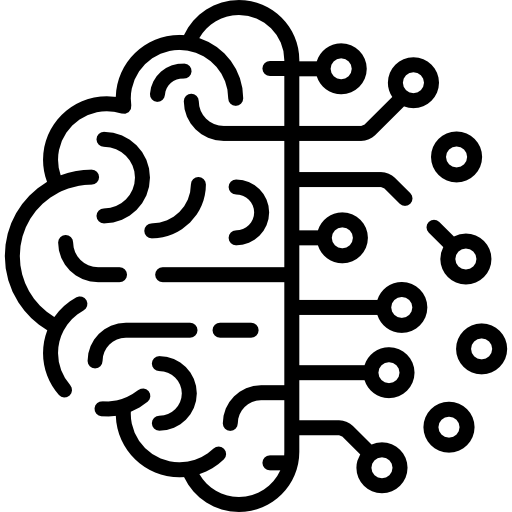


Υπολογιστική Νοημοσύνη

**Εργαστηριακή Άσκηση**

2023

**Μέρος Α**



**Στυλιανός Στυλιανάκης**  
**1059713**

[**Link to code**](https://github.com/Steliostyl/Computational_Intelligence_Project_2022-2023)**­­**

## Προεπεξεργασία και Προετοιμασία δεδομένων

### 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 περνάνε **τυποποίηση** (Standardization ή αλλιώς z-score), ώστε να αποφασίζει το νευρωνικό τη σημαντικότητα του καθενός.

Η κανονικοποίηση επιλέχθηκε αντί για κεντράρισμα επειδή 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.

Αντίστοιχα, δεν επιλέχθηκε η τυποποίηση επειδή

Τα κατηγορικά δεδομένα γίνονται **One-Hot Encode**.

### Διασταυρωμένη Επικύρωση

### Κεντράρισμα/Κανονικοποίηση/Τυποποίηση

Ανάλογα με την προεπεξεργασία είδα τα εξής αλλαγές στα αποτελέσματά μου

\*EPOCHS=50, Learning rate=0.001, Input Nodes=8520, Batch Size = 12, no early stopping, 1 Hidden Layer with N nodes and relu activation function, sigmoid for output

|  |  |  |
| --- | --- | --- |
|  | Raw Data | Κεντράρισμα |
| Train Binary Accuracy | 86.23% | 86.21% |
| Test Binary Accuracy | 86.11% | 86.11% |
| Train Acc | 35.18% | 35.56% |
| Test Acc | 34.07% | 35.25% |
| Train MSE | 0.1137 | 0.1103 |
| Test MSE | 0.1154 | 0.1121 |
| Train Loss | 0.3752 | 0.3662 |
| Test Loss | 0.3819 | 0.3730 |

|  |  |  |
| --- | --- | --- |
|  | Κανονικοποίηση | Τυποποίηση |
| Train Binary Accuracy | 85.52% | 85.52% |
| Test Binary Accuracy | 85.46% | 85.46% |
| Train Acc | 25.34% | 25.34% |
| Test Acc | 25.83% | 25.83% |
| Train MSE | 0.1260 | 0.1262 |
| Test MSE | 0.1263 | 0.1265 |
| Train Loss | 0.4212 | 0.4219 |
| Test Loss | 0.4218 | 0.4225 |

Σύμφωνα με τα πειράματά μου, το κεντράρισμα βοηθάει ελάχιστα ενώ τόσο η κανονικοποίηση όσο και η τυποποίηση μας δίνουν χειρότερα αποτελέσματα. Οπότε, στη συνέχεια θα χρησιμοποιήσω κεντράρισμα των δεδομένων.

### Cross-validation

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.

## Επιλογή αρχιτεκτονικής

### Εκπαίδευση και αξιολόγηση μοντέλου

Για την αξιολόγηση του νευρωνικού μου δικτύου χρησιμοποίησα Cross-Entropy και πιο συγκεκριμένα **Binary Cross Entropy**, καθώς τα labels μας είναι ανεξάρτητα μεταξύ τους και θέλουμε να παίρνουν τις τιμές 0 ή 1**.**

Η συνάρτηση *Cross Entropy*, αλλιώς γνωστή και ως **Log Loss** αθροίζει για κάθε κείμενο το loge(predicted\_value) των labels που είναι 1 μαζί με το loge(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).

### Νευρώνες εξόδου και Classification

Εφόσον οι κατηγορίες κειμένων που έχουμε είναι 20 θα έχουμε και 20 νευρώνες εξόδου. Επειδή κάθε κείμενο μπορεί να ανήκει σε περισσότερες από 1 κατηγορία, το πρόβλημά μας ανήκει στα προβλήματα τύπου multilabel classification.

### 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.

### Συνάρτηση ενεργοποίησης εξόδου

Για την έξοδο επιλέχθηκε η σιγμοειδής καμπύλη (sigmoid) ως συνάρτηση ενεργοποίησης, επειδή το πρόβλημα μας είναι classification και συγκεκριμένα Multilabel Classification. Αυτό σημαίνει ότι θέλουμε οι έξοδοι μας είναι ανεξάρτητες μεταξύ τους και θέλουμε να έχουν την τιμή 0 ή την τιμή 1. Κάτι τέτοιο επιτυγχάνεται με τη χρησιμοποίηση της συγκεκριμένης συνάρτησης (σε συνδυασμό με στρογγυλοποίηση).

### Αριθμός νευρώνων κρυφού επιπέδου

|  |  |  |  |
| --- | --- | --- | --- |
| Αριθμός νευρώνων στο κρυφό επίπεδο | CE Loss | MSE | Bin Acc |
| H1 = O | 0.3826 | 0.1154 | 85.86% |
| H1 = (I+O) / 2 | 0.3748 | 0.1128 | 86.10% |
| H1 = I + O | 0.3733 | 0.1122 | 86.18% |

#### Hidden Layer Nodes = Output

**Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated**

#### Hidden Layer Nodes = (Input + Output)/2

**Chart, line chart

Description automatically generated Table

Description automatically generated**

#### Hidden Layer Nodes = I + O

Chart, line chart

Description automatically generated Chart

Description automatically generated

Παρατηρώ ότι με την **αύξηση των κόμβων** του κρυφού επιπέδου εμφανίζεται και βελτίωση της απόδοσης του νευρωνικού μας δικτύου.

Η **ταχύτητα σύγκλισης** αυξάνεται κατά πολύ όταν αυξάνουμε τους αριθμούς των κόμβων του κρυφού επιπέδου από O σε (I+O)/2, ενώ με την περεταίρω αύξησή τους δεν παρατηρείται κάποια σημαντική αύξηση στην ταχύτητα σύγκλισης.

### Δεύτερο Hidden Layer H2

|  |  |  |  |
| --- | --- | --- | --- |
| Αριθμός νευρώνων στο κρυφό επίπεδο | CE Loss | MSE | Acc |
| H2 = Ο | 0.3693 | 0.1112 | 85.87% |
| H2 = (I+O) / 4 | 0.3620 | 0.1086 | 86.14% |
| H2 = (Ι+Ο) / 2 | 0.3675 | 0.1101 | 86.03% |

Είναι φανερό ότι το δεύτερο κρυφό επίπεδο μας δίνει τα καλύτερα αποτελέσματα όταν είναι μικρότερο από το 1ο (στην περίπτωσή μας έχει το μισό μέγεθος όταν H2 = (I+O) / 4 ), αλλά ταυτόχρονα μεγαλύτερο από το επίπεδο της εξόδου. Ενώ το CE Loss καθώς και το MSE φαίνονται μειωμένα, το Accuracy του δικτύου μας δε φαίνεται να έχει επηρεαστεί πολύ θετικά (αντιθέτως έχει πέσει λίγο), που σημαίνει ότι το πρόβλημά μας είναι πιθανό να μην επωφελείται σημαντικά από την αύξηση των κρυφών επιπέδων πάνω από 1.

### 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.

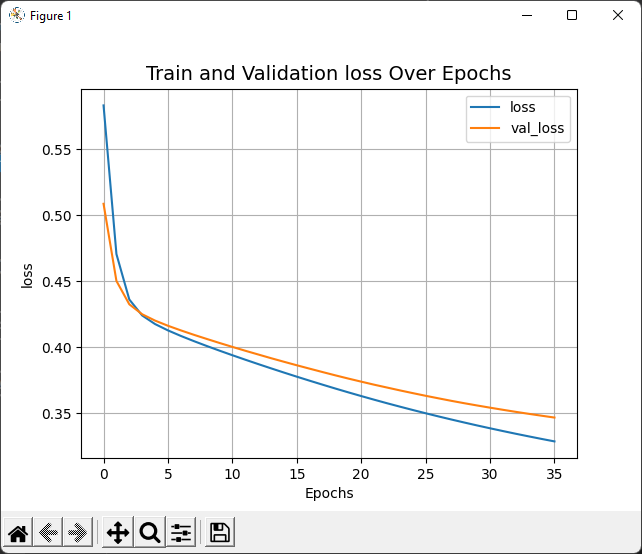
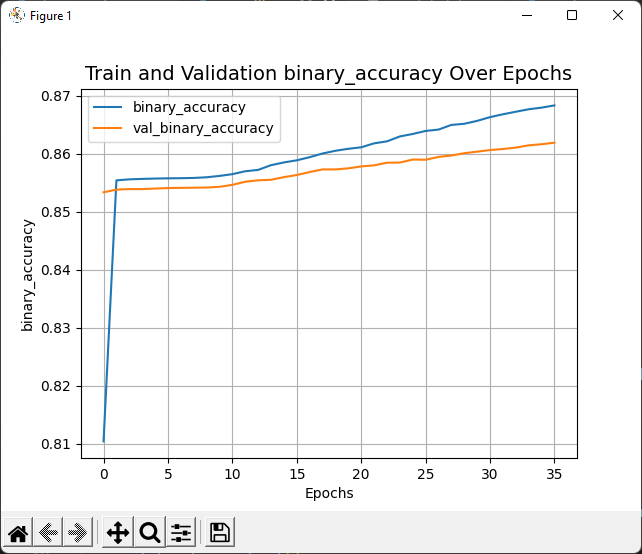
## Μεταβολές στο ρυθμό εκπαίδευσης και στη σταθερά ορμής

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 είναι απλή SGD. Για πολύ μεγάλες τιμές επηρεάζουν υπερβολικά την κίνησή του και γίνεται επιρρεπής σε μεγάλα σφάλματα.

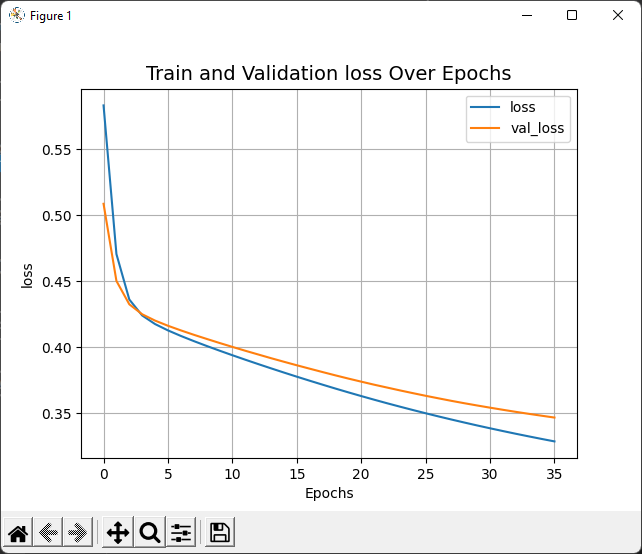
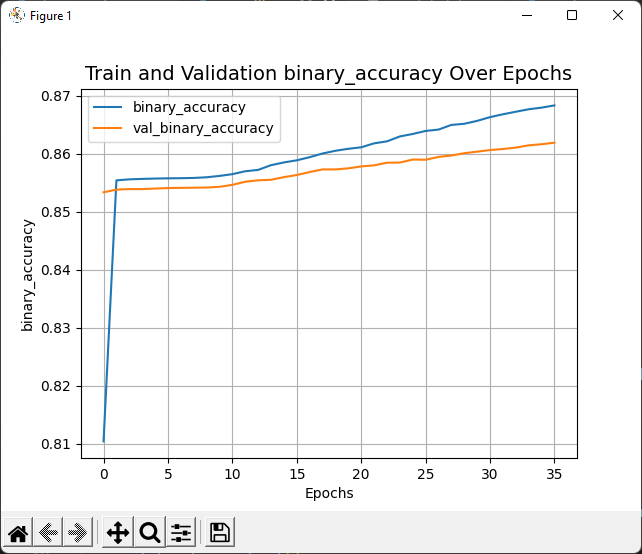
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| H | m | CE | MSE | Acc |
| 0.001 | 0.2 | 0.3612 | 0.1083 | 86.10% |
| 0.001 | 0.6 | 0.3436 | 0.1032 | 86.45% |
| 0.05 | 0.6 | 0.7765 | 0.1122 | 87.30% |
| 0.1 | 0.6 | 0.8487 | 0.1132 | 87.28% |

\* Για την 3η και την 4η σειρά του πίνακα δε χρησιμοποιήθηκε early stopping, ώστε να μεγαλοποιηθούν οι διαφορές και να φανεί ξεκάθαρα το overfitting.

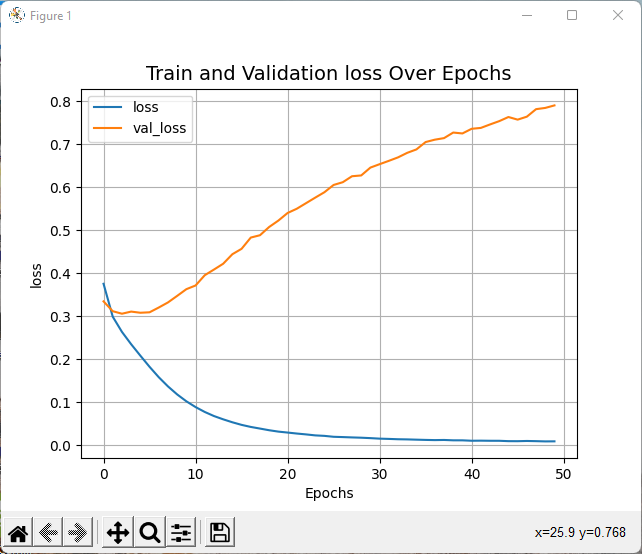
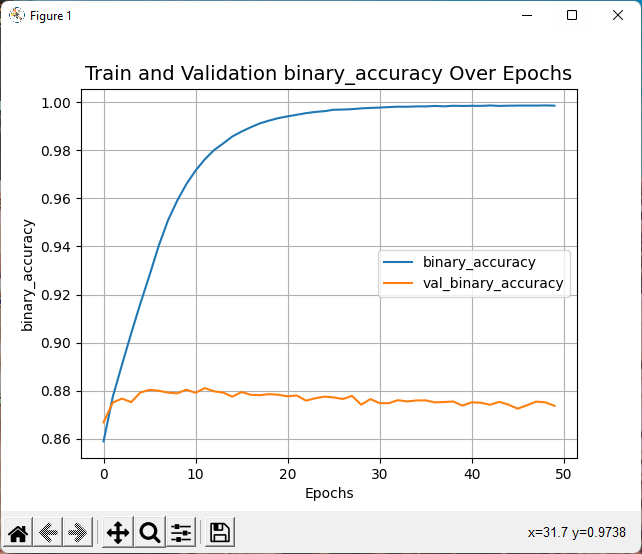
#### H = 0.001, m = 0.2

#### H = 0.001, m = 0.6

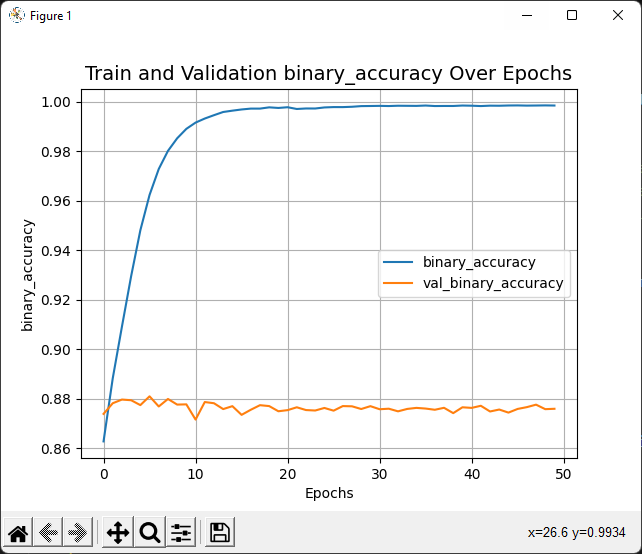
 

H = 0.05, m = 0.6

#### H = 0.1, m = 0.6

Chart, line chart

Description automatically generated 

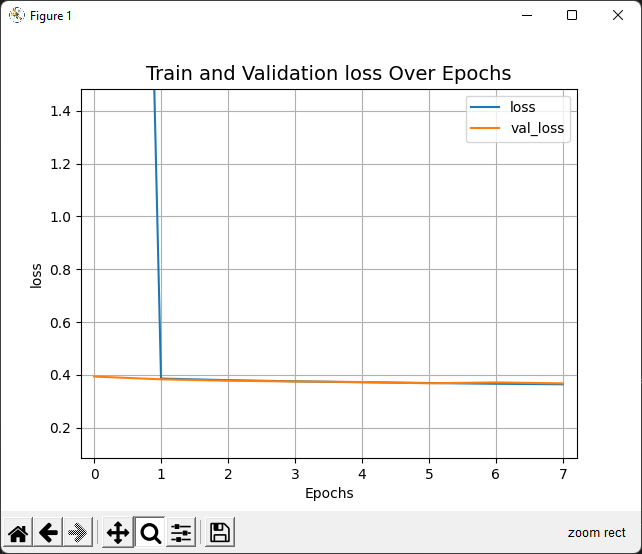
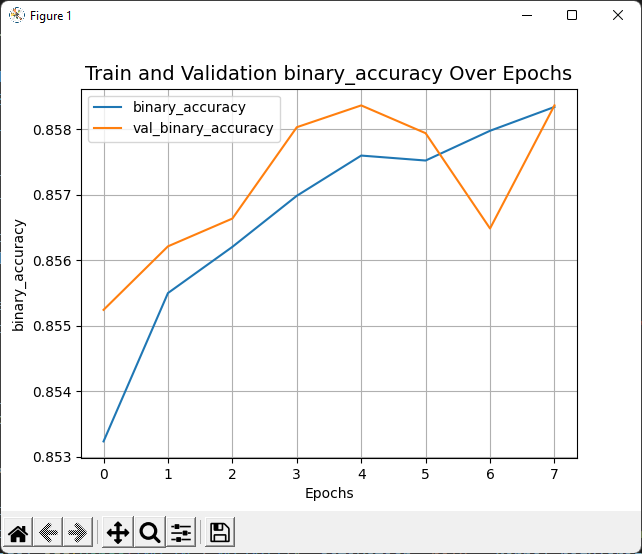
Παρατηρούμε ότι με ρυθμό μάθησης 0.05 και 0.1 υπάρχει ξεκάθαρο overfitting, δηλαδή το μοντέλο μας «αποστηθίζει» το dataset μας και το accuracy δε βελτιώνεται με το πέρασμα των epochs σε άλλες εισόδους.

## Ομαλοποίηση

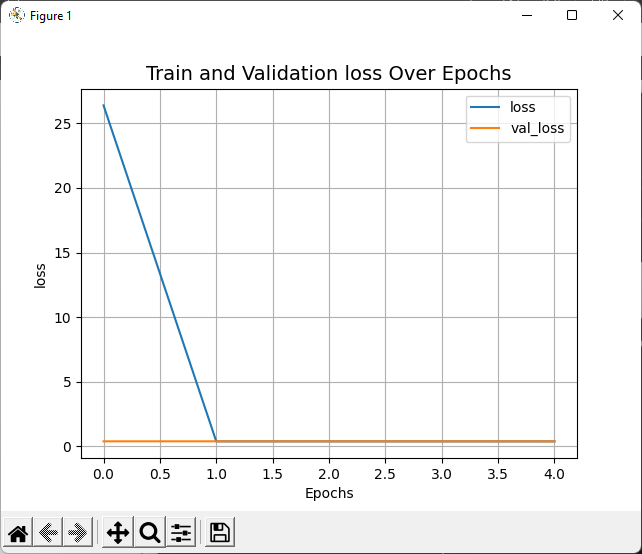
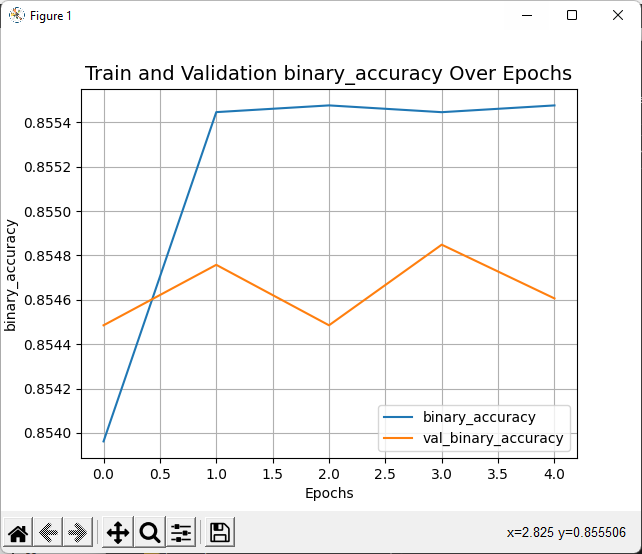
Με τη χρήση της ομαλοποίησης, το δίκτυό μας οδηγείται στην προτίμηση μικρότερων βαρών. Από το προηγούμενο ερώτημα παρατηρούμε ότι το μοντέλο με ρυθμό μάθησης H = 0.05 και ορμή m = 0.06 είχε την καλύτερη ακρίβεια, αλλά όπως προαναφέρθηκε έκανε overfit. Επειδή όμως η ομαλοποίηση με L2 είναι μία μέθοδος αποφυγής αυτού του προβλήματος, επέλεξα να κάνω αυτό το ερώτημα με τις παραμέτρους αυτές.

|  |  |  |  |
| --- | --- | --- | --- |
| Συντελεστής φθοράς | CE loss | MSE | Acc |
| 0.1 | 0.3656 | 0.1092 | 85.83% |
| 0.5 | 0.3902 | 0.1156 | 85.47% |
| 0.9 | 0.3892 | 0.1164 | 85.46% |

#### r = 0.1

#### r = 0.5

r = 0.9

