

1^η Εργασία Υπολογιστικής Νοημοσύνης

Ιωάννης Χαραλάμπους

AM:1059685

Link για των κώδικα: https://github.com/cgiannos/1h_ergasia_vpologistikh.git

A1. Προεπεξεργασία και Προετοιμασία δεδομένων [15 μονάδες]

Προσοχή: Ό,τι μετασχηματισμοί εφαρμοστούν στα δεδομένα του συνόλου εκπαίδευσης, οι ίδιοι θα πρέπει να εφαρμοστούν και στα δεδομένα του συνόλου ελέγχου ή εναλλακτικά να αντιστραφούν πρώτου μετρηθούν οι μετρικές αξιολόγησης παρακάτω.

α) Οι τιμές των pixels της εικόνας, όπως αναφέρθηκε, είναι ακέραιες και κινούνται στο διάστημα $[0,255]$. Με δεδομένη την συγκεκριμένη αποτύπωση και την πιθανή ανάγκη προσαρμογής των τιμών αυτών σε διαφορετική κλίμακα (εξάλειψη ενδεχόμενης πόλωσης), παρουσιάζονται οι εξής τρεις μέθοδοι:

- **Κεντράρισμα (Centering):** Με την μέθοδο αυτή αφαιρούμε τον μέσο όρο των τιμών των pixel μιας εικόνας από όλες τις τιμές που έχουν ήδη αποδοθεί.
- **Κανονικοποίηση (Normalization):** Με την μέθοδο αυτή μεταφέρουμε το εύρος τιμών των pixel των δειγμάτων σε νέα κλίμακα πχ $[0,1]$.
- **Τυποποίηση (Standardization):** Με την μέθοδο αυτή παρέχουμε στο δείγμα ιδιότητες όπως μηδενική μέση τιμή και μοναδιαία διακύμανση (Gaussian).

Εξετάστε τη χρησιμότητα των ανωτέρω μεθόδων για το συγκεκριμένο πρόβλημα και εφαρμόστε τη/τις στα δεδομένα εκπαίδευσης, αν κρίνετε σκόπιμο. [10]

Αρχικά όσων αφορά την τυποποίηση έχει ως προϋπόθεση ότι τα δεδομένα που χρησιμοποιούμε έχουν gaussian κατανομή. Ακόμα και αν δεν έχουν πρέπει οι αλγόριθμοι που χρησιμοποιούμε για την επεξεργασία των δεδομένων να κάνουν υποθέσεις με gaussian κατανομές όπως η γραμμική παλινδρόμηση ή η λογιστική παλινδρόμηση.

Όσων αφορά το κεντράρισμα τα δεδομένα μας κινούνται σε μεγάλο διάστημα $[0,255]$ άρα δεν θα έχουμε καλή απόδοση.

Τέλος η κανονικοποίηση είναι μια καλή τεχνική όταν δεν γνωρίζετε την κατανομή των δεδομένων σας ή όταν γνωρίζετε ότι η διανομή δεν είναι Gaussian. Η κανονικοποίηση είναι χρήσιμη όταν τα δεδομένα έχουν διαφορετικές κλίμακες και ο αλγόριθμος που χρησιμοποιείτε δεν κάνει υποθέσεις σχετικά με τη διανομή των δεδομένων σας, στα τεχνητά νευρικά δίκτυα. Άρα θα χρησιμοποιήσω αυτή την μέθοδο.

```
# normalization

x_test = x_test / 255
x_train = x_train / 255
```

β) Διασταυρούμενη Επικύρωση (*cross-validation*): Βεβαιωθείτε ότι έχετε διαχωρίσει τα δεδομένα σας σε σύνολα εκπαίδευσης και ελέγχου, ώστε να χρησιμοποιήσετε 5-fold CV για όλα τα πειράματα. [5]

Για την υλοποίηση του CV ορίζουμε ένα αντικείμενο τύπου Kfold με τα κατάλληλα ορίσματα. Στη συνέχεια ορίζουμε μία for με ορίσματα τα ίδια τα δεδομένα διαχωρισμένα μέσω της συνάρτησης kfold.split. Μέσα στη for κάνουμε την εκπαίδευση και αξιολόγηση αυτών των διαχωρισμένων δεδομένων.

```
#kfold

kfold = KFold(5, shuffle=True, random_state=1)
accuracies, histories, losses = list(), list(), list()

for train_ix, test_ix in kfold.split(x_train, y_train):
    model = define_model()
    trainX, trainY, testX, testY = x_train[train_ix], y_train[train_ix], x_train[test_ix], y_train[test_ix]

    #es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
    history = model.fit(trainX, trainY, batch_size=32, epochs=10, verbose=0, validation_data=(testX, testY))
    loss, accuracy = model.evaluate(testX, testY, verbose=0)

    accuracies.append(accuracy)
    histories.append(history)
    losses.append(loss)
```

A2. Επιλογή αρχιτεκτονικής [55 μονάδες]

Όσον αφορά την τοπολογία των ΤΝΔ για την εκπαίδευση τους με τον Αλγόριθμο Οπισθοδιάδοσης του Σφάλματος (back-propagation), θα χρησιμοποιήσετε ΤΝΔ με ένα *κρυφό επίπεδο* και θα πειραματιστείτε με τον αριθμό των κρυφών κόμβων. Για την εκπαίδευση του δικτύου χρησιμοποιήστε αρχικά ρυθμό μάθησης $\eta = 0.001$.

α) Η εκπαίδευση και αξιολόγηση των μοντέλων σας μπορεί να γίνει με χρήση *Cross-Entropy* (CE), καθώς και *Μέσου Τετραγωνικού Σφάλματος* (MSE) για τις τιμές που περιέχονται στα σύνολα εκπαίδευσης και ελέγχου. Να εξηγήσετε με απλά λόγια ποια είναι η σημασία των παραπάνω μετρικών για το συγκεκριμένο πρόβλημα. [5]

Η CE μετρά την απόδοση ενός μοντέλου ταξινόμησης του οποίου η έξοδος είναι μια τιμή πιθανότητας μεταξύ 0 και 1 και προτιμάται για ταξινόμηση, ενώ το MSE είναι μία από τις καλύτερες επιλογές για παλινδρόμηση. Στην ταξινόμηση εργαζόμαστε με ένα πολύ συγκεκριμένο σύνολο πιθανών τιμών εξόδου, επομένως το MSE δεν είναι χρήσιμο. Όταν λοιπόν η συνάρτηση κόστους ακολουθεί κανονική κατανομή τότε προκύπτει και το MSE, ενώ αν ακολουθεί διωνυμική κατανομή προκύπτει το CE.

β) Πόσες εισόδους θα χρειαστείτε στο ΤΝΔ, δεδομένου ότι μια είσοδος πρέπει να αναπαριστά μια εικόνα 28X28; [5]

Θα πρέπει να χρησιμοποιήσουμε σαν εισόδους διανύσματα. Επομένως τις εικόνες που είναι 28×28 τις μετατρέπουμε σε διανύσματα 784 δεδομένων με αποτέλεσμα να χρειαζόμαστε 784 εισόδους.

γ) Πόσους νευρώνες θα χρειαστείτε στο επίπεδο εξόδου, δεδομένου του ζητούμενου της ταξινόμησης σε διαφορετικές κλάσεις; [5]

Η ταξινόμηση σε διαφορετικές τάξεις γίνεται με βάση των labels τα οποία έχουν τιμές [0-9]. Άρα το επίπεδο εξόδου χρειάζεται 10 νευρώνες για να κάνει αυτή την ταξινόμηση.

δ) Να επιλέξετε κατάλληλη συνάρτηση ενεργοποίησης για τους κρυφούς κόμβους και να τεκμηριώσετε την επιλογή σας. [5]

Από τις τρεις συναρτήσεις ενεργοποίησης για τους κρυφούς κόμβους (ReLU, Tanh, sigmoid), η Tanh και sigmoid ταιριάζουν στο νευρονικό μας δίκτυο. Εφόσον τα δεδομένα μας έχουμε κανονικοποιηθεί στο διάστημα [0,1] η sigmoid συνάρτηση η οποία λειτουργεί στο διάστημα [0,1] θα παράγει καλύτερα αποτελέσματα από την Tanh που λειτουργεί στο διάστημα [-1,1]. Άρα επιλέγω την sigmoid συνάρτηση ενεργοποίησης.

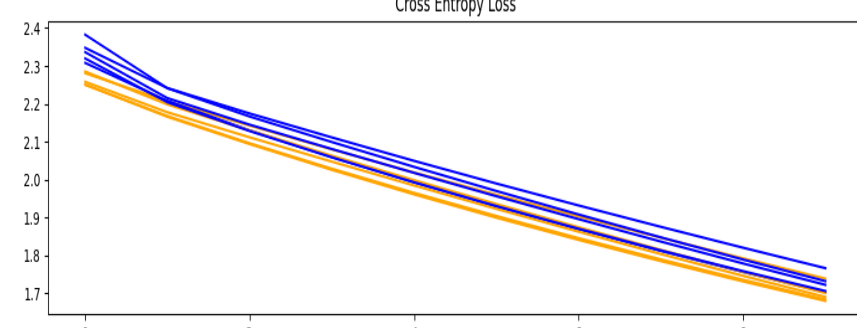
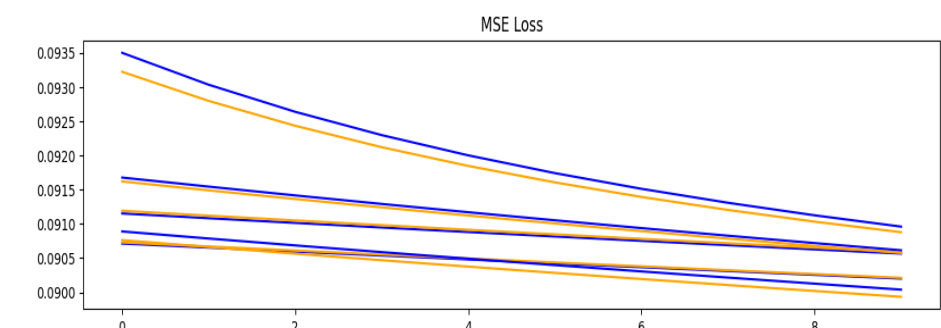
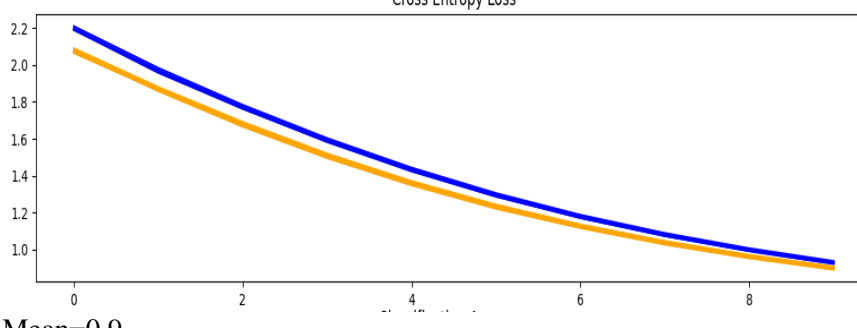
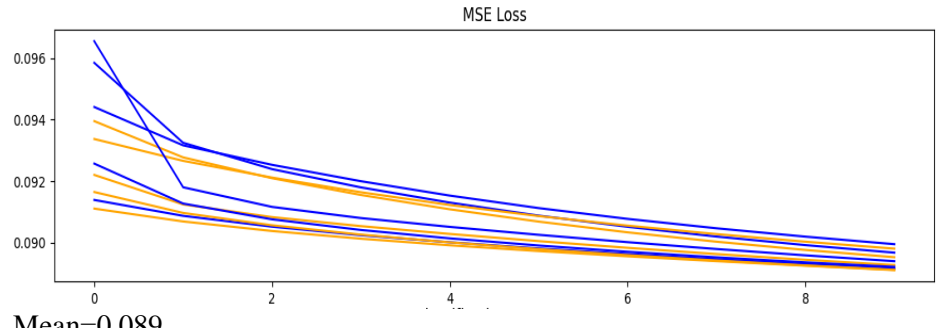
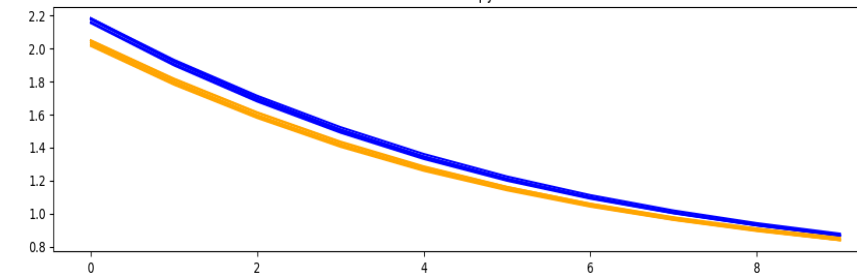
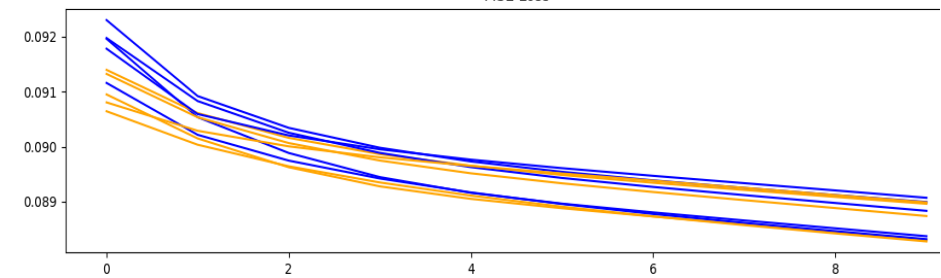
ε) Ποια συνάρτηση ενεργοποίησης θα χρησιμοποιήσετε για το επίπεδο εξόδου; Σιγμοειδή, Softmax ή κάποια άλλη; [5]

Στο επίπεδο εξόδου θα χρησιμοποιήσω την softmax συνάρτηση ενεργοποίησης. Αυτό το κάνω γιατί θέλω να κανονικοποιήσω τις εξόδους που βρίσκονται στο διάστημα [0,9], σε διάστημα [0,1] και θέλω το άθροισμα όλων των εξόδων να κάνει 1. Έτσι οι εξόδοι χρησιμοποιούνται ως πιθανότητες και η μεγαλύτερη πιθανότητα επιλέγεται ως η πρόβλεψη για το one-hot vector.

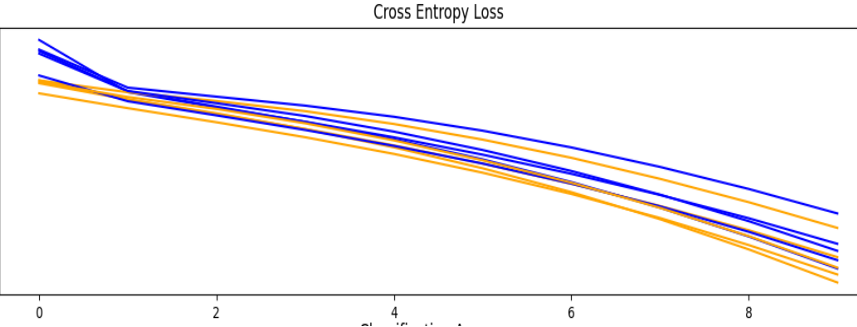
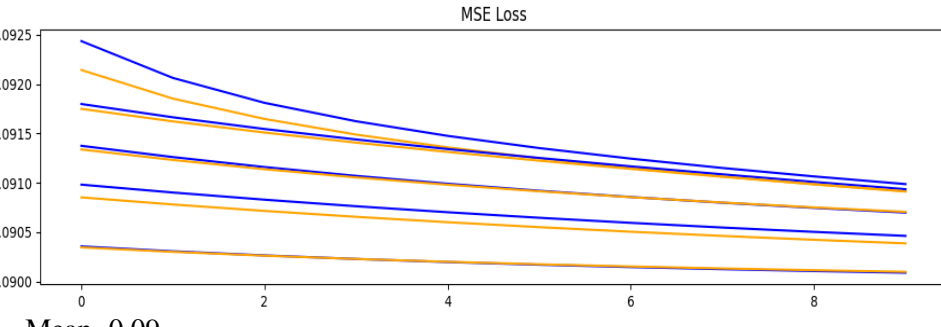
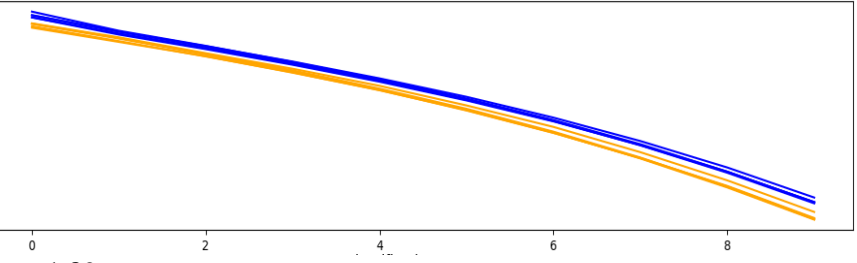
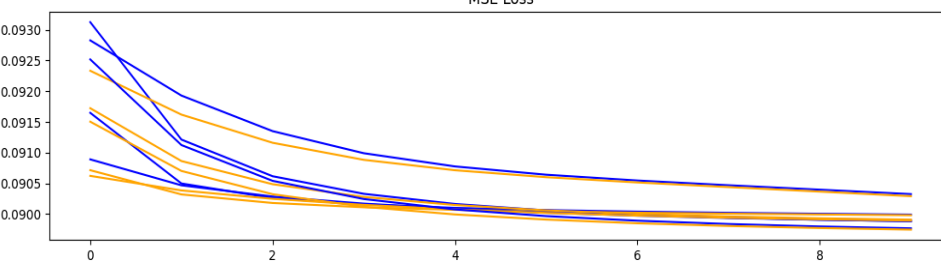
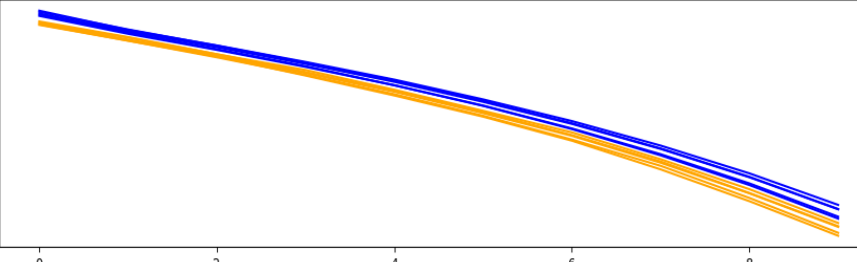
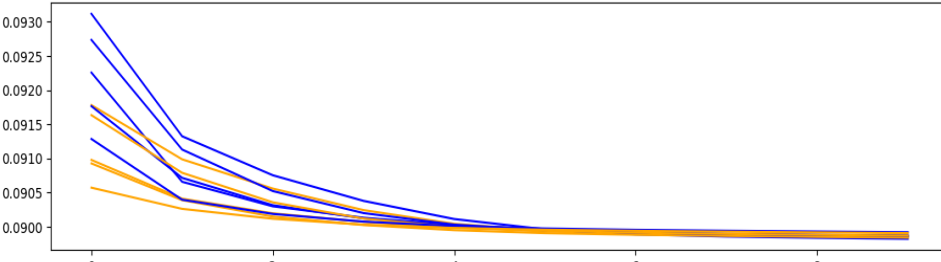
στ) Πειραματιστείτε με 3 διαφορετικές τιμές για τον αριθμό των νευρώνων του κρυφού επιπέδου και συμπληρώστε τον παρακάτω πίνακα. Εμπειρικά ενδεδειγμένες τιμές για τον αριθμό των κρυφών κόμβων βρίσκονται στο διάστημα $[O, I+O]$ (I αριθμός εισόδων, O αριθμός εξόδων, H αριθμός κόμβων στο κρυφό επίπεδο). Να συμπεριλάβετε και τις γραφικές παραστάσεις σύγκλισης (M.O.) ανά κύκλο εκπαίδευσης. Διατυπώστε τα συμπεράσματά σας σχετικά με (i) τον αριθμό των κρυφών κόμβων, (ii) την επιλογή της συνάρτησης κόστους και (iii) την ταχύτητα σύγκλισης ως προς τις εποχές εκπαίδευσης. [15]

Σύμφωνα με τα αποτελέσματα βλέπουμε πως όταν έχουμε 10 κόμβους η συνάρτηση λάθους έχει υψηλές τιμές και με τις δύο συναρτήσεις κόστους, άρα δεν είναι ιδανική επιλογή. Όσον αφορά τις συναρτήσεις για 397 και 794 κόμβους βλέπουμε πως οι συναρτήσεις έχουν παρόμοιες τιμές όμως και στην MSE και στην CE τα δεδομένα συγκλίνουν ως προς τις εποχές εκπαίδευσης πιο γρήγορα στους 794 κόμβους καθώς έχει και τους μικρότερους μέσους όρους η συνάρτηση λάθους κόμβους, άρα είναι και οι βέλτιστοι κόμβοι.

Η συνάρτηση με το πορτοκαλί χρώμα περιέχει τις τιμές της συνάρτησης κόστους από τα δεδομένα του cross-validation και η συνάρτηση με το μπλε χρώμα τις τιμές από τα δεδομένα που εκπαιδεύουμε.

Αριθμός νευρώνων στο κρυφό επίπεδο	CE loss	MSE
$H1=O=10$	<div></div> <div>Mean=1.698</div>	<div></div> <div>Mean=0.09</div>
$H1=(I+O)/2=397$	<div></div> <div>Mean=0.9</div>	<div></div> <div>Mean=0,089</div>
$H1=I+O=794$	<div></div> <div>Mean=0.84</div>	<div></div> <div>Mean=0.088</div>

ζ) Με τον βέλτιστο αριθμό κρυφών κόμβων που βρήκατε στο στ) δοκιμάστε να προσθέσετε ένα ακόμα κρυφό επίπεδο H₂ στο δίκτυο. Πειραματιστείτε με τον αριθμό των κόμβων του H₂. Περιγράψτε μια λογική για τη στοίχιση των κρυφών επιπέδων (είναι καλό να έχουν τον ίδιο αριθμό κόμβων; Μειούμενο; Αυξανόμενο;). Να αναφέρετε CE και MSE και να διατυπώσετε τα συμπεράσματα σχετικά με την προσθήκη κρυφών επιπέδων. [10]

Αριθμός νευρώνων στο κρυφό επίπεδο	CE loss	MSE
$H2=10$ $H1=794$	<div></div> <div>Mean=2.15</div>	<div></div> <div>Mean=0.09</div>
$H2=397$ $H1=794$	<div></div> <div>Mean=1.89</div>	<div></div> <div>Mean=0.0899</div>
$H2=794$ $H1=794$	<div></div> <div>Mean=1.91</div>	<div></div> <div>Mean=0.089</div>

Από τις γραφικές παραστάσεις βλέπουμε πως τα καλύτερα αποτελέσματα με δεύτερο κρυφό επίπεδο τα έχουμε όταν το δευτερο κρυφό επίπεδο έχει τους ίδιους κόμβους με το πρώτο επίπεδο.Επίσης συμπαιρénουμε πως όταν χρησιμοποιούμε δεύτερο κρυφό επίπεδο οι μέσοι όροι λάθους δεν είναι καλύτερη από όταν έχουμε ένα κρυφό επίπεδο.Παρόλο λοιπόν που κάνουμε πιο αποτελεσματικό το νευρωνικό δίκτυο τα αποτελέσματα δεν βελτιώνονται.Αυτό μπορεί να συμβαίνει λόγω έλλειψης πληροφορίας στο δίκτυο και καθώς το δίκτυο γίνεται πιο περίπλοκο τα δεδομένα που παράγει εμφανίζουν λάθοι, ενώ στο πιο απλό δίκτυο η έλλειψη πληροφορίες δεν εμφανίζεται.Έτσι το πιο αποτελεσματικό νευρωνικό δίκτυο μου ,έχει ένα κρυφό επίπεδο και 794 κόμβους.

η) Κριτήριο τερματισμού. Επιλέξτε και τεκμηριώστε κατάλληλο κριτήριο τερματισμού της εκπαίδευσης κάθε φορά (για κάθε fold). Μπορεί να χρησιμοποιηθεί η τεχνική του πρόωρου σταματήματος (early stopping); [5]

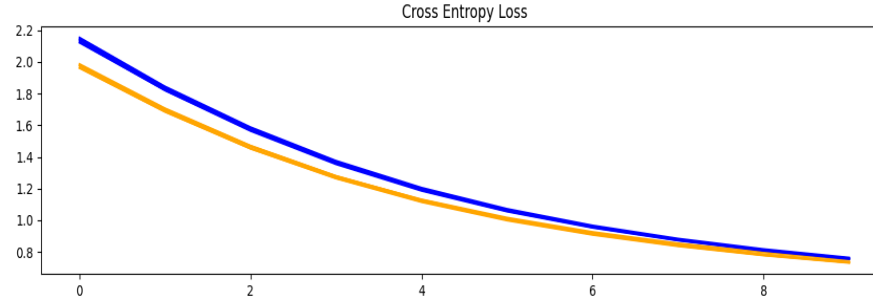
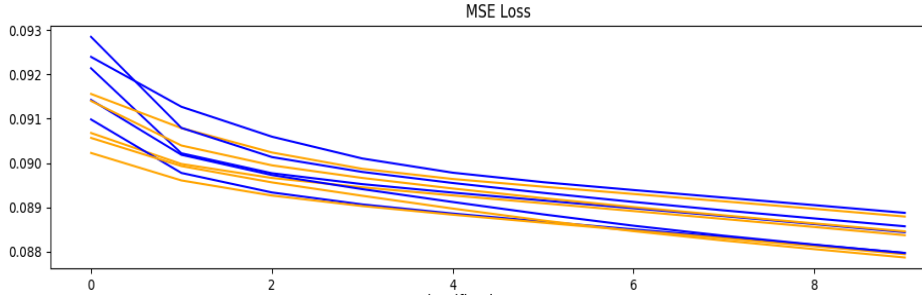
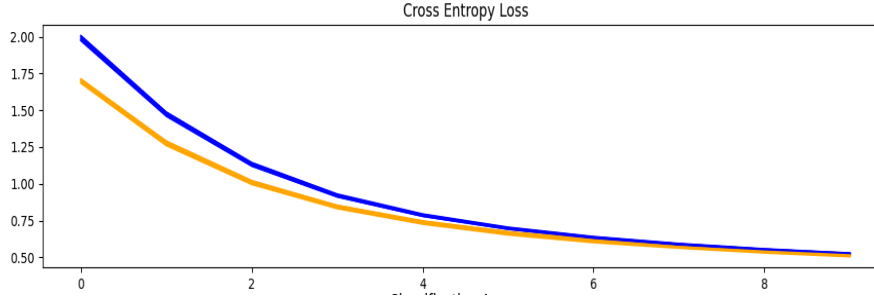
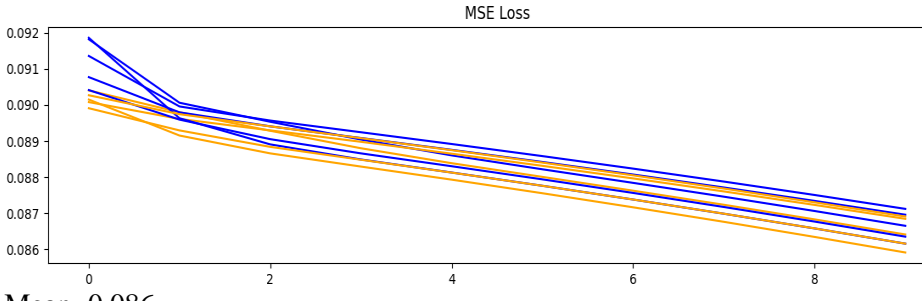
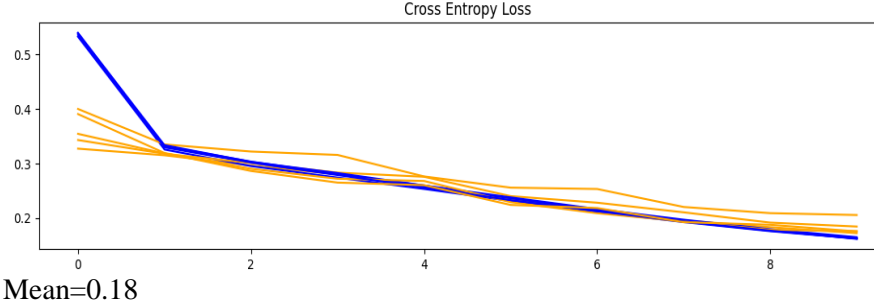
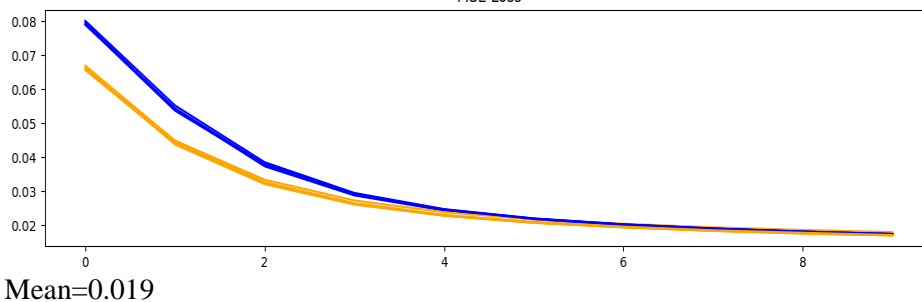
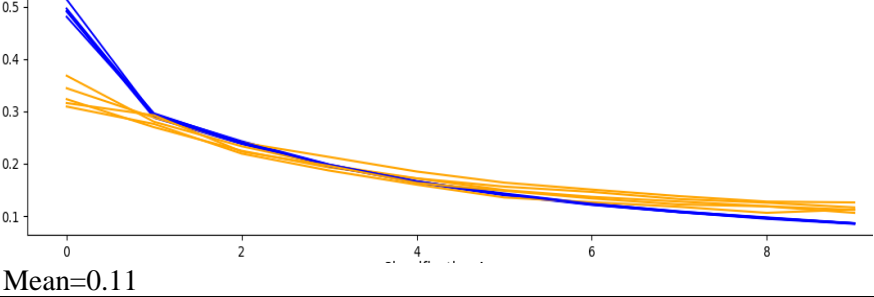
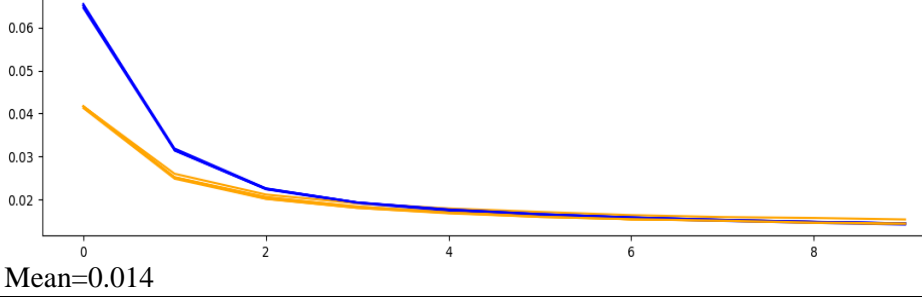
A3. Μεταβολές στον ρυθμό εκπαίδευσης και σταθεράς ορμής [15 μονάδες]
Επιλέγοντας την τοπολογία που δίνει το καλύτερο αποτέλεσμα βάσει του προηγούμενου ερωτήματος, να πραγματοποιήσετε βελτιστοποίηση των υπερπαραμέτρων ρυθμού εκπαίδευσης η και σταθεράς ορμής m με χρήση CV και να συμπληρώστε τον παρακάτω πίνακα. Να συμπεριλάβετε και τις γραφικές παραστάσεις σύγκλισης (M.O.) ως προς τους κύκλους εκπαίδευσης που θα χρειαστούν. Να τεκμηριώσετε θεωρητικά γιατί $m < 1$.

Χρησιμοποιώ την τοπολογία με ένα κρυφό επίπεδο και 794 κόμβους καθώς είχε το μικρότερο μέσο όρο σφάλματος.

Προκειμένου να βελτιστοποιήσουμε τις υπερπαραμέτρους πρέπει να προσθέσουμε παραμέτρους στον optimizer που επεξεργάζεται το

```
opt = keras.optimizers.SGD(learning_rate=0.1, momentum=0.6)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

νευρωνικό μας δίκτυο

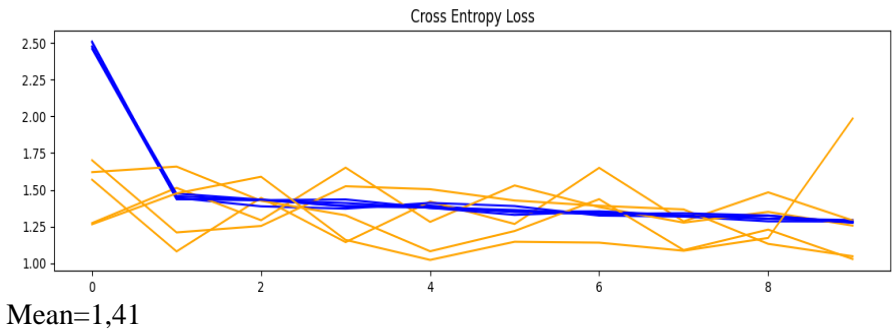
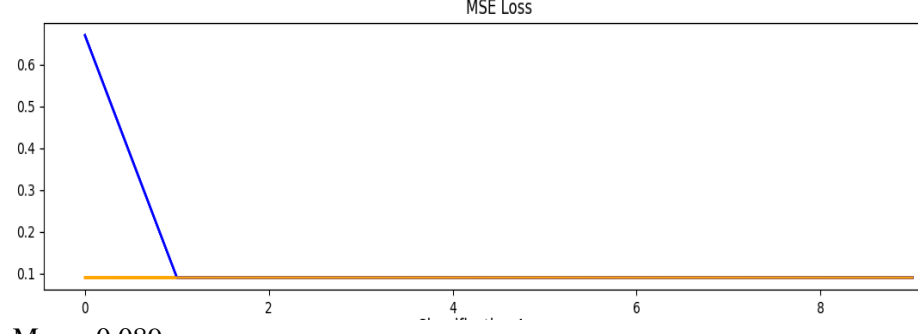
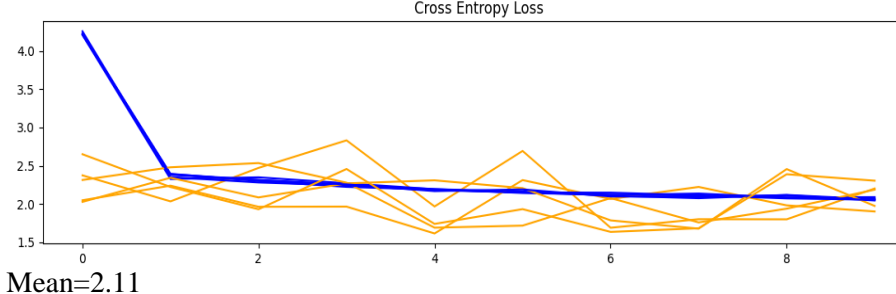
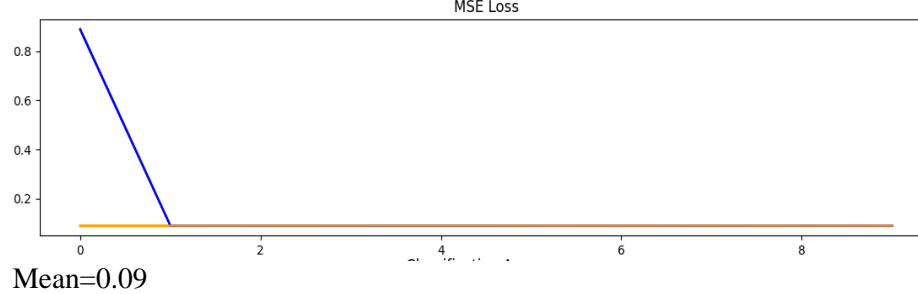
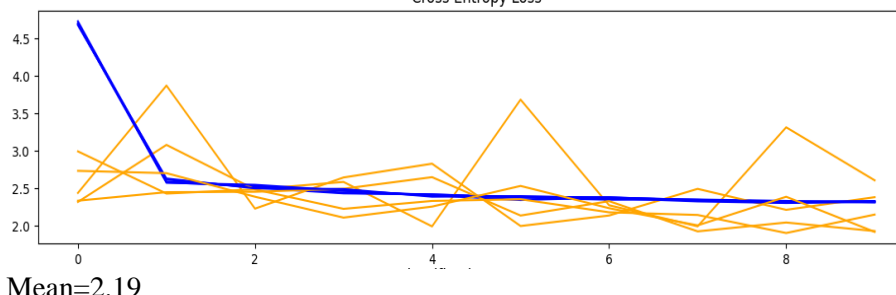
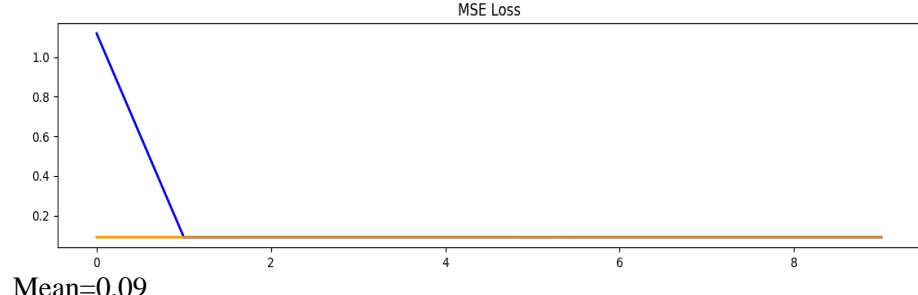
η	m	CE loss	MSE
0.001	0.2	<div></div> <div>Mean=0.73</div>	<div></div> <div>Mean=0.088</div>
0.001	0.6	<div></div> <div>Mean=0.51</div>	<div></div> <div>Mean=0.086</div>
0.05	0.6	<div></div> <div>Mean=0.18</div>	<div></div> <div>Mean=0.019</div>
0.1	0.6	<div></div> <div>Mean=0.11</div>	<div></div> <div>Mean=0.014</div>

Συμπαιρénοντας βλέπουμε ότι με την αύξηση των υπεπαραμέτρων οι τιμές της συνάρτηση λάθους μειώνονται .Συγκεκριμένα όταν έχουμε η=0.1 και m=0.6 ο μέσος όρος σφάλματος μειώνεται κατά 8 φορές από την βέλτιστη εκδοχή του δικτύου μας στο προηγούμενο ερώτημα.Άρα οι υπερπαραμέτροι βελτιστωποιούν το νευρωνικό μας δίκτυο.

Όσων αφορά τον περιορισμό του m $0 < m < 1$ το πρόβλημα λύνετε από την συνάρτηση του Gradient decent με την προσθήκη του momentum.
Έχουμε : $\Delta x(k) = x(k+1) - x(k) = -\alpha \frac{\partial F(x(k))}{\partial x(k)} + \beta \Delta x(k-1)$

Στη συνέχεια θεωρούμε πως το x έχει μία διάσταση και παίρνουμε τον μετασηματισμό Z και προκύπτει : $\Delta X(z) = -\alpha \partial F(x(k)) \partial x(k) + z^{-1} \beta \Delta X(z) \Rightarrow \Delta X(z) = \frac{-\alpha \partial F(x(k))}{1 - \beta z^{-1}}$

Όπου το $\frac{1}{1 - \beta z^{-1}}$ λειτουργεί ως χαμηλοπερατό φίλτρο και προκειμένου αυτό το φίλτρο να είναι σταθερό πρέπει $0 < \beta < 1$.

A4. Ομαλοποίηση [15 μονάδες]		
Μια μέθοδος για την αποφυγή υπερπροσαρμογής του δικτύου και βελτίωση της γενικευτικής του ικανότητας είναι η ομαλοποίηση του διανύσματος των βαρών (regularization). Να εφαρμόσετε <i>L2</i> ομαλοποίηση (φθορά βαρών) και να επανεκπαιδεύσετε το δίκτυό σας, όπως προέκυψε από το A3, αξιολογώντας διάφορες τιμές για τον συντελεστή φθοράς <i>r</i> . i) <i>r</i> = 0.1 ii) <i>r</i> = 0.5 iii) <i>r</i> = 0.9		
Συμπληρώστε τον παρακάτω πίνακα για κάθε μία από τις παραπάνω περιπτώσεις με χρήση 5-fold CV. Να συμπεριλάβετε και τις γραφικές παραστάσεις σύγκλισης (Μ.Ο.) ανά κύκλο εκπαίδευσης.		
Προκειμένου να υλοποιηθεί η ομαλοποίηση πρέπει να προσθέσουμε το <code>kernel_regularizer=l2(0.1)</code> στις παραμέτρους του κρύφου επιπέδου εισόδου .		
Συντελεστής φθοράς	CE loss	MSE
0.1	 Mean=1,41	 Mean=0.089
0.5	 Mean=2.11	 Mean=0.09
0.9	 Mean=2.19	 Mean=0.09

Η ομαλοποίηση χρησιμοποιείται για να αποφευχθεί η υπερπροσαρμογή στο δίκτυο και να περιοριστούν οι μεγάλες τιμές στα βάρη.Ως αποτέλεσμα χάνεται η ακρίβεια στις συναρτήσεις λάθους των δεδομένων του cross validation και ο μέσος όρος λάθους αυξάνεται.Καθώς λοιπόν το νευρωνικό δικτύό μας έκανε μια αρκετά λεπτομερή ταξινόμηση ,τώρα με την ομαλοποίηση των βαρών μας χάνεται αυτή η ακρίβεια του ενώ δεν είναι και απαραίτητη η αντιμετώπιση της υπερπροσαρμογής καθώς το νευρωνικό δίκτυο είναι αρκετά βελτιστοποιημένο.

A5. Convolutional Neural Network. [προαιρετικό ερώτημα - 10 μονάδες bonus]
Τα Convolutional Neural Networks (CNN) αποτελούν μοντέλα βαθιάς μάθησης που βρίσκουν εφαρμογή σε διάφορους τομείς, ενώ είναι ιδιαίτερα αποδοτικά σε εφαρμογές επεξεργασίας εικόνας. Σας ζητείται να πειραματιστείτε με διαφορετικές αρχιτεκτονικές CNN δικτύων (τουλάχιστον δύο), υλοποιώντας CNN μοντέλα που μετά από εκπαίδευση, να είναι ικανά να αναγνωρίζουν χειρόγραφα ψηφία. Για την υλοποίηση των επιπέδων του CNN μπορείτε να χρησιμοποιήσετε οποιαδήποτε από τις προτεινόμενες βιβλιοθήκες. Περιγράψτε αναλυτικά την αρχιτεκτονική δομή των μοντέλων και τις παραμέτρους εκπαίδευσης. Να αναφέρετε τιμές για CE και MSE, σχολιάστε τα αποτελέσματα και συγκρίνετέ τις υλοποιήσεις μεταξύ τους, αλλά και με την υλοποίηση των προηγούμενων ερωτημάτων.

Προκειμένου να υλοποιηθεί ένα CNN ,δεν χρειάζεται να αλλάξουμε πολλά στον κώδικα του NN .Αρχικά χρειάζεται να αλλάξουμε τον τρόπο που δημιουργούμαι το μοντέλο μας.

```
# NN
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    model.summary()

    opt = keras.optimizers.SGD(learning_rate=0.1, momentum=0.6)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

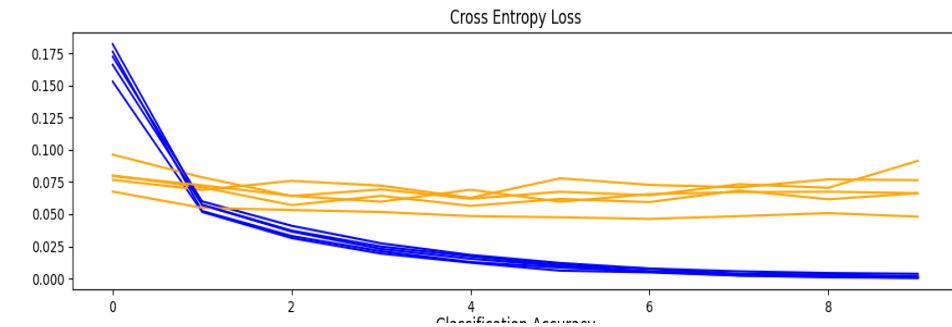
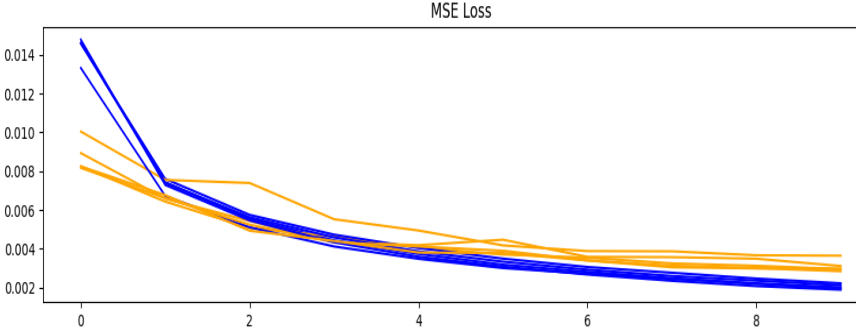
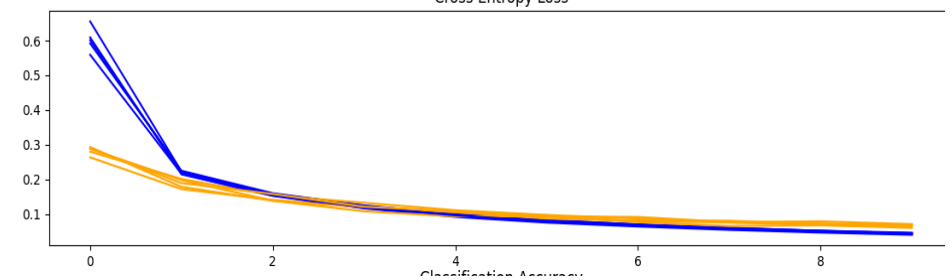
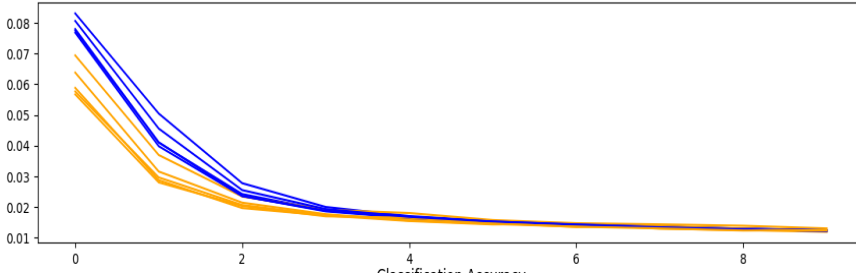

Στο μοντέλο του νευρωνικού μας δικτύου πλέον έχουμε τρία επιπλέον επίπεδα πριν το κρυφο επίπεδο εισόδου.Συγκεκριμένα το επίπεδο Conv2d Δημιουργούνται φίλτρα τα οποία κάνουν επεξεργασία των εικόνων, ενώ το MaxPooling2D μειώνει το μέγεθος των επεξεργαζμένων εικόνων .Τέλος το επίπεδο Flatten κάνει τα δεδομένα των εικόνων διανύσματα,ώστε να εισαχθούν στις εισόδους του κρυφού επιπέδου εισόδου.

Επίσης προκειμένου να εισαχθούν τα δεδομένα μας στο CNN πρέπει να περάσουν μία επιπλέον προεπεξεργασία καθώς οι συνάρτηση Conv2D του keras απαιτούν τα δεδομένα να έχουν μόνο gray κλίμακα.

```
# reshape dataset to have a single channel
x_train = x_train.values.reshape((x_train.shape[0], 28, 28, 1))
x_test = x_test.values.reshape((x_test.shape[0], 28, 28, 1))
```

Τα νέα δεδομένα μας είναι ήδη τύπου numpy οπότε δεν χρειάζεται να προστεθεί όπως στα NN.

Ως αρχιτεκτονική του CNN χρησιμοποιώ ένα δίκτυο εισόδου με 100 κόμβους και ένα εξόδου με 10 κόμβους .Όλα τα επίδεδα έχουν ως υπερπαραμέτρους η=0.1 και m=0.6.Το επίπεδο εξόδου χρησιμοποιεί την softmax συνάρτηση ενεργοποίησης ,ενώ τα υπόλοιπα επίπεδα την συνάρτηση ενεργοποίηση RELU.Στην εναλλακτική αρχιτεκτονική μου χρησιμοποιώ ως συνάρτηση ενεργοποίηση όλων των επιπέδων την sigmoid εκτός από το επίπεδο εξόδου που χρησιμοποιώ την softmax καθώς είναι η βέλτιστη.

αρχιτεκτονική relu	<div>CE LOSS</div> <div><p>Mean=0.069</p></div>	<div>MSE LOSS</div> <div><p>Mean=0.003</p></div>
sigmoid	<div>CE LOSS</div> <div><p>Mean=0.066</p></div>	<div>MSE LOSS</div> <div><p>Mean=0.012</p></div>

Συμπαιρένοντας από τις δύο αρχιτεκτονικές βλέπουμε πως με την RELU οι τιμές σφάλματος είναι μικρότερές από τις τιμές της sigmoid παρολαυτά η ταχύτητα σύγκλισης δεν είναι καλύτερη από την sigmoid με αποτέλεσμα ο Μ.Ο. σφάλματος είναι σχεδόν ίδιος.Έτσι βλέπουμε πώς το CNN με RELU συναρτήσεις ενεργοποίησης έχει μικρότερες τιμές λάθους ,ενώ με sigmoid έχει πιο γρήγορη ταχύτητα σύγκλισης των δύο εποχών.

Όσων αφορά την διαφορά ανάμεσα στη CNN και NN υλοποίησης του δικτύου μας βλέπουμε πως στο CNN το σφάλμα μειώνεται αρκετά,παρόλο που το ίδιο το δίκτυο περιέχει λίγους κόμβους .Όμως το CNN είναι πολύ χρονοβόρο στην επεξεργασία του, ειδικά αν προσθέσουμε και παραπάνω κόμβους γιατί και το απλό νευρωνικό δίκτυο είναι πιο εύχρηστο.

