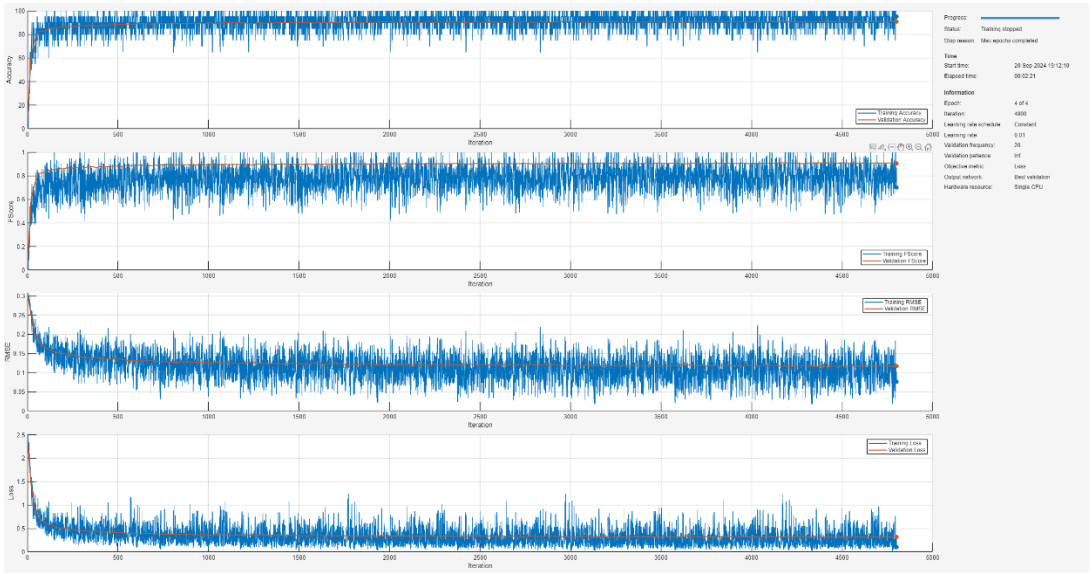
5%



10%



50%



Παράρτημα Κώδικα :

Παρατηρήσεις:

- Θα χρειαστεί να φορτώσετέ στον υπολογιστή σας το αρχείο mnist.mat διότι από εκεί αντλώ τα δεδομένα εκχωρώντας τα μετά σε δικά μου structs.
- Ο κώδικας χρησιμοποιεί συναρτήσεις όπως η testnet η οποίες χρειάζονται την τελευταία έκδοση της matlab 2024b ,επίσης για την αποθήκευση και μετά φόρτωση checkpoint δικτύων χρησιμοποιούνται address paths τα οποία είναι συγκεκριμένα για τον δικό μου υπολογιστή , με συγχωρείται αλλά δεν κατάφερα να βρω έναν γενικό τρόπο.
- Όπως ανέφερα και πάνω **ΔΕΝ** έχω υλοποιήσει τα **Confusion Matrix** και **ΔΕΝ** κατάφερα να προχωρήσω στην εκπαίδευση δικτύου χρησιμοποιώντας **features** λόγω υπέρογκων δεδομένων στην RAM.

Μέρος Α:

Ερώτημα 1:

```
% Θέμα 1
% από http://yann.lecun.com/exdb/mnist/ δεν μπόρεσα να τα κατεβάσω καθώς
% έλεγε ότι δεν κατέχω την άδεια.
%βρήκα από αλλού το ίδιο αρχείο
load("mnist.mat");
who

%Θέμα 1 Ταξινόμηση Εικόνων
%2.1.1

index=ones(1,10).*10; %αρχικοποίηση έτσι ώστε να μην ταιριάζει σε κανένα αριθμό
duplicate=false; %αρχικοποίηση έτσι ώστε η ο πρώτος αριθμός να τυπώνεται
subplotRows = 2; % αριθμός γραμμών στο sublot
subplotCols = 5; % στύλες
plotCount = 1;   % Counter για την θέση στο subplot

figure;

for i = 1:30
   index(i)=training.labels(i);    %αποθηκεύει τους αριθμούς που διαπεράσαμε
   compare = (index(1:plotCount-1) == training.labels(i));  %τηρεί αν ο νέος αριθμός είναι ίδιος με τα πορηγούμενα
   duplicate = ismember(1, compare); %τηρεί το αποτέλεσμα της ισότητας

   if duplicate==false % Εαν δεν είναι διπλότυπο


      % Παρουσιάση των αριθμών
      subplot(subplotRows, subplotCols, plotCount);
      imshow(rescale(training.images(:, :, i), 0, 255));
      colormap('gray');
      colorbar;
      axis square;
      title(sprintf('Αριθμός: %d', training.labels(i)));

      % Move to the next subplot position
      plotCount = plotCount + 1;

      % Σταματάμε στα 10 λόγο του If γνωρίζουμε ότι έχουμε τυπώσει και τα
      % 10 ψηφία  και όχι κάποια διπλότυπα
      if plotCount > 10
         break;
      end
   end
end

%μεταβλητές χρησιμοποιήθηκαν για να ελέγχω το πρόγραμμα πειραματικά διότι
%είχα προβλήματα με την παρουσιάση διπλότυπων αριθμών
index
```

```
compare
duplicate
training.labels(i)
```

Ερώτημα 2-3-4:

```
%2.1.2
clear
load("mnist.mat");

%trainning data χωρισμένο σε training set και validation όπου το validation
%set δείχνει το λάθος στο νευρονικό δίκτυο
%μετά έχω το test set δεδομένα που δεν έχει ξαναδεί το δίκτυο για να
%παρατηρήσω την πραγματική του απόδοση σε γενικές περιπτώσεις δεδομένων
%εισόδου
%επομένος χωρίζω το training data σε 80-20% training-validation και έχω και το test set μου


splitIndex = 60000*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});    %48.000 εικόνες για το
trainingset
validationSet = struct('images',{training.images(:,:,splitIndex:end)},'labels',{training.labels(splitIndex:end)});  %12.000 εικόνες για
validationset

%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)

reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,48000);  % Μέγεθος: [28, 28, 1, 48000]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []);  % Μέγεθος: [28, 28, 1, 12000]

%τεστ μετατροπής των δεδομένων μου για την είσοδο

imagestart= size(trainingSet.images)
imagereshaped= size(reshapeTrainingSetimages)
imagelabels= size(trainingSet.labels)

% x=2;
%
% figure
%     image (rescale(training.images(:,:,x),0,255));
%     colormap("gray");
%     colorbar
%     axis square equal
%     title(training.labels(x))
%
% figure
%     image (rescale(trainingSet.images(:,:,x),0,255));
%     colormap("gray");
%     colorbar
%     axis square equal
%     title(trainingSet.labels(x))


%2.1.3

epoch =4;   %εποχές
b =20;  %minibatchsize
epochsize=48000/b;
inputSize =[28 28 1];    %σύμφωνα με τα χαρακτηριστικά των εικόνων mnist
classes = numel(unique(training.labels));   %10 κλάσεις

%ορισμός επιπέδων του δικτύου , γνωρίζω ότι δεν αναφέρονται ρητά τα softmax
%και classification επίπεδα παρόλα αυτά χωρίς την ποαρουσία τους το δίκτυο
%δεν μαθαίνει τίποτα ,έχω και σχετικό παράδειγμα στην αναφορά

layers = [
    imageInputLayer(inputSize)
    convolution2dLayer([5,5],6)
    reluLayer
    averagePooling2dLayer([2,2],"Stride",[2,2])
```

```
    convolution2dLayer([5,5],12)
    reluLayer
    averagePooling2dLayer([2,2],"Stride",[2,2])
    fullyConnectedLayer(classes)
    softmaxLayer
   ];

options = trainingOptions("sgdm", ...
    MiniBatchSize=b, ...
    MaxEpochs=epoch, ...
    Metrics=["accuracy","fscore","rmse"], ...
    Verbose=true, ...
    VerboseFrequency=20,...
    ValidationData={reshapeValidationSetimages, categorical(validationSet.labels)}, ...
    ValidationFrequency=b, ...
    OutputNetwork="best-validation", ...
    ExecutionEnvironment="cpu",...
    Plots="training-progress", ...
    CheckpointPath="E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ", ...
    CheckPointFrequency=1 ...
    );

%το έφτιαξε το categorical στο trainingSet.labels αλλιώς buggαρε

[diktyo,info] = trainnet(reshapeTrainingSetimages,categorical(trainingSet.labels),layers,'crossentropy',options);

show(info)


%δείχνω τις τιμές των μετρικών στο τέλος κάθε εποχής
for i=1:4
    blabla=["metrics from epoch",i];
    disp(blabla)
    TrainingHistory=info.TrainingHistory(i*epochsize,:)
    ValidationHistory=info.ValidationHistory(i*epochsize/20 +1,:)
end

%info.TrainingHistory
%info.ValidationHistory
BestNetworkIterration = info.OutputNetworkIteration


%2.1.4

reshapeTestSetImages = reshape(test.images, 28, 28, 1, []); %το testSet μου Μέγεθους: [28, 28, 1, 10000]
diktyoepoch4 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__9600__2024_09_25__11_03_46.mat");
diktyoepoch3 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__7200__2024_09_25__11_01_01.mat");
diktyoepoch2 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__4800__2024_09_25__10_58_16.mat");
diktyoepoch1 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__2400__2024_09_25__10_55_37.mat");

testbest = testnet(diktyo,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatch Size=b)
testepoch4 =
testnet(diktyoepoch4,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
testepoch3 =
testnet(diktyoepoch3,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
testepoch2 =
testnet(diktyoepoch2,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
testepoch1 =
testnet(diktyoepoch1,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)

%%
% classes2 = numel(unique(training.labels));   %10 κλάσεις
%
%
% predictedLabels = minibatchpredict(diktyo, reshapeTestSetImages,MiniBatchSize=b);
% predictedLabelsDecoded = onehotdecode(predictedLabels,classes2,2,"double");
%
% whos predictedLabelsDecoded
% reshapedPredictedLabelsDecoded = reshape(predictedLabelsDecoded,[],1);
% whos reshapedPredictedLabelsDecoded
%
```

```
% trueLabels =test.labels;
%
%
% % Display the confusion matrix as a chart
% confusionchart(trueLabels, PredictedLabelsDecoded);
```

Ερώτημα 5:

```
%2.1.5
%Τα παρακάτω βήματα είναι τα βήματα 2-4 χρησιμοποιόντας το 5% του συνόλου

%2.1.2
clear
load("mnist.mat");

percent = 5/100;
wholeSize = 60000;
newSize = wholeSize*percent
splitIndex = newSize*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});    %48.000 εικόνες για το
trainingset
validationSet = struct('images',{training.images(:,:,splitIndex:newSize)},'labels',{training.labels(splitIndex:newSize)}); %12.000 εικόνες
για validationset

%Μεγέθη του Training και Validation Set
T = size(trainingSet.labels)
V = size(validationSet.labels)
%%
%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)
reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,[]);  % Μέγεθος: [28, 28, 1, T(1,1)]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []);  % Μέγεθος: [28, 28, 1, V(1,1)]

%Επικύρωση ακεραιότητας
size(reshapeTrainingSetimages)
size(reshapeValidationSetimages)



%%

%2.1.3

epoch =4;   %εποχές
b =20;  %minibatchsize
epochsize=T(1,1)/b;
inputSize =[28 28 1];    %σύμφωνα με τα χαρακτηριστικά των εικόνων mnist
classes = numel(unique(training.labels));   %10 κλάσεις

%ορισμός επιπέδων του δικτύου , γνωρίζω ότι δεν αναφέρονται ρητά τα softmax
%και classification επίπεδα παρόλα αυτά χωρίς την ποαρουσία τους το δίκτυο
%δεν μαθαίνει τίποτα ,έχω και σχετικό παράδειγμα στην αναφορά

layers = [
    imageInputLayer(inputSize)
    convolution2dLayer([5,5],6)
    reluLayer
    averagePooling2dLayer([2,2],"Stride",[2,2])
    convolution2dLayer([5,5],12)
    reluLayer
    averagePooling2dLayer([2,2],"Stride",[2,2])
    fullyConnectedLayer(classes)
    softmaxLayer
   ];

options = trainingOptions("sgdm", ...
    MiniBatchSize=b, ...
    MaxEpochs=epoch, ...
    Metrics=["accuracy","fscore","rmse"], ...
```

```
    Verbose=true, ...
    VerboseFrequency=20,...
    ValidationData={reshapeValidationSetimages, categorical(validationSet.labels)}, ...
    ValidationFrequency=b, ...
    OutputNetwork="best-validation", ...
    ExecutionEnvironment="cpu",...
    Plots="training-progress", ...
    CheckpointPath="E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ", ...
    CheckPointFrequency=1 ...
    );

%το έφτιαξε το categorical στο trainingSet.labels αλλιώς buggαρε

[diktyo,info] = trainnet(reshapeTrainingSetimages,categorical(trainingSet.labels),layers,'crossentropy',options);

show(info)


%δείχνω τις τιμές των μετρικών στο τέλος κάθε εποχής
for i=1:4
    blabla=["metrics from epoch",i];
    disp(blabla)
    TrainingHistory=info.TrainingHistory(i*epochsize,:)
    ValidationHistory=info.ValidationHistory(i*epochsize/20 +1,:)
end

%info.TrainingHistory
%info.ValidationHistory
BestNetworkIterration = info.OutputNetworkIteration


%2.1.4

reshapeTestSetImages = reshape(test.images, 28, 28, 1, []); %το testSet μου Μέγεθους: [28, 28, 1, 10000]

testbest = testnet(diktyo,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatch Size=b)
```

Μέρος Β:

Ερώτημα 2-3-4 Αποτυχία Feature Extract:

```
%2.1.2
clear
load("mnist.mat");

%trainning data χωρισμένο σε training set και validation όπου το validation
%set δείχνει το λάθος στο νευρονικό δίκτυο
%μετά έχω το test set δεδομένα που δεν έχει ξαναδεί το δίκτυο για να
%παρατηρήσω την πραγματική του απόδοση σε γενικές περιπτώσεις δεδομένων
%εισόδου
%επομένος χωρίζω το training data σε 80-20% training-validation και έχω και το test set μου


percent = 100/100;
wholeSize = 60000;
newSize = wholeSize*percent
splitIndex = newSize*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});    %48.000 εικόνες για το
trainingset
validationSet = struct('images',{training.images(:,:,splitIndex:newSize)},'labels',{training.labels(splitIndex:newSize)}); %12.000 εικόνες
για validationset

%Μεγέθη του Training και Validation Set
T = size(trainingSet.labels);
V = size(validationSet.labels);

%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
```

```
%(βάθος=3)

reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,48000);  % Μέγεθος: [28, 28, 1, 48000]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []);  % Μέγεθος: [28, 28, 1, 12000]

%τεστ μετατροπής των δεδομένων μου για την είσοδο

img = reshapeTrainingSetimages(:,:,:,1);

%test με default παραμέτρους
[featureVector,hogVisualization] = extractHOGFeatures(img);
whos featureVector

figure Name 'Default';
imshow(img);
hold on;
plot(hogVisualization);

%test με ελαφριές παραμέτρους
[featureVector,hogVisualization] = extractHOGFeatures(img,CellSize=[8,8],NumBins=3);
whos featureVector

figure Name 'Ελαφρί';
imshow(img);
hold on;
plot(hogVisualization);


%test παραμέτρους βελτιστοποιημένες για ακρίβεια ,λόγω της μικρής εικόνας
[featureVector,hogVisualization] = extractHOGFeatures(img,CellSize=[4,4],NumBins=11);
whos featureVector

figure Name 'Παραμετροποιημένη';
imshow(img);
hold on;
plot(hogVisualization);

TrainingHOGset = struct('Features',cell(1,splitIndex),'Labels',{training.labels(1:splitIndex)});
ValidationHOGset = struct('Features',cell(splitIndex,V(1,1)),'labels',{training.labels(splitIndex:end)});

parfor i=1:splitIndex
    feature = extractHOGFeatures(reshapeTrainingSetimages(:,:,:,i),CellSize=[8,8],NumBins=3);
    TrainingHOGset(i).Features = feature;
end

parfor i=splitIndex:V(1,1)
    feature = extractHOGFeatures(reshapeTrainingSetimages(:,:,:,i),CellSize=[8,8],NumBins=3);
    ValidationHOGset(i).Features = feature;
end

%test μορφής
size(TrainingHOGset(:,1).Features)
size(TrainingHOGset(1).Features)

%%

%2.1.3

epoch =4;   %εποχές
b =20;  %minibatchsize
epochsize=T(1,1)/b;
inputSize =[28 28 1];    %σύμφωνα με τα χαρακτηριστικά των εικόνων mnist
classes = numel(unique(training.labels));   %10 κλάσεις


%ορισμός επιπέδων του δικτύου , γνωρίζω ότι δεν αναφέρονται ρητά τα softmax
%και classification επίπεδα παρόλα αυτά χωρίς την ποαρουσία τους το δίκτυο
%δεν μαθαίνει τίποτα ,έχω και σχετικό παράδειγμα στην αναφορά

layers = [
    featureInputLayer(1)
    fullyConnectedLayer(classes)
    softmaxLayer
   ];
```

```
options = trainingOptions("sgdm", ...
    MiniBatchSize=b, ...
    MaxEpochs=epoch, ...
    Metrics=["accuracy","fscore","rmse"], ...
    Verbose=true, ...
    VerboseFrequency=20,...
    ValidationData={ValidationHOGset.Features, categorical(validationSet.labels)}, ...
    ValidationFrequency=b, ...
    OutputNetwork="best-validation", ...
    ExecutionEnvironment="cpu",...
    Plots="training-progress", ...
    CheckpointPath="E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ\featureextract", ...
    CheckPointFrequency=1 ...
    );

%το έφτιαξε το categorical στο trainingSet.labels αλλιώς buggαρε

[diktyo,info] = trainnet(TrainingHOGset.Features,categorical(trainingSet.labels),layers,'crossentropy',options);

show(info)

%%

%δείχνω τις τιμές των μετρικών στο τέλος κάθε εποχής
for i=1:4
    blabla=["metrics from epoch",i];
    disp(blabla)
    TrainingHistory=info.TrainingHistory(i*epochsize,:)
    ValidationHistory=info.ValidationHistory(i*epochsize/20 +1,:)
end

%info.TrainingHistory
%info.ValidationHistory
BestNetworkIterration = info.OutputNetworkIteration

%%

%2.1.4

reshapeTestSetImages = reshape(test.images, 28, 28, 1, []); %το testSet μου Μέγεθους: [28, 28, 1, 10000]
TestSize = size(test.labels)

TestHOGset = struct('Features',cell(1,TestSize(1,1)));

parfor i=1:TestSize(1,1)
    feature = extractHOGFeatures(reshapeTestSetImages(:,:,:,i));
    TestHOGset(i).Features = feature;
end

% diktyoepoch4 = importdata();
% diktyoepoch3 = importdata();
% diktyoepoch2 = importdata();
% diktyoepoch1 = importdata();

%δεν θα δουλέψει καθώς δεν υπάρχει αρκετός χώρος στη ram για να εκπαιδευτεί
%το δίκτυο
testbest = testnet(diktyo,TestHOGset.Features,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
% testepoch4 =
testnet(diktyoepoch4,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
% testepoch3 =
testnet(diktyoepoch3,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
% testepoch2 =
testnet(diktyoepoch2,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
% testepoch1 =
testnet(diktyoepoch1,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
```

Ερώτημα 2-3-4 Ενναλακτικής Εισόδου χωρίς Συνέλιξη:

```
%2.1.2
clear
load("mnist.mat");

%trainning data χωρισμένο σε training set και validation όπου το validation
%set δείχνει το λάθος στο νευρονικό δίκτυο
```

```
%μετά έχω το test set δεδομένα που δεν έχει ξαναδεί το δίκτυο για να
%παρατηρήσω την πραγματική του απόδοση σε γενικές περιπτώσεις δεδομένων
%εισόδου
%επομένος χωρίζω το training data σε 80-20% training-validation και έχω και το test set μου


percent = 100/100;
wholeSize = 60000;
newSize = wholeSize*percent
splitIndex = newSize*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});    %48.000 εικόνες για το
trainingset
validationSet = struct('images',{training.images(:,:,splitIndex:newSize)},'labels',{training.labels(splitIndex:newSize)}); %12.000 εικόνες
για validationset

%Μεγέθη του Training και Validation Set
T = size(trainingSet.labels)
V = size(validationSet.labels)

%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)
reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,[]);  % Μέγεθος: [28, 28, 1, T(1,1)]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []);  % Μέγεθος: [28, 28, 1, V(1,1)]

%Επικύρωση ακεραιότητας
size(reshapeTrainingSetimages)
size(reshapeValidationSetimages)
%τεστ μετατροπής των δεδομένων μου για την είσοδο

imagestart= size(trainingSet.images)
imagereshaped= size(reshapeTrainingSetimages)
imagelabels= size(trainingSet.labels)

% x=2;
%
% figure
%     image (rescale(training.images(:,:,x),0,255));
%     colormap("gray");
%     colorbar
%     axis square equal
%     title(training.labels(x))
%
% figure
%     image (rescale(trainingSet.images(:,:,x),0,255));
%     colormap("gray");
%     colorbar
%     axis square equal
%     title(trainingSet.labels(x))


%2.1.3

epoch =4;   %εποχές
b =20;  %minibatchsize
epochsize=T(1,1)/b;
inputSize =[28 28 1];    %σύμφωνα με τα χαρακτηριστικά των εικόνων mnist
classes = numel(unique(training.labels));   %10 κλάσεις


%ορισμός επιπέδων του δικτύου , γνωρίζω ότι δεν αναφέρονται ρητά τα softmax
%και classification επίπεδα παρόλα αυτά χωρίς την ποαρουσία τους το δίκτυο
%δεν μαθαίνει τίποτα ,έχω και σχετικό παράδειγμα στην αναφορά

layers = [
    imageInputLayer(inputSize)
    fullyConnectedLayer(classes)
    softmaxLayer
   ];

options = trainingOptions("sgdm", ...
    MiniBatchSize=b, ...
    MaxEpochs=epoch, ...
```

```
    Metrics=["accuracy","fscore","rmse"], ...
    Verbose=true, ...
    VerboseFrequency=20,...
    ValidationData={reshapeValidationSetimages, categorical(validationSet.labels)}, ...
    ValidationFrequency=b, ...
    OutputNetwork="best-validation", ...
    ExecutionEnvironment="cpu",...
    Plots="training-progress", ...
    CheckpointPath="E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ", ...
    CheckPointFrequency=1 ...
    );

%το έφτιαξε το categorical στο trainingSet.labels αλλιώς buggαρε

[diktyo,info] = trainnet(reshapeTrainingSetimages,categorical(trainingSet.labels),layers,'crossentropy',options);

show(info)


%δείχνω τις τιμές των μετρικών στο τέλος κάθε εποχής
for i=1:4
    blabla=["metrics from epoch",i];
    disp(blabla)
    TrainingHistory=info.TrainingHistory(i*epochsize,:)
    ValidationHistory=info.ValidationHistory(i*epochsize/20 +1,:)
end

%info.TrainingHistory
%info.ValidationHistory
BestNetworkIterration = info.OutputNetworkIteration


%2.1.4

reshapeTestSetImages = reshape(test.images, 28, 28, 1, []); %το testSet μου Μέγεθους: [28, 28, 1, 10000]
diktyoepoch4 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__9600__2024_09_18__15_02_06.mat");
diktyoepoch3 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__7200__2024_09_18__14_59_56.mat");
diktyoepoch2 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__4800__2024_09_18__14_57_45.mat");
diktyoepoch1 = importdata("E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση
Εικόνας\ΜέροςΒ\net_checkpoint__2400__2024_09_18__14_55_39.mat");

testbest = testnet(diktyo,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatch Size=b)
testepoch4 =
testnet(diktyoepoch4,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
testepoch3 =
testnet(diktyoepoch3,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
testepoch2 =
testnet(diktyoepoch2,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
testepoch1 =
testnet(diktyoepoch1,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
```

Ερώτημα 5:

```
%2.1.5
%Τα παρακάτω βήματα είναι τα βήματα 2-4 χρησιμοποιόντας το 5% του συνόλου

%2.1.2
clear
load("mnist.mat");

percent = 5/100;
wholeSize = 60000;
newSize = wholeSize*percent
splitIndex = newSize*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});    %48.000 εικόνες για το
trainingset
validationSet = struct('images',{training.images(:,:,splitIndex:newSize)},'labels',{training.labels(splitIndex:newSize)}); %12.000 εικόνες
για validationset

%Μεγέθη του Training και Validation Set
```

```
T = size(trainingSet.labels)
V = size(validationSet.labels)
%%
%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)
reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,[]);  % Μέγεθος: [28, 28, 1, T(1,1)]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []);  % Μέγεθος: [28, 28, 1, V(1,1)]

%Επικύρωση ακεραιότητας
size(reshapeTrainingSetimages)
size(reshapeValidationSetimages)



%%

%2.1.3

epoch =4;   %εποχές
b =20;  %minibatchsize
epochsize=T(1,1)/b;
inputSize =[28 28 1];    %σύμφωνα με τα χαρακτηριστικά των εικόνων mnist
classes = numel(unique(training.labels));   %10 κλάσεις

%ορισμός επιπέδων του δικτύου , γνωρίζω ότι δεν αναφέρονται ρητά τα softmax
%και classification επίπεδα παρόλα αυτά χωρίς την ποαρουσία τους το δίκτυο
%δεν μαθαίνει τίποτα ,έχω και σχετικό παράδειγμα στην αναφορά

layers = [
    imageInputLayer(inputSize)
    fullyConnectedLayer(classes)
    softmaxLayer
   ];

options = trainingOptions("sgdm", ...
    MiniBatchSize=b, ...
    MaxEpochs=epoch, ...
    Metrics=["accuracy","fscore","rmse"], ...
    Verbose=true, ...
    VerboseFrequency=20,...
    ValidationData={reshapeValidationSetimages, categorical(validationSet.labels)}, ...
    ValidationFrequency=b, ...
    OutputNetwork="best-validation", ...
    ExecutionEnvironment="cpu",...
    Plots="training-progress", ...
    CheckpointPath="E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ", ...
    CheckPointFrequency=1 ...
    );

%το έφτιαξε το categorical στο trainingSet.labels αλλιώς buggαρε

[diktyo,info] = trainnet(reshapeTrainingSetimages,categorical(trainingSet.labels),layers,'crossentropy',options);

show(info)


%δείχνω τις τιμές των μετρικών στο τέλος κάθε εποχής
for i=1:4
    blabla=["metrics from epoch",i];
    disp(blabla)
    TrainingHistory=info.TrainingHistory(i*epochsize,:)
    ValidationHistory=info.ValidationHistory(i*epochsize/20 +1,:)
end

%info.TrainingHistory
%info.ValidationHistory
BestNetworkIterration = info.OutputNetworkIteration


%2.1.4

reshapeTestSetImages = reshape(test.images, 28, 28, 1, []); %το testSet μου Μέγεθους: [28, 28, 1, 10000]

testbest = testnet(diktyo,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatch Size=b)
```

Άλλα:

Test 10 εποχές:

```
%2.1.2
clear
load("mnist.mat");

%trainning data χωρισμένο σε training set και validation όπου το validation
%set δείχνει το λάθος στο νευρονικό δίκτυο
%μετά έχω το test set δεδομένα που δεν έχει ξαναδεί το δίκτυο για να
%παρατηρήσω την πραγματική του απόδοση σε γενικές περιπτώσεις δεδομένων
%εισόδου
%επομένος χωρίζω το training data σε 80-20% training-validation και έχω και το test set μου


splitIndex = 60000*0.8; % σημείο διαίρεσης του struct που κρατάει τα training data

% τα δυο νέα structs
trainingSet = struct('images',{training.images(:,:,1:splitIndex)},'labels',{training.labels(1:splitIndex)});    %48.000 εικόνες για το
trainingset
validationSet = struct('images',{training.images(:,:,splitIndex:end)},'labels',{training.labels(splitIndex:end)});  %12.000 εικόνες για
validationset

%μετατροπή τους ετσί ώστε να φορτώνονται σωστά στην trainnet ,
% ουσιαστικά απλά δηλώνω ότι έχουνε βάθος=1 δηλαδή είναι greyscale και όχι rgb
%(βάθος=3)

reshapeTrainingSetimages= reshape(trainingSet.images,28,28,1,48000);  % Μέγεθος: [28, 28, 1, 48000]
reshapeValidationSetimages = reshape(validationSet.images, 28, 28, 1, []);  % Μέγεθος: [28, 28, 1, 12000]

%τεστ μετατροπής των δεδομένων μου για την είσοδο

imagestart= size(trainingSet.images)
imagereshaped= size(reshapeTrainingSetimages)
imagelabels= size(trainingSet.labels)

% x=2;
%
% figure
%     image (rescale(training.images(:,:,x),0,255));
%     colormap("gray");
%     colorbar
%     axis square equal
%     title(training.labels(x))
%
% figure
%     image (rescale(trainingSet.images(:,:,x),0,255));
%     colormap("gray");
%     colorbar
%     axis square equal
%     title(trainingSet.labels(x))


%2.1.3

epoch =10;   %εποχές
b =20;  %minibatchsize
epochsize=48000/b;
inputSize =[28 28 1];    %σύμφωνα με τα χαρακτηριστικά των εικόνων mnist
classes = numel(unique(training.labels));   %10 κλάσεις

%ορισμός επιπέδων του δικτύου , γνωρίζω ότι δεν αναφέρονται ρητά τα softmax
%και classification επίπεδα παρόλα αυτά χωρίς την ποαρουσία τους το δίκτυο
%δεν μαθαίνει τίποτα ,έχω και σχετικό παράδειγμα στην αναφορά

layers = [
    imageInputLayer(inputSize)
    convolution2dLayer([5,5],6)
    reluLayer
```

```
    averagePooling2dLayer([2,2],"Stride",[2,2])
    convolution2dLayer([5,5],12)
    reluLayer
    averagePooling2dLayer([2,2],"Stride",[2,2])
    fullyConnectedLayer(classes)
    softmaxLayer
   ];

options = trainingOptions("sgdm", ...
    MiniBatchSize=b, ...
    MaxEpochs=epoch, ...
    Verbose=true, ...
    VerboseFrequency=20,...
    ValidationData={reshapeValidationSetimages, categorical(validationSet.labels)}, ...
    ValidationFrequency=b, ...
    OutputNetwork="best-validation", ...
    ExecutionEnvironment="cpu",...
    Plots="training-progress", ...
    CheckpointPath="E:\Ioannis~Kazixis\CEID\H-Εξάμηνο\Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας\ΜέροςΒ", ...
    CheckPointFrequency=1 ...
    );

%το έφτιαξε το categorical στο trainingSet.labels αλλιώς buggαρε

[diktyo,info] = trainnet(reshapeTrainingSetimages,categorical(trainingSet.labels),layers,'crossentropy',options);

show(info)


%δείχνω τις τιμές των μετρικών στο τέλος κάθε εποχής
for i=1:4
    blabla=["metrics from epoch",i];
    disp(blabla)
    TrainingHistory=info.TrainingHistory(i*epochsize,:)
    ValidationHistory=info.ValidationHistory(i*epochsize/20 +1,:)
end

%info.TrainingHistory
%info.ValidationHistory
BestNetworkIterration = info.OutputNetworkIteration


%2.1.4

reshapeTestSetImages = reshape(test.images, 28, 28, 1, []); %το testSet μου Μέγεθους: [28, 28, 1, 10000]
% diktyoepoch4 = importdata();
% diktyoepoch3 = importdata();
% diktyoepoch2 = importdata();
% diktyoepoch1 = importdata();

testbest = testnet(diktyo,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatch Size=b)
% testepoch4 =
testnet(diktyoepoch4,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
% testepoch3 =
testnet(diktyoepoch3,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
% testepoch2 =
testnet(diktyoepoch2,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
% testepoch1 =
testnet(diktyoepoch1,reshapeTestSetImages,categorical(test.labels),["accuracy","rmse","fscore","crossentropy"],MiniBatchSize=b)
```

Υπολογισμός Ελάττωσης Ακρίβειας:

```
%B
accuracy_best_B = 91.91
accuracy_worse_B =88.05
blaB= (accuracy_worse_B/accuracy_best_B)*100
accuracy_loss_B=100-blaB

%A
accuracy_best_A = 98.56
accuracy_worse_A =92.52
blaA= (accuracy_worse_A/accuracy_best_A)*100
accuracy_loss_A=100-blaA
```