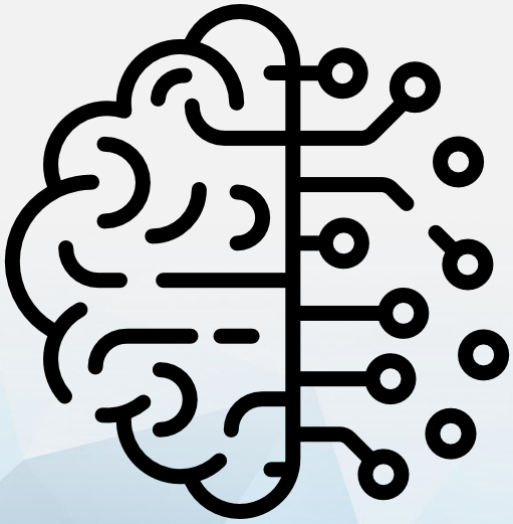
Υπολογιστική Νοημοσύνη

Εργαστηριακή Άσκηση

2023

Μέρος Α



Στυλιανός Στυλιανάκης

1059713

Link to code

Α1. Προεπεξεργασία και Προετοιμασία δεδομένων

a) Κωδικοποίηση και προεπεξεργασία δεδομένων

Για την ανάγνωση και την κωδικοποίηση των αρχείων εισόδων είναι υπεύθυνο το αρχείο **data_prep.py**. Συγκεκριμένα, στη συνάρτηση `preprocessDataset` διαβάζεται το σύνολο δεδομένων μας, φορτώνεται σε ένα dataframe και στη συνέχεια ξεχωρίζονται οι αριθμητικές και οι κατηγορηματικές στήλες.

```
def preprocessDataset() -> pd.DataFrame:
  # Read dataset file and load it to a dataframe
  original_df = pd.read_csv(files_folder / "dataset-HAR-PUC-Rio.csv",
                            delimiter=';', low_memory=False, decimal=',')

  # Extract numerical columns
  numerical_columns = original_df.select_dtypes(include="number")

  # Normalize (min max scale) numerical values
  normalized_values = StandardScaler().fit_transform(numerical_columns.values)
  normalized_df = pd.DataFrame(columns=numerical_columns.columns,
                               data=normalized_values)

  # One-hot encode categorical features
  encoder = ce.OneHotEncoder(handle_unknown='return_nan', return_df=True,
                             use_cat_names=True)
  encoded_cat_features = encoder.fit_transform(
      original_df[['user', 'class', 'gender']])

  # Combine the 2 dataframes
  final_df = pd.concat([normalized_df, encoded_cat_features], axis=1)

  # Reorder columns
  cols = final_df.columns.to_list()
  final_df.columns = cols[:-7] + ["gender_man", "gender_woman"] + cols[-7:-2]

  # Save processed dataset to file
  final_df.to_csv(files_folder / "Processed dataset.csv", index=False)

  return final_df
```

Επειδή εύρος, καθώς και η κλίμακα των τιμών διαφέρουν σημαντικά ανά χαρακτηριστικό, τα αριθμητικά δεδομένα θα πρέπει να περάσουν κάποιου

είδους αναπροσαρμογή, ώστε οι τιμές με μεγαλύτερα εύρη (αλλά και γενικότερα τιμές) να επηρεάζουν το ίδιο το μοντέλο, ανάλογα και με τα βάρη που τους ορίζονται. Οι 3 επιλογές αναπροσαρμογής δεδομένων είναι οι εξής:

- **Κεντράρισμα** (Centering), η αφαίρεση δηλαδή της μέσης τιμής από όλες τις τιμές ενός χαρακτηριστικού. Με το κεντράρισμα δε μειώνεται το εύρος (έστω R), απλώς μετακινούνται οι τιμές ώστε να βρίσκονται στο [-R/2, R]. Έτσι, ιδιότητες με μεγαλύτερα εύρη θα επηρεάζουν περισσότερο το μοντέλο. Αυτό καθιστά τη μέθοδο του κεντραρίσματος ακατάλληλη για τις ιδιότητές μας.
- **Κανονικοποίηση** (Normalization ή Min-Max scale). Η κανονικοποίηση μεταφέρει όλες τις τιμές των ιδιοτήτων στο διάστημα [0, 1] διαιρώντας τις αρχικές τους τιμές με τη μέγιστη τιμή του εκάστοτε χαρακτηριστικού. Είναι μία καλή υποψήφια μέθοδος, όμως λόγω του ότι έχουμε κάποιες ακραίες τιμές (outliers) στα δεδομένα μας, δεν είναι ιδανική μέθοδος.
- **Τυποποίηση** (Standardization ή z-score). Η τυποποίηση είναι το κεντράρισμα ακολουθούμενο από τη διαίρεση τους με την τυπική απόκλιση. Η τυποποίηση επηρεάζεται λιγότερο από τις ακραίες τιμές, που είναι και ο λόγος που επιλέχθηκε για την εργασία μας αντί για την κανονικοποίηση.

Τα κατηγορικά δεδομένα γίνονται **One-Hot Encode**, για παράδειγμα οι στάσεις κωδικοποιούνται ως 5 δυαδικά χαρακτηριστικά τα οποία είναι 0 ή 1 ανάλογα με την κλάση στην οποία βρίσκεται ο χρήστης.

b) Διασταυρωμένη Επικύρωση

Με τη γραμμή `kfold = KFold(n_splits=5, shuffle=True)` χωρίζω τα δεδομένα μου σε 5 σύνολα εκπαίδευσης ώστε να τα χρησιμοποιήσω αργότερα για το 5-Fold Cross Validation. Ανακατεύοντας τα δεδομένα επιτυγχάνεται και η ισορροπία στην αναλογία κλάσεων σε κάθε fold.

```
# Define k-fold
cv = KFold(n_splits=5, shuffle=True)
```

A2. Επιλογή αρχιτεκτονικής

a) Εκπαίδευση και αξιολόγηση μοντέλου

Για την αξιολόγηση του νευρωνικού μου δικτύου χρησιμοποίησα Cross-Entropy και πιο συγκεκριμένα **Categorical Cross Entropy** ως συνάρτηση κόστους, καθώς κάθε είσοδος ανήκει ακριβώς σε ένα label, άρα ο αντίστοιχος νευρώνας θα πρέπει να ενεργοποιείται και οι υπόλοιποι να παραμένουν ανενεργοί. Αυτό γίνεται κατά το compilation του μοντέλου, όπως φαίνεται στο παρακάτω snippet κώδικα στην πρώτη γραμμή.

Η συνάρτηση *Cross Entropy*, αλλιώς γνωστή και ως **Log Loss** αθροίζει για κάθε πρόβλεψη το $\log_e$(predicted_value) των labels που είναι 1 μαζί με το $\log_e$(1-predicted_value) των labels που είναι 0 για κάθε κατηγορία.

Η συνάρτηση Μέσου Τετραγωνικού Σφάλματος (MSE) αθροίζει την τετραγωνισμένη διαφορά της predicted value από την πραγματική τιμή του label. Για παράδειγμα, αν ένα κείμενο ανήκει σε μία κατηγορία και το νευρωνικό έβγαλε την τιμή 0.7 στον αντίστοιχο νευρώνα, τότε το MSE για αυτόν τον νευρώνα θα ήταν $(1-0.7)^2 = 0.09$.

Η συνάρτηση Accuracy ελέγχει πόσο συχνά η προβλεπόμενη τιμή από το νευρωνικό ισούται με την επιθυμητή τιμή (label).

b) Νευρώνες εξόδου και Classification

Εφόσον οι κατηγορίες στάσεων είναι 5 θα έχουμε και 5 νευρώνες εξόδου.

Επειδή κάθε είσοδος ανήκει σε ακριβώς **1** κατηγορία από τις 5, το πρόβλημά μας ανήκει στα προβλήματα τύπου multiclass classification.

c) Συνάρτηση ενεργοποίησης για τους κρυφούς κόμβους

Ως συνάρτηση ενεργοποίησης των κρυφών κόμβων επιλέχθηκε η **relu**, αφού είναι πιο εύκολη στην εκπαίδευση και είναι η πιο ευρέως χρησιμοποιημένη για κρυφούς κόμβους σε σύγχρονα νευρωνικά δίκτυα, ενώ παρήγαγε και τα καλύτερα αποτελέσματα σε κάποια σύντομα πειράματα μου.

d) Συνάρτηση ενεργοποίησης εξόδου

Για την έξοδο επιλέχθηκε η **softmax** ως συνάρτηση ενεργοποίησης, επειδή το πρόβλημα μας είναι Muliclass Classification, που σημαίνει ότι μόνο ένας από τους 5 νευρώνες εξόδου θέλουμε να ενεργοποιείται κάθε φορά. Στην πραγματικότητα, χρησιμοποιώντας τη συνάρτηση softmax, το μοντέλο αποδίδει την πιθανότητα η είσοδος να ανήκει σε κάθε κλάση. Έτσι, οι 5 έξοδοι μας θα έχουν τιμές στο διάστημα [0, 1], ενώ το άθροισμά τους θα ισούται με 1.

e) Αριθμός νευρώνων κρυφού επιπέδου

Αριθμός νευρώνων στο κρυφό επίπεδο	CE Loss	MSE	Acc
$H_1 = O$	0.4172	0.0427	0.8548
$H_1 = (I+O) / 2$	0.2710	0.0263	0.9159
$H_1 = I + O$	0.2310	0.0218	0.9326

Αυξάνοντας τον αριθμό των νευρώνων του κρυφού επιπέδου, το μοντέλο βελτιώνεται.

f) Κριτήριο Τερματισμού

Κατά την επιλογή εποχών δεν μπορούμε να ξέρουμε εξ αρχής ποιος είναι ένας καλός αριθμός εποχών ώστε το μοντέλο μας να μην κάνει over ή underfit. Για αυτόν τον σκοπό, χρησιμοποιούμε τη μέθοδο **“early stopping”**, με την οποία μπορούμε να σταματήσουμε την εκπαίδευση ενός fold πρόωρα, όταν δούμε ότι έχει **σταματήσει** πλέον να βελτιώνεται. Έτσι, στην περίπτωση της εργασίας μου επέλεξα να χρησιμοποιήσω early stopping όταν σταματήσει να βελτιώνεται το validation loss ώστε να αποφευχθεί το **overfit**. Βέβαια, λόγο περιορισμένου χρόνου και υπολογιστικών πόρων, το μοντέλο δεν προλαβαίνει να κάνει overfit συνήθως, όμως με το early stopping μειώνεται ο χρόνος εκπαίδευσης. Η συνάρτηση για τον πρόωρο τερματισμό που επέλεξα φαίνεται στο παρακάτω κομμάτι κώδικα.

```
  early_stopping = EarlyStopping(monitor="val_accuracy", patience=10,
                                 mode="max", min_delta=0.001,
                                 restore_best_weights=True)
```

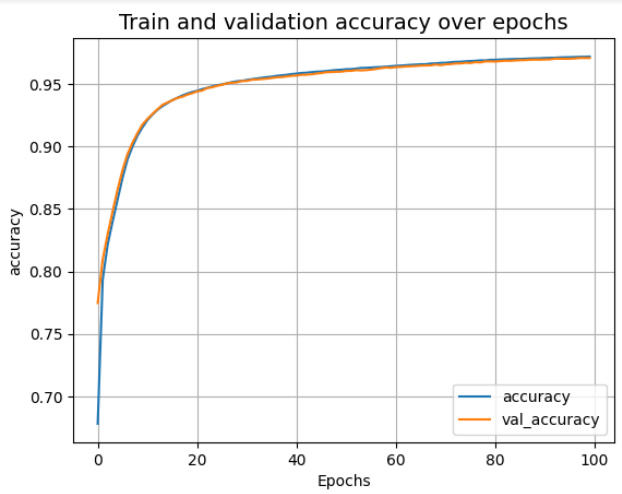
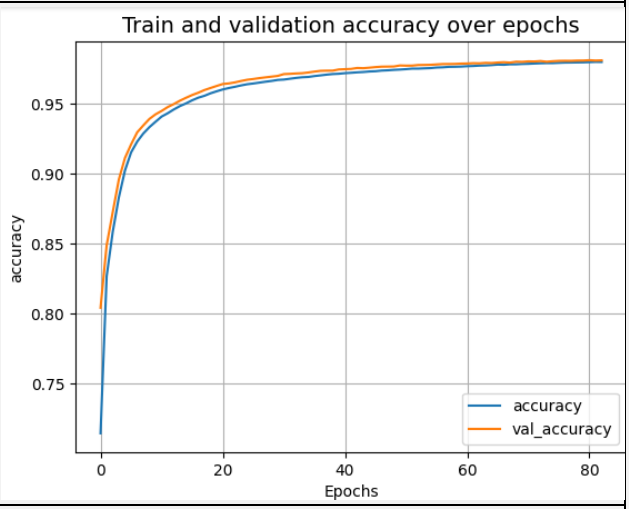
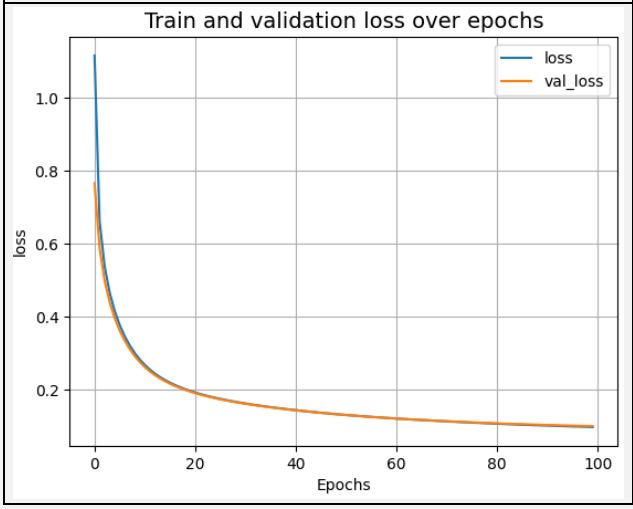
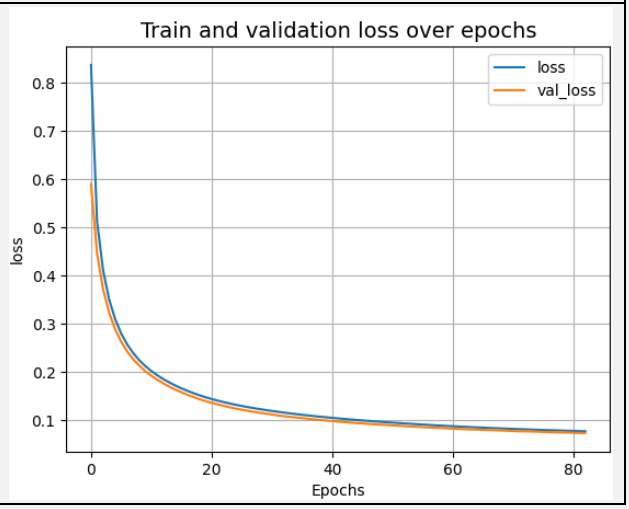
Λαμβάνοντας όλα τα παραπάνω υπόψιν, δημιουργούμε το μοντέλο μας όπως φαίνεται στο παρακάτω κομμάτι κώδικα:

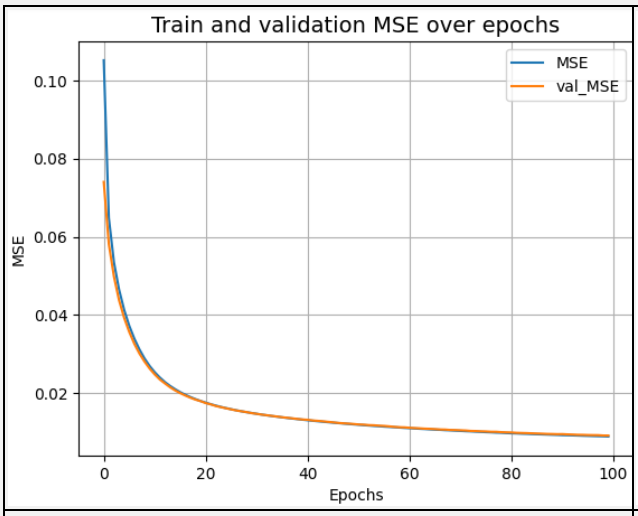
```
def getModel() -> Sequential:
  """Defines the model and returns it."""

  model = Sequential()
  model.add(Dense(NW_IN + NW_OUT, input_dim=NW_IN, activation='relu'))
  model.add(Dense(NW_OUT, activation='softmax'))
  model.compile(loss='categorical_crossentropy',
                optimizer=gradient_descent_v2.SGD(learning_rate=H,
                                                  momentum=M), metrics=metrics)
  return model
```

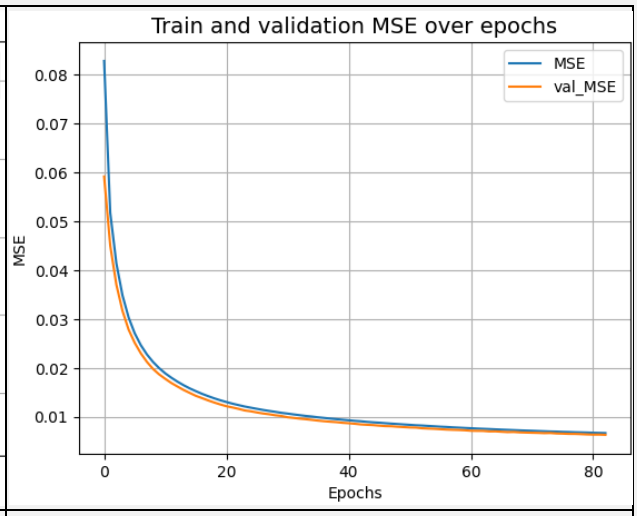
A3. Μεταβολές στο ρυθμό εκπαίδευσης και στη σταθερά ορμής

Η ορμή είναι μία μέθοδος με την οποία επιταχύνεται το Stochastic Gradient Descent ώστε να συγκλίνει πιο γρήγορα σε κάποιο τοπικό ελάχιστο του Loss Function. Η παράμετρος m ορίζει πόσο επηρεάζεται ο αλγόριθμος από την προηγούμενή του κίνηση. Όταν έχει την τιμή 0, ο αλγόριθμός μας δεν επηρεάζεται από την προηγούμενή του κίνηση και είναι απλή SGD. Για πολύ μεγάλες τιμές επηρεάζουν υπερβολικά την κίνησή του και γίνεται επιρρεπής σε μεγάλα σφάλματα.

H	m	CE	MSE	Acc
0.001	0.2	0.1030	0.0092	0.9712
0.001	0.6	0.0810	0.0072	0.9781
0.05	0.6	0.0502	0.0042	0.9869
0.1	0.6	0.0533	0.0043	0.9865

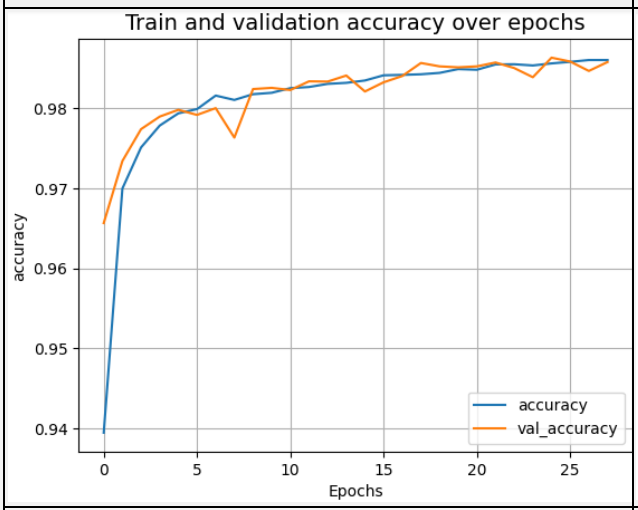
H = 0.001, m = 0.2	H = 0.001, m = 0.6
	
	

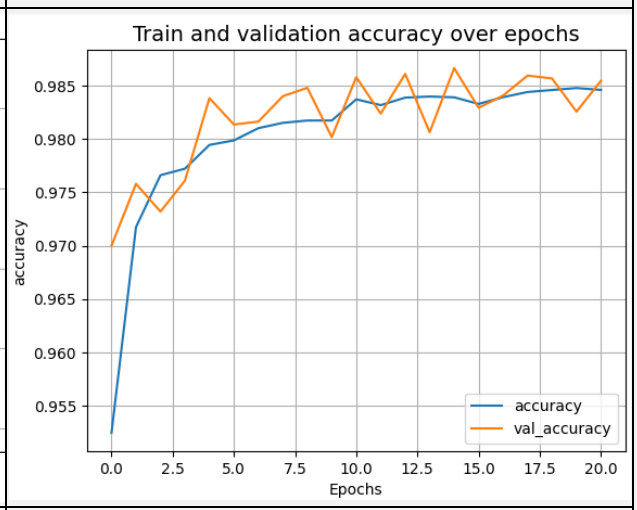


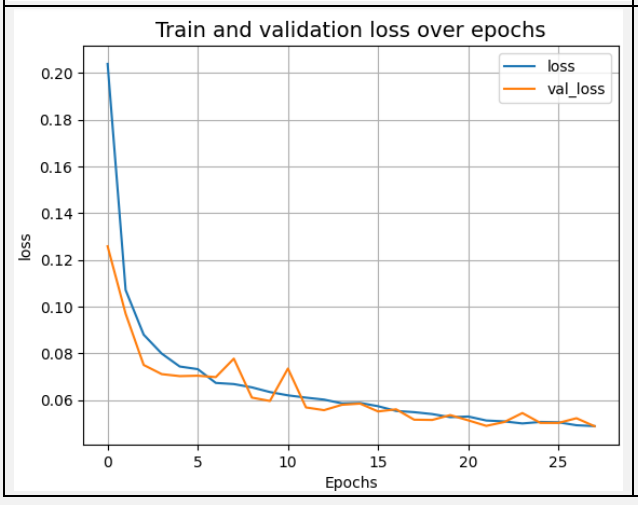


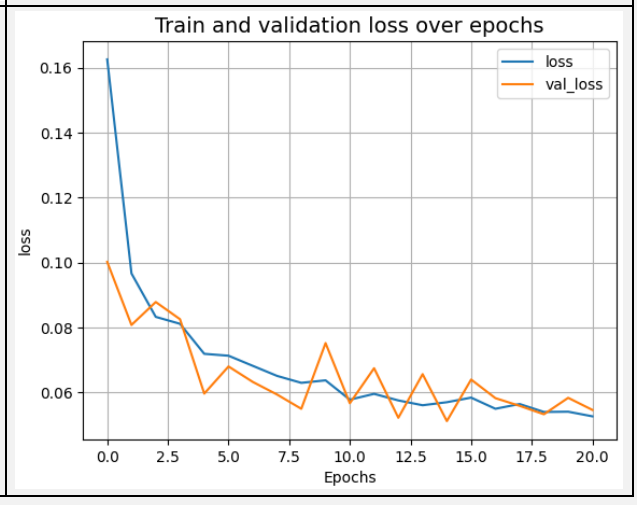
H = 0.05, m = 0.6

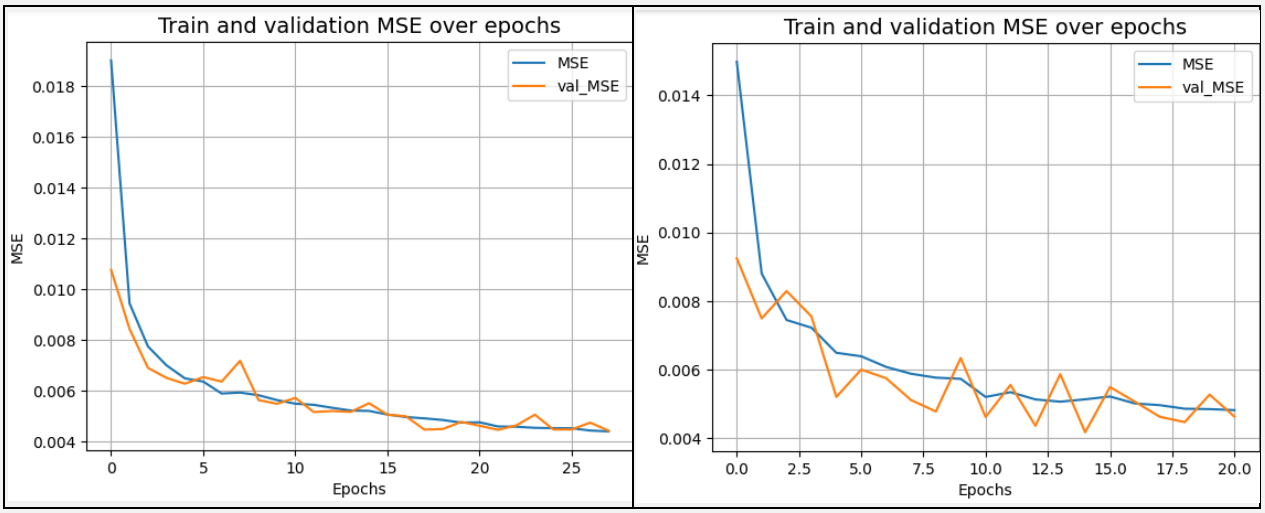
H = 0.1, m = 0.6









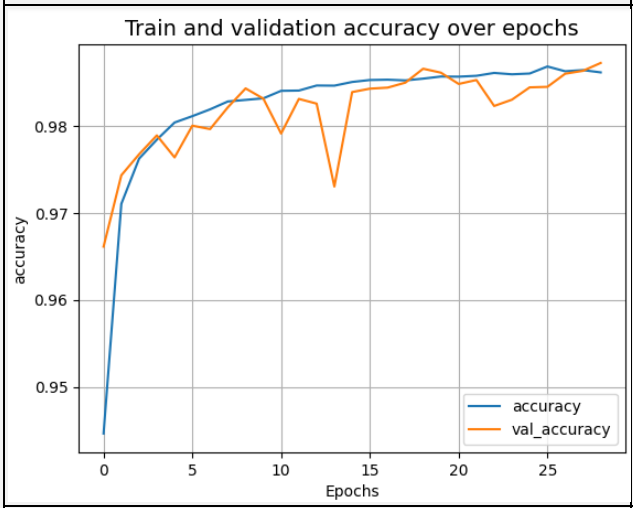
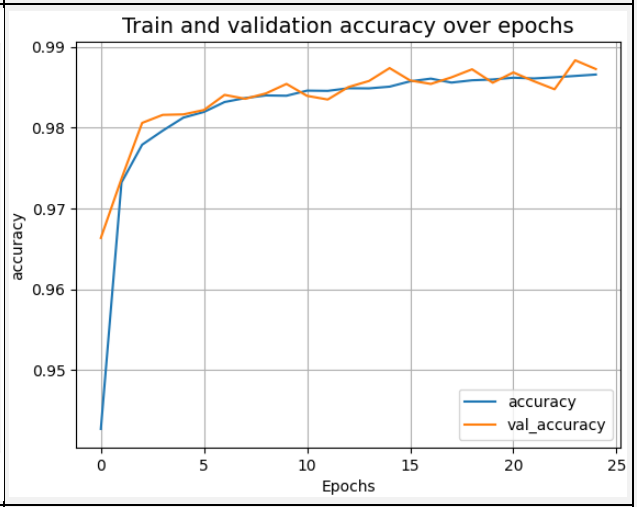
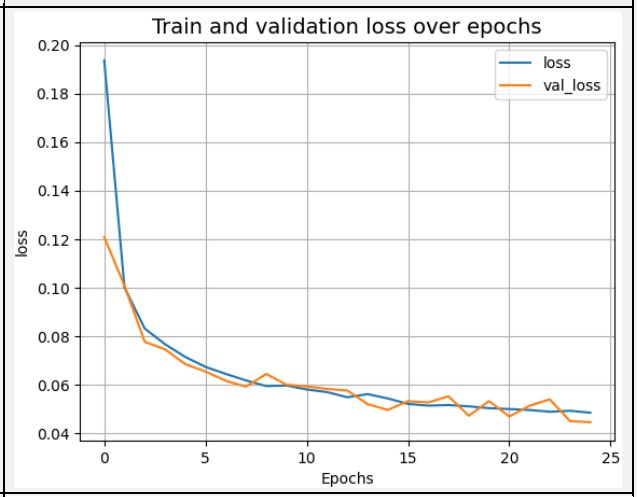
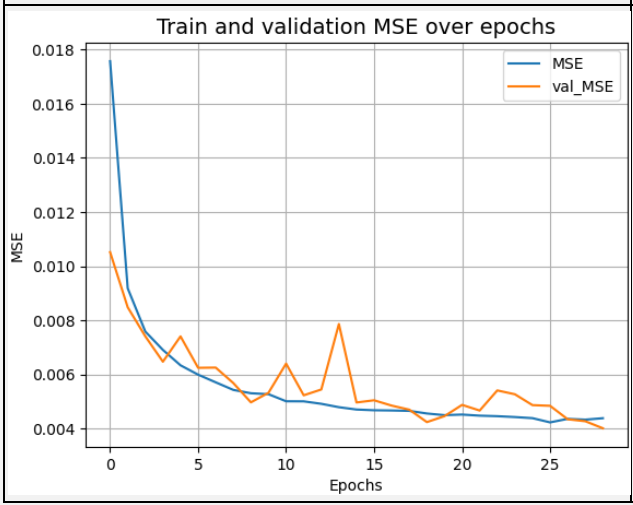
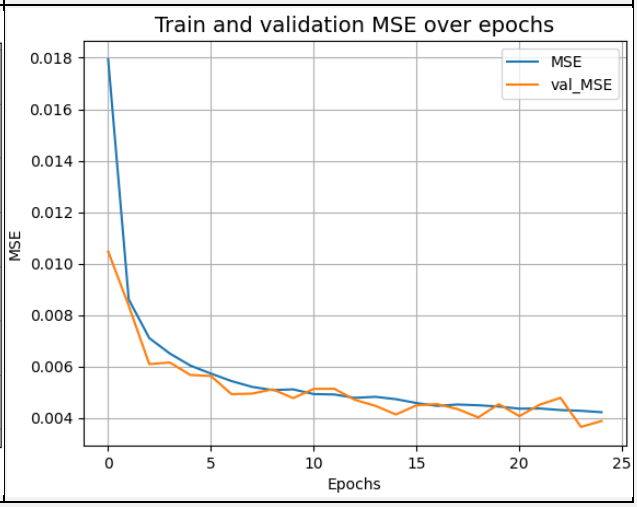


Παρατηρούμε ότι με ρυθμό μάθησης 0.001 αυξάνοντας την ορμή από 0.2 σε 0.6, αλλά και αυξάνοντας το ρυθμό μάθησης μέχρι και το 0.05, το μοντέλο συγκλίνει νωρίτερα κι έχει καλύτερες μετρικές. Κατά την περαιτέρω όμως αύξηση του ρυθμού μάθησης σε 0.1, το μοντέλο μπορεί να συγκλίνει λίγο νωρίτερα, όμως η εκπαίδευση είναι αρκετά ασταθής ενώ η οι μετρικές σταματάνε να βελτιώνονται.

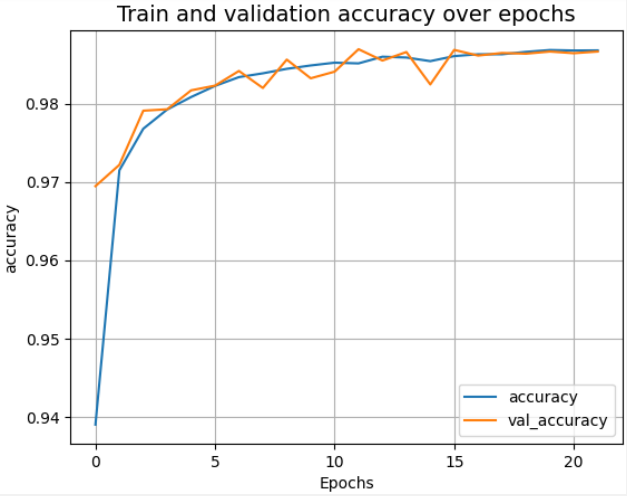
A4. Ομαλοποίηση

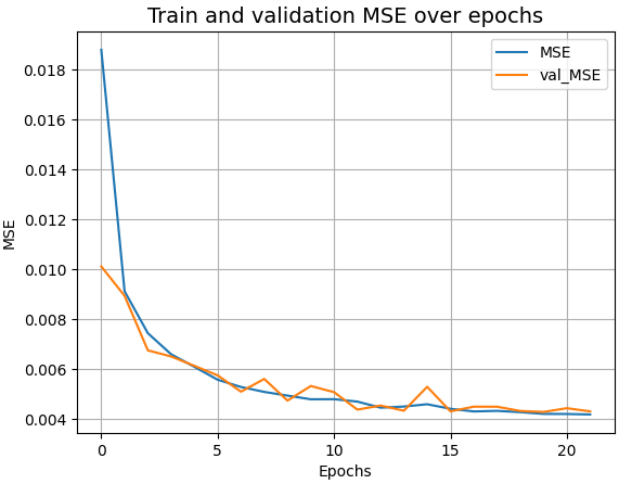
Με τη χρήση της ομαλοποίησης, το δίκτυό μας οδηγείται στην προτίμηση μικρότερων βαρών, αποφεύγοντας έτσι το overfitting. Στην υλοποίησή μου δε φαίνεται να υπάρχει το φαινόμενο αυτό, παρόλα αυτά παρακάτω φαίνεται η εκπαίδευση του μοντέλου με παραμέτρους H = 0.05 και m = 0.06 (όπως προέκυψαν από το προηγούμενο ερώτημα) χρησιμοποιώντας κανονικοποίηση L2. Στην πραγματικότητα, υπάρχει μία ιδιαιτερότητα στο dataset ότι οι τιμές των ιδιοτήτων επαναλαμβάνονται με μικρές διαφορές όσο το άτομο βρίσκεται σε μία στάση, οδηγώντας έτσι στην απουσία πραγματικού “validation set”. Έτσι, ενώ σε ένα διαφορετικό dataset (ίσως και μία διαφορετική υλοποίηση) θα υπήρχε προφανές overfitting για ρυθμό μάθησης >= 0.05, στην παρούσα υλοποίηση δε φαίνεται τόσο χαρακτηριστικά το παραπάνω φαινόμενο.

Συντελεστής φθοράς	CE loss	MSE	Acc
0.1	0.0506	0.0043	0.9864
0.5	0.0504	0.0420	0.9869
0.9	0.0524	0.0043	0.9864

R=0.1	R=0.5
 Train and validation accuracy over epochs accuracy Epochs accuracy val_accuracy	 Train and validation accuracy over epochs accuracy Epochs accuracy val_accuracy
 Train and validation loss over epochs loss Epochs loss val_loss	 Train and validation loss over epochs loss Epochs loss val_loss
 Train and validation MSE over epochs MSE Epochs MSE val_MSE	 Train and validation MSE over epochs MSE Epochs MSE val_MSE

R = 0.9



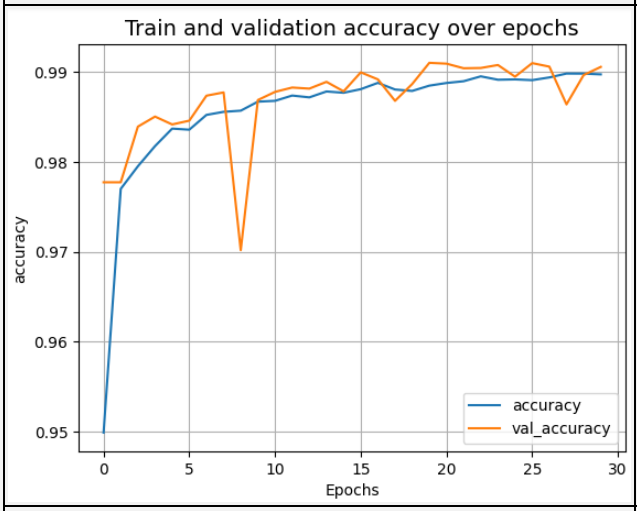
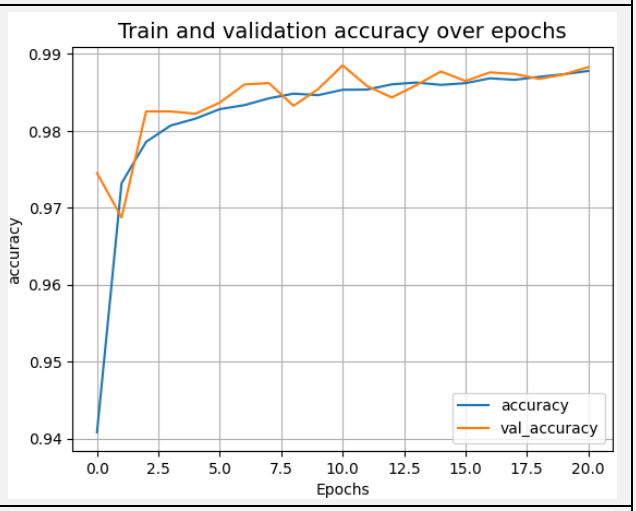
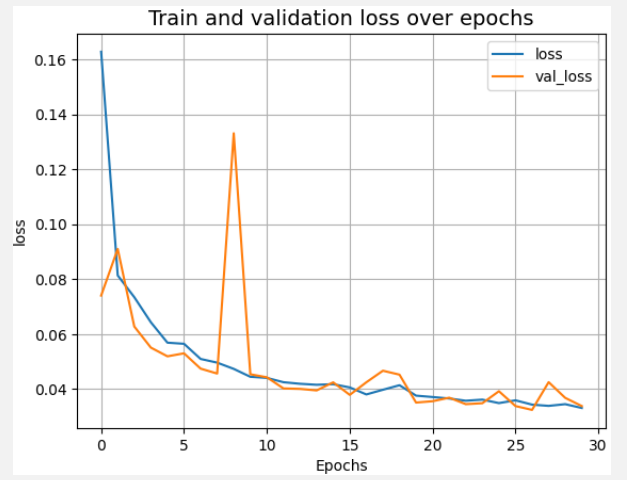
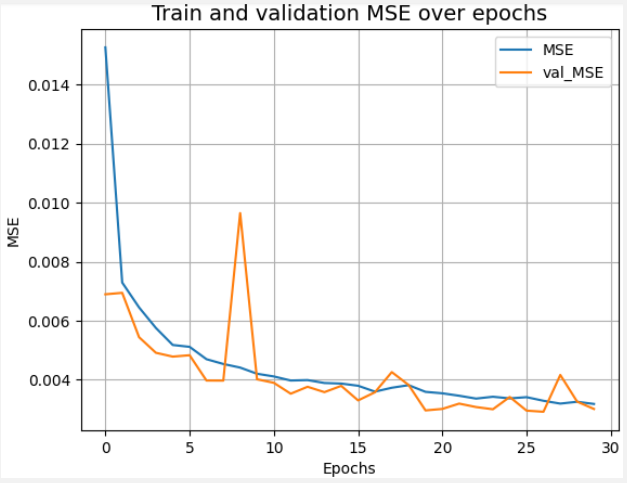
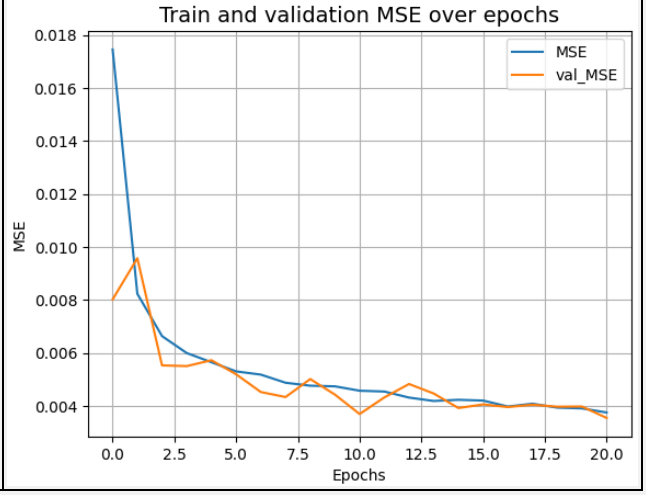


Παρατηρούμε ότι η ομαλοποίηση δεν επηρεάζει ιδιαίτερα την εκπαίδευση.

Α5. Βαθύ Νευρωνικό Δίκτυο

Για το ερώτημα αυτό χρησιμοποιήθηκε το μοντέλο με τις καλύτερες μετρικές, δηλαδή αυτό με ρυθμό μάθησης 0.05, ορμή 0.6, χωρίς ομαλοποίηση.

Κρυφά επίπεδα	CE loss	MSE	Acc
1	0.0502	0.0042	0.9869
2	0.0379	0.0035	0.9890
3	0.0402	0.0035	0.9887

2 κρυφά επίπεδα	3 κρυφά επίπεδα
	
	
	

Παρατηρούμε ότι με 2 κρυφά επίπεδα έχουμε τις καλύτερες μετρικές, ενώ με 3 έχουμε γρηγορότερη εκπαίδευση.