Πολυτεχνείο Κρήτης

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Επίλυση Προβλημάτων Ταυτοποίησης Ρωγμών με χρήση Τεχνητών Νευρωνικών Δικτύων

Επιμέλεια: Σταματέλου Ευθαλία

Επιβλέπων καθηγητής: Σταυρακάκης Γεώργιος

Μέλη επιτροπής: Σταυρουλάκης Γεώργιος, Ζερβάκης Μιχαήλ

ΧΑΝΙΑ 2022

Περίληψη

Στην παρούσα διπλωματική εργασία, παρουσιάζεται η επίλυση ενός αντίστροφου προβλήματος αναγνώρισης ρωγμών στη μηχανική με χρήση πολυστρωματικού νευρωνικού δικτύου που εκπαιδεύεται με οπισθόδρομη διάδοση σφάλματος (backpropagation). Παραδείγματα θα είναι η εύρεση ρωγμών με τη χρήση μετρήσεων στατική παραμόρφωσης ή αρμονικής διέγερσης. Πιο συγκεκριμένα, το εν λόγω πρόβλημα αφορά στον προσδιορισμό της ύπαρξης και των χαρακτηριστικών μιας κρυμμένης ρωγμής, η οποία βρίσκεται μέσα σε μια ελαστική δομή. Για το σκοπό αυτό χρησιμοποιούμε μετρήσεις της δομικής απόκρισης ,εντός των διαθέσιμων ορίων για συγκεκριμένες φορτώσεις. Το ευθύ πρόβλημα πρόκειται να επιλυθεί με μεθόδους υπολογιστικής μηχανικής (μέθοδος συνοριακών στοιχείων) και το αντίστροφο μέσω εφαρμογής του νευρωνικού δικτύου.. Ως προς την υλοποίηση του νευρωνικού δικτύου, πρόκειται να χρησιμοποιήσουμε Python λόγω της καλής της απόδοσης αλλά και της δυνατότητας που μας δίνει για βελτιστοποίηση των απαιτούμενων υπολογιστικών διαδικασιών. Οι υπάρχουσες βιβλιοθήκες κρίνονται κατάλληλες για οποιεσδήποτε γενικές εργασίες σε σχέση με την επεξεργασία των δεδομένων ενώ επίσης μας παρέχεται επαρκής κάλυψη σε περίπτωση που υπάρξει ανάγκη ενσωμάτωσης του ML σε άλλο λογισμικό. Επιπλέον, χρησιμοποιήθηκαν TensorFlow και Keras το οποία αποτελούν βιβλιοθήκες ανοιχτού κώδικα για αριθμητικούς υπολογισμούς υψηλής απόδοσης που αναπτύχθηκαν από την ομάδα του Google Brain. Οι βιβλιοθήκες αυτές μπορούν να εκπαιδεύσουν και να τρέξουν βαθιά πολυεπίπεδα νευρωνικά δίκτυα και χρησιμοποιούνται ευρέως στον τομέα της έρευνας και της εφαρμογής της βαθιάς μάθησης. Η μέθοδος είναι γενική και επεκτείνεται για την επίλυση αντιστρόφων προβλημάτων και προβλημάτων ταυτοποίησης παραμέτρων σε οποιοδήποτε πρόβλημα μηχανικής του συνεχούς για το οποίο υπάρχουν αντίστοιχα δεδομένα. Η υλοποίηση του νευρωνικού δικτύου θα παρουσιαστεί βήμα-βήμα με τη μορφή εγχειριδίου και εν τέλει πρόκειται να αναλυθούν τα αποτελέσματα που ελήφθησαν.

Abstract

In the present dissertation, the solution of an inverse problem of crack identification in engineering is presented using a multilayer neural network that is trained with backpropagation. Examples will be cracks in static or harmonic excitation (ie a problem that lies in the field of elastodynamics). More specifically, this problem concerns the identification of the existence and characteristics of a hidden crack, which is located within an elastic structure. For this purpose we use structural response measurements, within the available limits for specific loads. The direct problem is to be solved by methods of computer engineering (boundary element method) and the inverse problem is being solved through the application of a backpropagation neural network. As for the implementation of the neural network, we are going to use Python due to its good performance and the ability it gives us to optimize the required computing processes. Existing libraries are considered suitable for any general work related to data processing and we are also provided with sufficient coverage in case there is a need to integrate ML into other software. In addition, TensorFlow and Keras are used, which are open source libraries for high performance numerical calculations developed by the Google Brain team. They can train and run deep multilevel neural networks and are widely used in the field of research and application of deep learning. The method is general and can be used for the solution of inverse and parameter identification problems for every problem of mechanics of continuum for which similar data are available. The implementation of the neural network will be presented step by step in the form of a manual and finally the results obtained will be analyzed.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, καθηγητή ΗΜΜΥ κ. Γεώργιο Σταυρακάκη για την ανάθεση της διπλωματικής μου εργασίας όπως επίσης και για τη χρήσιμη καθοδήγηση του.

Επιπλέον, θα ήθελα να ευχαριστήσω πολύ τον καθηγητή ΜΠΔ κ. Γεώργιο Σταυρουλάκη, μέλος της τριμελούς επιτροπής, για τη σημαντική βοήθειά του, την επιστημονική του υποστήριξη αλλά και για το χρόνο που αφιέρωσε για την επίλυση αποριών προκειμένου να ολοκληρωθεί η εν λόγω διπλωματική εργασία.

Τέλος, επιθυμώ να ευχαριστήσω τον καθηγητή ΗΜΜΥ κ. Μιχαήλ Ζερβάκη, ο οποίος δέχτηκε να είναι μέλος της τριμελούς επιτροπής αξιολόγησης της διπλωματικής μου εργασίας.

Περιεχόμενα

Κεφάλαιο 1

1.1 Εισαγωγή

Αντικείμενο της παρούσας έρευνας, αποτελεί η ανάπτυξη πειραματικής λύσης για μη καταστρεπτικά προβλήματα αναγνώρισης ρωγμών σε υλικά, τα οποία χαρακτηρίζονται από γραμμική ελαστική συμπεριφορά. Τα προβλήματα ταυτοποίησης ανήκουν στην κατηγορία των αντίστροφων προβλημάτων υπό την έννοια ότι για αυτά μπορεί να οριστεί ένα μοντέλο συστήματος, μπορούν να οριστούν δεδομένα εισόδου και εξόδου, αλλά οι τιμές των παραμέτρων που επηρεάζουν αυτό το σύστημα παραμένουν άγνωστες. Έτσι ο διαχωρισμός μεταξύ ευθέων και αντίστροφων προβλημάτων παραμένει σχετικά αυθαίρετος.

Θεωρείται πως η δομή που εξετάζεται έχει γραμμική-ελαστική συμπεριφορά. Έτσι, η αρχή της στατικής-δυναμικής ομοιότητας επιτρέπει τη χρήση ελαστοστατικών τεχνικών επίλυσης οριακών στοιχείων (Boundary Element Method) με πίνακες οι οποίοι εξαρτώνται από τη συχνότητα διέγερσης. Φυσικά, κάτι τέτοιο αφορά στην αριθμητική αντιμετώπιση του άμεσου προβλήματος και συντελέστηκε στην εργασία Neural crack identification in steady state elastodynamics[1], συνέχεια της οποίας αποτελεί η παρούσα διπλωματική εργασία. Το αντίστροφο πρόβλημα, γύρω από το οποίο διαρθρώνεται η συγκεκριμένη εργασία, αφορά στον υπολογισμό των χαρακτηριστικών των ρωγμών (συντεταγμένες κέντρου) και επιλύεται με νευρωνικό δίκτυο οπίσθιας διάδοσης. Περισσότερες πληροφορίες για την αντιμετώπιση στατικών προβλημάτων και δυναμικών προβλημάτων στο πεδίο του χρόνου καθώς και σύνδεση με τεχνικές επεξεργασίας μεγάλων δεδομένων στη μηχανική βρίσκονται στις δημοσιεύσεις [89], [90].

Πιο συγκεκριμένα, ο προσδιορισμός του σφάλματος εισόδου-εξόδου διατυπώνεται ως ένα πρόβλημα βελτιστοποίησης το οποίο έχει να κάνει με τη διαφορά μεταξύ των μετρούμενων και των επιθυμητών αποκρίσεων εντός του χώρου των μεταβλητών που οριοθετούν την εξεταζόμενη δομή και τις αναμενόμενες βλάβες. Αυτή η επιλογή γίνεται, καθώς οι κλασικές προσεγγίσεις βελτιστοποίησης δεν είναι πάντα συμφέρουσες λόγω της φύσης του προβλήματος που εξετάζεται. Πιο συγκεκριμένα, στο αντίστροφο πρόβλημα, μικρές αλλαγές μιας ορισμένης δομικής παραμέτρου μπορεί να οδηγήσουν σε μεγάλες ή μικρές διακυμάνσεις της δομικής απόκρισης ανάλογα με τη θέση ή τον τύπο της παραμέτρου. Επιπλέον, λόγω της μη γραμμικότητας της δομικής απόκρισης, ως συνάρτηση των μεταβλητών που προσδιορίζουν μια ρωγμή, το πρόβλημα βελτιστοποίησης που προκύπτει είναι συνήθως μη κυρτό. Έτσι, υπάρχει η πιθανότητα το μαθηματικό μοντέλο για το αντίστροφο πρόβλημα να επιδέχεται πολλαπλών λύσεων, να αντιστοιχεί δηλαδή σε πρόβλημα με πολλά τοπικά ελάχιστα. Έτσι τεχνικές επίλυσης όπως αυτή των νευρωνικών δικτύων είναι προτιμητέες καθώς έχουν την ικανότητα να προσαρμόζονται αυτόματα στις αλλαγές κλίμακας, ξεπερνώντας το προαναφερθέν πρόβλημα.

Παρόλο που για άλλες εφαρμογές στη μηχανική τα δεδομένα θα μπορούσαν να μετρηθούν πειραματικά, στη συγκεκριμένη περίπτωση για την εκπαίδευση και τον έλεγχο της αποτελεσματικότητας του μοντέλου που κατασκευάστηκε, τα δεδομένα είναι αποτέλεσμα μιας τεχνικής υπολογιστικής μηχανικής που βασίζεται σε συνοριακά στοιχεία.(BEM). Πρόκειται

άλλωστε για μια μεθοδολογία η οποία εφαρμόζεται γενικότερα στον ποιοτικό έλεγχο δομών ή επιμέρους δομικών στοιχείων.

Επιπλέον, τονίζεται πως στην παρούσα εργασία γίνεται η υπόθεση της γραμμικής συμπεριφοράς ρωγμών χωρίς πρόσφυση. Συνεπώς, το πλάτος των δονήσεων που προκαλούνται στη δομή λόγω της διέγερσης θεωρείται πως είναι μικρότερο από το πλάτος των ρωγμών που υπάρχουν ούτως ώστε να μην προκύπτουν μονόπλευρες επαφές ή τριβές μεταξύ των πλευρών της ρωγμής.

1.2 Σκοπός της εν λόγω εργασίας

Το πρόβλημα που μελετάται έχει επιλυθεί με τη χρήση αλγορίθμων και πακέτων νευρωνικών δικτύων σε περιβάλλον Matlab [89]. Όσων αφορά στην παρούσα εργασία, αποφασίστηκε η εναλλαγή της λύσης του αντίστροφου προβλήματος ταυτοποίησης ρωγμών με χρήση σύγχρονων αποδοτικότερων μέσων αλλά και η παρουσίαση αυτής με τη μορφή εγχειριδίου. Αναλυτικότερα, επιλέχθηκε η Python δεδομένου ότι είναι μια δυναμική γλώσσα προγραμματισμού υψηλού επιπέδου η οποία παρέχει τη δυνατότητα χρήσης εξειδικευμένων πακέτων σε ότι αφορά όχι μόνο στα νευρωνικά δίκτυα αλλά και στην υλοποίηση πολύπλοκων μαθηματικών υπολογισμών. Αναφορικά με τα πακέτα (η αλλιώς βιβλιοθήκες) χρησιμοποιήθηκαν το TensorFlow, το Keras και το Pandas τα οποία θα αναλυθούν στη συνέχεια. Κρίνεται σκόπιμο να αναφερθεί πως ουσιαστικό στόχο της εργασίας αποτελεί η συμβολή στην έρευνα σε σχέση με τα νευρωνικά δίκτυα εφόσον αποτελούν ένα από τα πιο σύγχρονα και ακριβή εργαλεία επίλυσης σύνθετων μαθηματικών προβλημάτων. Τέλος, σκόπιμη είναι και η επιλογή των μέσων χάρη στα οποία αναπτύχθηκε το νευρωνικό δίκτυο καθώς πρόκειται για υπερσύγχρονα εργαλεία ανοιχτού κώδικα τα οποία μπορούν να επαναχρησιμοποιηθούν ελεύθερα σε μελλοντική έρευνα.

1.3 Διάρθρωση της εργασίας

Στο Κεφάλαιο 2 δίνεται μια σύντομη περιγραφή, αρχικά για το ευθύ πρόβλημα και έπειτα για το αντίστροφο πρόβλημα το οποίο αποτελεί αντικείμενο της παρούσας εργασίας.

Στο Κεφάλαιο 3 αναλύονται οι όροι της τεχνητής νοημοσύνης, της μηχανικής μάθησης και της βαθιάς μάθησης όπως επίσης και οι διαφορές μεταξύ τους.

Στο Κεφάλαιο 4 παρέχεται λεπτομερής ανάλυση των νευρωνικών δικτύων, των επιμέρους στοιχείων τους ενώ επίσης γίνεται αναφορά στη σημασία των συνόλων δεδομένων που επεξεργάζονται αντίστοιχα τα μοντέλα νευρωνικών δικτύων.

Στο Κεφάλαιο 5 αναπτύσσεται το εγχειρίδιο μεταβολής του κώδικα Matlab σε κώδικα Python, ενώ επίσης δίνονται περαιτέρω πληροφορίες σχετικά με τα στοιχεία που απαρτίζουν το νευρωνικό δίκτυο και επισημαίνονται ορισμένες διαφορές.

Στο Κεφάλαιο 6 παρουσιάζονται συγκεντρωτικά τα αποτελέσματα της κάθε υλοποίησης.

Το Κεφάλαιο 7 περιέχει τον επίλογο, τα συμπεράσματα και τις προτάσεις για μελλοντική έρευνα.

Το Παράρτημα περιλαμβάνει τον κώδικα σε Python.

Κεφάλαιο 2

2.1 Εισαγωγή

Στη συγκεκριμένη εργασία, αναπτύσσεται μια προσέγγιση για την επίλυση του αντίστροφου προβλήματος της θέσης ενός ελαττώματος σε μια ελαστική πλάκα, συνδυάζοντας ένα τεχνητό νευρωνικό δίκτυο (ANN) και τη μέθοδο των συνοριακών στοιχείων (BEM). Το αντίστροφο πρόβλημα διατυπώνεται ως πρόβλημα παλινδρόμησης, το οποίο εξάγει τη λύση της υψηλότερης πιθανότητας μέσω της μηχανής μάθησης (ML) από έναν συγκεκριμένο όγκο δεδομένων.

2.2 Ευθύ Πρόβλημα

2.2.1 Ευθύ Πρόβλημα

Η αποτελεσματικότητα και η ακρίβεια της παραγωγής δεδομένων είναι εγγυημένη με τη χρήση μεθόδων υπολογιστικής μηχανικής. Τα δεδομένα που χρησιμοποιήθηκαν εδώ παρήχθησαν με τη χρήση της μεθόδου συνοριακών στοιχείων (boundary element method), η οποία συσχετίζει τιμές παραμόρφωσης στα όρια πλακών με τις αντίστοιχες δυνάμεις. Επιπλέον σύνορα περιγράφουν κυκλικές οπές στο εσωτερικό του παραμορφώσιμου σώματος οι οποίες προσομοιάζουν βλάβες και λοιπά ελαττώματα. Με τη χρήση του υπολογιστικού μοντέλου για διάφορες θέσεις και μεγέθη βλαβών δημιουργείται το σύνολο δεδομένων που θα χρησιμοποιηθούν κατά τη διαδικασία εκπαίδευσης του νευρωνικού δικτύου. Ουσιαστικά αντικείμενο μελέτης αποτελεί μια πλάκα στην οποία τοποθετούνται οπές (κυκλικά ελαττώματα) και την οποία φορτίζουμε πάντα με την ίδια δύναμη και στηρίζουμε στις ίδιες θέσεις. Αυτό το οποίο εν τέλει υπολογίζουμε είναι η παραμορφωμένη εικόνα της πλάκας αυτής, καθώς αντικείμενο του ενδιαφέροντός μας αποτελούν οι μετακινήσεις των στοιχείων στην εξωτερική πλευρά της πλάκας. Αυτά τα χαρακτηριστικά μπορούν να μετρηθούν εφόσον δεν μπορούμε να γνωρίζουμε τη συμβαίνει στο εσωτερικό του σώματος αυτού. Άλλωστε στην υπολογιστική μηχανική συνήθως υπολογίζουμε την κατανομή της παραμόρφωσης με βάση τη γνωστή θέση του ελαττώματος και οριακές συνθήκες. Ο κώδικας QUADPLEH χρησιμοποιείται για να ληφθεί η κατανομή της παραμόρφωσης στους κόμβους αυτής της ελαττωματικής πλάκας με τις βλάβες [1].

Η επιλογή της μεθόδου οριακών στοιχείων έγινε, καθώς το γεγονός πως η ίδια βασίζεται σε οριακές ολοκληρωτικές εξισώσεις μπορεί να προσφέρει υψηλή ακρίβεια. Η διαδικασία διακριτοποίησης πραγματοποιείται μόνο στα σύνορα όπου οι λύσεις χρειάζονται ως είσοδοι, και ως εκ τούτου, μειώνεται αμέσως ο υπολογιστικός φόρτος εργασίας για τη δημιουργία δεδομένων. Εναλλακτικά μπορεί να χρησιμοποιηθεί η περισσότερο διαδεδομένη μέθοδος των πεπερασμένων στοιχείων, η οποία όμως απαιτεί τη διακριτοποίηση και του εσωτερικού του παραμορφώσιμου σώματος και οδηγεί σε αυξημένες απαιτήσεις υπολογιστικής δύναμης και χρόνου υπολογισμών.

2.2.2 Ορισμός του προβλήματος και της λύσης του με τη χρήση της μεθόδου συνοριακών στοιχείων - Boundary Element Method

Η μέθοδος οριακών στοιχείων (BEM) είναι μια αριθμητική υπολογιστική μέθοδος επίλυσης γραμμικών μερικών διαφορικών εξισώσεων που έχουν διαμορφωθεί ως ολοκληρωτικές εξισώσεις (δηλαδή σε οριακή ακέραια μορφή). Παραδείγματα εφαρμογής της BEM συμπεριλαμβάνουν τη μηχανική ρευστών, την ακουστική, την ηλεκτρομαγνητική ή ακόμα και τον ποιοτικό έλεγχο υλικών όπως στην περίπτωση που εξετάζεται στην παρούσα εργασία.[2]

2.3 Αντίστροφο πρόβλημα

Όσων αφορά στο αντίστροφο πρόβλημα που εξετάζεται στην παρούσα εργασία,αυτό αφορά στην εύρεση παραμέτρων που χαρακτηρίζουν μια ελλειψοειδή ρωγμή, από μετρήσεις μετακινήσεων στην εξωτερική πλευρά ενός στατικά φορτισμένου δίσκου. Αντίθετα με το ευθύ πρόβλημα, οι οριακές συνθήκες παραμένουν αμετάβλητες ωστόσο ποικίλει η τοποθεσία των ελαττωμάτων (ρωγμών). Το νευρωνικό δίκτυο δηλαδή μαθαίνει τη σχέση: x $\longrightarrow$ z, όπου x είναι οι μετρήσεις των μετακινήσεων και z οι συντεταγμένες του κέντρου της εκάστοτε ρωγμής.

Το ANN, το οποίο κατασκευάζεται , εκπαιδεύεται χρησιμοποιώντας το σύνολο δεδομένων που παράχθηκαν από τη μέθοδο οριακών στοιχείων που αναφέρθηκε νωρίτερα προκειμένου να προβλεφθούν οι κεντρικές συντεταγμένες των κυκλικών οπών. Η προτεινόμενη προσέγγιση απαιτεί μόνο τα δεδομένα των οριακών τάσεων της εξεταζόμενης δομής ως δεδομένα εισόδου και μια απλή διαδικασία εκπαίδευσης. Συνεπώς, κατασκευάζεται ένα αποτελεσματικό εποπτευόμενο μοντέλο μηχανικής μάθησης το οποίο βασίζεται σε αλγόριθμο παλινδρόμησης και εκπαιδεύεται καταγράφοντας τη σχέση μεταξύ των εισόδων (παραμορφώσεων στο σύνορο) και τη θέση των ελαττωμάτων (στοιχεία εξόδου). Εν τέλει, δημιουργούνται οι τοποθεσίες των οπών χρησιμοποιώντας το σύνολο δεδομένων δοκιμής.

Γενικότερα, η επίλυση του αντίστροφου προβλήματος που αναλύθηκε στο Κεφάλαιο 1 διεκπεραιώνεται με τη χρήση ενός τεχνητού νευρωνικού δικτύου του οποίου η αρχιτεκτονική αλλά και περαιτέρω τεχνικές λεπτομέρειες της κατασκευής του, πρόκειται να αναλυθούν στα επόμενα κεφάλαια. Συνοπτικά, οι είσοδοί του είναι 82, όσοι δηλαδή και οι κόμβοι στους οποίους έχει διακριτοποιηθεί το εξωτερικό όριο της πλάκας που μελετάμε, και αποτελούν δεδομένα παραμορφώσεων εφόσον έχει προηγηθεί φόρτιση με μία συγκεκριμένη δύναμη στο εξωτερικό όριο της πλάκας αυτής. Πρόκειται δηλαδή για ένα στατικό πρόβλημα όπου οι μετρήσεις της πλάκας σε επιλεγμένα σημεία συσχετίζονται με τη θέση των βλαβών. Από τη στιγμή λοιπόν που οι εξωτερικές συνοριακές συνθήκες είναι γνωστές, άγνωστες σε αυτή την περίπτωση είναι οι ρωγμές που υπάρχουν στην πλάκα. Συνεπώς οι έξοδοι του δικτύου είναι δύο και αντιπροσωπεύουν τις συντεταγμένες του κέντρου του εκάστοτε ελαττώματος.

Ευρύτερα, οι τιμές της παραμόρφωσης μαζί με τις κεντρικές συντεταγμένες των ρωγμών χωρίζονται ως δεδομένα εκπαίδευσης, και δοκιμής τα οποία χρησιμοποιούνται αντίστοιχα για τη διαδικασία εκπαίδευσης και τον έλεγχο του νευρωνικού δικτύου. Τα αποτελέσματα του τεχνητού νευρωνικού δικτύου θα δοθούν συγκεντρωτικά ενώ πρόκειται να παρουσιαστεί ξεχωριστά η επίδοσή του σε σχέση με τα δεδομένα πάνω στα οποία εκπαιδεύτηκε αλλά και σε σχέση με άλλα

με τα οποία δεν είχε έρθει σε επαφή (test data).Τέλος, πρόκειται να παρουσιαστούν και αποτελέσματα που αφορούν σε linear regression μεταξύ των τιμών που έχουν υπολογιστεί μέσω του νευρωνικού δικτύου και των τιμών-στόχων.

Κεφάλαιο 3

3.1 Τεχνητή Νοημοσύνη

Ο όρος τεχνητή νοημοσύνη αφορά στον κλάδο της πληροφορικής ο οποίος έχει να κάνει με τη σχεδίαση και την υλοποίηση ευφυών υπολογιστικών συστημάτων, τα οποία μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς, τέτοια ώστε να υπονοούν έστω και στοιχειώδη ευφυΐα: μάθηση, προσαρμοστικότητα, εξαγωγή συμπερασμάτων, κατανόηση από συμφραζόμενα, επίλυση προβλημάτων κ.λπ. Ο Τζον Μακάρθι όρισε τον τομέα αυτόν ως «επιστήμη και μεθοδολογία της δημιουργίας νοημόνων μηχανών».[3]

Ο όρος εισάχθηκε για πρώτη φορά το 1956 στη διάσκεψη του Dartmouth. Ως ευφυή συστήματα θεωρούμε τα υπολογιστικά εκείνα συστήματα που σκέφτονται και ενεργούν όπως ο άνθρωπος. Η τεχνητή νοημοσύνη εφάπτεται πολλών επιστημών πέρα από αυτή της πληροφορικής όπως της ψυχολογίας, της φιλοσοφίας, της γλωσσολογίας, της λογικής κ.λπ. Ένα από τα βασικά ερωτήματα των επιστημόνων είναι το κατά πόσον είναι εφικτή η τεχνητή νοημοσύνη. Ο Alan Turing το 1950 πρότεινε μία δοκιμασία (ένα test) που εξετάζει την ικανότητα μιας μηχανής να επιδεικνύει ευφυή συμπεριφορά που ισοδυναμεί με αυτή ενός ανθρώπου. Για να περάσει αυτή τη δοκιμασία ένας υπολογιστής θα πρέπει να έχει διάφορες ικανότητες όπως επεξεργασία φυσικής γλώσσας, αναπαράσταση γνώσης, μηχανική όραση κ.λπ.

Ένα κλασσικό παράδειγμα της δοκιμασίας Turing είναι το εξής: Έχουμε δύο δωμάτια που δεν επικοινωνούν μεταξύ τους, δύο ανθρώπους και έναν υπολογιστή. Ο ένας άνθρωπος τοποθετείται στο πρώτο δωμάτιο ενώ ο άλλος στο δεύτερο μαζί με τον υπολογιστή. Ο άνθρωπος στο δεύτερο δωμάτιο ή ο υπολογιστής (η επιλογή είναι τυχαία) απαντούν σε μία ερώτηση. Οι απαντήσεις αποστέλλονται στον άνθρωπο του πρώτου δωματίου με γραπτό κείμενο κάθε φορά. Ο άνθρωπος αυτός πρέπει από την απάντηση που λαμβάνει να αποφασίσει αν αυτή έχει δοθεί από τον Η/Υ ή από τον άνθρωπο. Η δοκιμασία δεν εξετάζει αν οι απαντήσεις είναι ορθές αλλά αν προέρχονται από άνθρωπο ή Η/Υ. Αν από τις απαντήσεις δεν είναι δυνατό να διευκρινιστεί ποια απάντηση δόθηκε από τον Η/Υ και ποια από τον άνθρωπο, τότε ο άνθρωπος και ο Η/Υ έχουν παρόμοια ευφυία.[4]

3.2 Μηχανική Μάθηση

3.2.1 Ανάλυση

Η μηχανική μάθηση μπορεί να οριστεί ως το φαινόμενο κατά το οποίο ένα σύστημα βελτιώνει την απόδοσή του κατά την εκτέλεση μιας συγκεκριμένης εργασίας, χωρίς να υπάρχει ανάγκη να προγραμματιστεί εκ νέου. Βάσει του ορισμού αυτού, η μηχανική μάθηση έχει ως σκοπό τη δημιουργία μηχανών ικανών να μαθαίνουν, να βελτιώνουν, δηλαδή, την απόδοσή τους σε κάποιους τομείς μέσω της αξιοποίησης προηγούμενης γνώσης και εμπειρίας. Ένας σχετικός γενικός ορισμός Μηχανικής Μάθησης δίνεται από τον Mitchell (1997): «Ένα πρόγραμμα

υπολογιστή λέμε ότι μαθαίνει από την εμπειρία E ως προς κάποια κλάση εργασιών T και μέτρο απόδοσης P, αν η απόδοσή του σε εργασίες από το T, όπως μετριέται από το P, βελτιώνεται μέσω της εμπειρίας E.»[6]

Στην επαγωγική μάθηση (Inductive Learning), με τη διαδικασία της επαγωγής (induction) ο άνθρωπος μαθαίνει κατανοώντας το περιβάλλον του μέσω παρατηρήσεων και δημιουργεί μια απλοποιημένη (αφαιρετική) εκδοχή του που ονομάζεται νοητικό μοντέλο (mental model). Επιπλέον, ο άνθρωπος έχει τη δυνατότητα να οργανώνει και να συσχετίζει τις εμπειρίες και τις παρατηρήσεις του δημιουργώντας νέες δομές που ονομάζονται νοητικά πρότυπα (mental patterns), με αξιοποίηση και του επαγωγικού και του απαγωγικού συλλογισμού. Στη δημιουργία νέων προτύπων από παλαιά βασίζονται οι τρόποι μάθησης που εξαρτώνται σε μεγαλύτερο ή μικρότερο βαθμό από την προϋπάρχουσα γνώση για ένα πρόβλημα, όπως είναι η μάθηση από επεξηγήσεις και η μάθηση από περιπτώσεις. Σε σχέση με την ανθρώπινη ικανότητα προς μάθηση, οι φιλόσοφοι θέτουν το ερώτημα: «Πώς μπορεί ένας επαγωγικός συλλογισμός που οδηγεί στη μάθηση να αξιολογηθεί ως προς την ορθότητά του;». Αντίστοιχα, οι ψυχολόγοι ρωτούν: «Πώς αποθηκεύει ο εγκέφαλος τα αποτελέσματα της διαδικασίας της μάθησης, δηλαδή τα νοητικά μοντέλα και τα πρότυπα;». Στο χώρο της τεχνητής νοημοσύνης απλώς ρωτούν: «Πώς μπορεί μία μηχανή να δημιουργήσει νέα μοντέλα και πρότυπα μάθησης από συγκεκριμένα παραδείγματα και πόσο αξιόπιστα είναι αυτά τα μοντέλα και πρότυπα στην πράξη;». Με βάση τα παραπάνω, μπορεί να δοθεί ο ακόλουθος εναλλακτικός ορισμός για τη μηχανική μάθηση: Μηχανική Μάθηση ονομάζεται η ικανότητα ενός υπολογιστικού συστήματος να δημιουργεί μοντέλα ή πρότυπα από ένα σύνολο δεδομένων.

Ως κλάδος της τεχνητής νοημοσύνης, η μηχανική μάθηση ασχολείται με τη μελέτη αλγορίθμων που βελτιώνουν τη συμπεριφορά τους σε κάποια εργασία που τους έχει ανατεθεί χρησιμοποιώντας την εμπειρία τους. Όσον αφορά τη σχεδίαση των συστημάτων μηχανικής μάθησης, για τα συστήματα που ανήκουν στη συμβολική τεχνητή νοημοσύνη, η δυνατότητα μάθησης προσδιορίζεται ως η ικανότητα πρόσκτησης επιπλέον γνώσης, που επιφέρει μεταβολές στην υπάρχουσα καταχωρημένη γνώση είτε αλλάζοντας χαρακτηριστικά της είτε με αυξομείωσή της. Στην περίπτωση των συστημάτων τεχνητής νοημοσύνης που ανήκουν στη μη συμβολική τεχνητή νοημοσύνη (όπως η περίπτωση των τεχνητών νευρωνικών δικτύων), ως μάθηση προσδιορίζεται η δυνατότητα που διαθέτουν τα συστήματα στο να μετασχηματίζουν την εσωτερική τους δομή, παρά στο να μεταβάλλουν κατάλληλα τη γνώση που έχει καταχωρηθεί μέσα σε αυτά κατά το σχεδιασμό τους.

3.2.2 Είδη Μηχανικής Μάθησης

Εν γένει, ο τομέας της Μηχανικής Μάθησης αναπτύσσει τρεις τρόπους μάθησης, ανάλογους με τους τρόπους με τους οποίους μαθαίνει ο άνθρωπος: επιβλεπόμενη μάθηση, μη επιβλεπόμενη μάθηση και ενισχυτική μάθηση.

Πιο αναλυτικά:

- Επιβλεπόμενη Μάθηση (Supervised Learning) είναι η διαδικασία όπου ο αλγόριθμος κατασκευάζει μια συνάρτηση που απεικονίζει δεδομένες εισόδους (σύνολο εκπαίδευσης) σε γνωστές επιθυμητές εξόδους, με απώτερο στόχο τη γενίκευση της συνάρτησης αυτής

και για εισόδους με άγνωστη έξοδο. Χρησιμοποιείται σε προβλήματα:

- Ταξινόμησης (Classification)
- Πρόγνωσης (Prediction)
- Διερμηνείας (Interpretation)

- Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning), όπου ο αλγόριθμος κατασκευάζει ένα μοντέλο για κάποιο σύνολο εισόδων υπό μορφή παρατηρήσεων χωρίς να γνωρίζει τις επιθυμητές εξόδους. Χρησιμοποιείται σε προβλήματα:

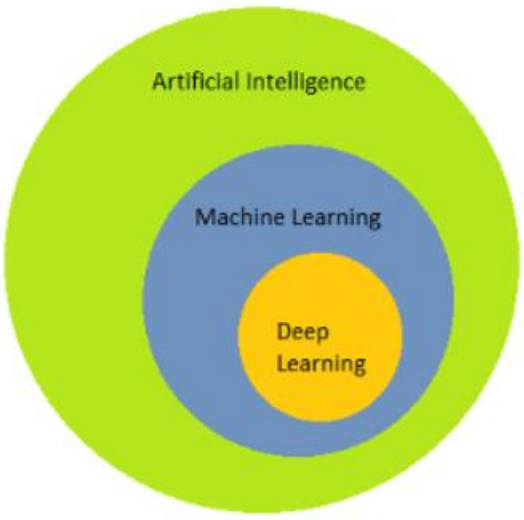
- Ανάλυσης Συσχετισμών (Association Analysis)
- Ομαδοποίησης (Clustering)

- Ενισχυτική Μάθηση (Reinforcement Learning), όπου ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών μέσα από άμεση αλληλεπίδραση με το περιβάλλον. Χρησιμοποιείται κυρίως σε προβλήματα σχεδιασμού (Planning), όπως για παράδειγμα ο έλεγχος κίνησης ρομπότ και η βελτιστοποίηση εργασιών σε εργοστασιακούς χώρους.[5]

3.3 Deep Learning

3.3.1 Ανάλυση

Η βαθιά μάθηση είναι μια υποκατηγορία της μηχανικής μάθησης και της τεχνητής νοημοσύνης (AI)(εικόνα 1), η οποία μιμείται τον τρόπο με τον οποίο οι άνθρωποι αποκτούν ορισμένους τύπους γνώσης. Η βαθιά μάθηση είναι ένα σημαντικό στοιχείο της επιστήμης δεδομένων, η οποία περιλαμβάνει στατιστικά και προγνωστικά μοντέλα.[7]



Εικόνα 1

Στη βαθιά μάθηση, ένας αλγόριθμος υπολογιστή μαθαίνει να εκτελεί εργασίες ταξινόμησης

απευθείας σε πολύπλοκα δεδομένα τα οποία βρίσκονται μορφή εικόνων, κειμένου ή ήχου. Αυτοί οι αλγόριθμοι μπορούν να επιτύχουν ακρίβεια αιχμής και μερικές φορές είναι ικανοί ακόμα και να ξεπεράσουν την απόδοση σε ανθρώπινο επίπεδο. Εκπαιδεύονται μέσω μεγάλου συνόλου δεδομένων τα οποία χρησιμοποιούνται σαν είσοδοι σε νευρωνικά δίκτυα, η χρησιμότητα και ο τρόπος λειτουργίας των οποίων θα αναλυθεί στη συνέχεια.

Επιπλέον, κρίνεται σωστό να αναφερθεί πως οι υποκατηγορίες του deep learning είναι όμοιες με αυτές που αναφέρθηκαν παραπάνω για το machine learning.

3.3.2 Υπέρ και Κατά του Deep Learning

Πλεονεκτήματα

- Δυνατότητα δημιουργίας νέων χαρακτηριστικών από τα περιορισμένα διαθέσιμα σύνολα δεδομένων εκπαίδευσης

- Μπορεί να εργαστεί σε τεχνικές εκμάθησης χωρίς επίβλεψη με αποτέλεσμα να βοηθά στη δημιουργία ενεργών και αξιόπιστων αποτελεσμάτων

- Με τη συνεχή εκπαίδευση, η αρχιτεκτονική του έχει γίνει προσαρμοστική στις αλλαγές και είναι σε θέση να εργάζεται σε διάφορα προβλήματα

- Μειώνει τον χρόνο που απαιτείται για την επίλυση προβλημάτων

Μειονεκτήματα

- Η πλήρης εκπαιδευτική διαδικασία βασίζεται στη συνεχή ροή των δεδομένων, η οποία μειώνει τα περιθώρια βελτίωσης της εκπαιδευτικής διαδικασίας

- Έλλειψη διαφάνειας στην αναθεώρηση σφαλμάτων. Δεν υπάρχουν ενδιάμεσα βήματα για τον προσδιορισμό συγκεκριμένου σφάλματος. Για να επιλυθεί το πρόβλημα, αναθεωρείται ένας πλήρης αλγόριθμος.

- Ανάγκη για ακριβούς πόρους, μονάδες επεξεργασίας υψηλής ταχύτητας και ισχυρές GPU για την εκπαίδευση πολύπλοκων μοντέλων.

3.3.3 Εφαρμογές του Deep Learning

- Εικονικοί Βοηθοί:

Η βασική λειτουργία που εξυπηρετεί τη μετάφραση της ομιλίας και της γλώσσας της ομιλίας του ανθρώπου είναι το Deep Learning. Κοινά παραδείγματα εικονικών βοηθών είναι οι Cortana, Siri και Alexa

- Chat Bots

Η συνεχής αλληλεπίδραση των Chat Bots με τους ανθρώπους για την παροχή υπηρεσιών πχ σε πελάτες απαιτεί αληθοφανείς και σωστές απαντήσεις. Προκειμένου να δοθούν οι απαντήσεις

αυτές απαιτούνται αλγόριθμοι deep learning.

- Μεταφράσεις:

Η αυτόματη μετάφραση της ομιλίας σε πολλές γλώσσες απαιτεί εφαρμογή Deep Learning. Πρόκειται για έναν χρήσιμο μηχανισμό για τουρίστες, ταξιδιώτες αλλά και κρατικούς αξιωματούχους.

- Αναγνώριση προσώπου

Η αναγνώριση προσώπου έχει πολλά χαρακτηριστικά, από τη χρήση στην ασφάλεια έως τη δυνατότητα προσθήκης ετικετών που χρησιμοποιούνται στα social media.Επίσης μπορεί να χρησιμοποιηθεί ακόμα και για ταυτοποίηση προσώπου σε περίπτωση που αλλάξουν συγκεκριμένα χαρακτηριστικά του(πχ βάρος, χτένισμα).

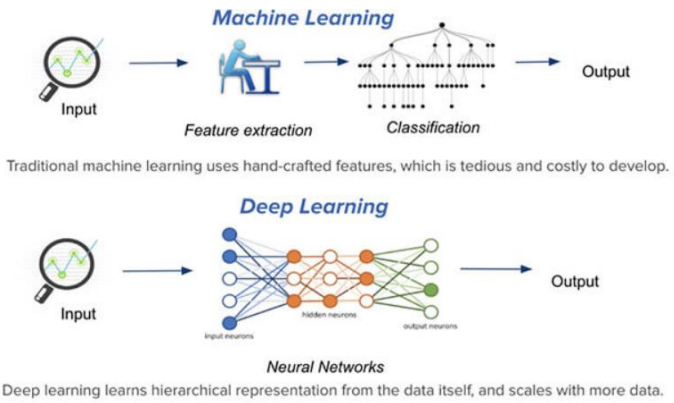
- Αυτόνομα οχήματα

Για να κατανοήσουμε τα σενάρια των δρόμων, τη λειτουργία των σημάτων, τους πεζούς, τη σημασία των διαφορετικών σημάτων, τα όρια ταχύτητας και πολλές άλλες καταστάσεις όπως αυτές, απαιτείται μεγάλος όγκος πραγματικών δεδομένων. Με την πληθώρα δεδομένων, η αποτελεσματικότητα των αλγορίθμων βελτιώνεται, γεγονός που αυξάνει τη ροή λήψης αποφάσεων.[8]

3.4 Διαφορές μεταξύ Deep Learning και Machine Learning

Η κύρια διαφορά μεταξύ Machine Learning και Deep Learning αφορά στον τρόπο με τον οποίο δομούνται οι εφαρμογές του Deep Learning. Στο Machine Learning υπάρχει ένα επίπεδο δεδομένων ενώ στο Deep Learning τα επίπεδα είναι πολυάριθμα (εικόνα 2).

Στο Deep Learning, οι πολυεπίπεδες αναπαραστάσεις μαθαίνονται από μοντέλα που ονομάζονται νευρωνικά δίκτυα, δομημένα σε στρώματα τα οποία βρίσκονται στοιβαγμένα το ένα πάνω στο άλλο. Ο όρος νευρωνικό δίκτυο αποτελεί αναφορά στη νευροβιολογία, αλλά παρόλο που μερικές από τις κεντρικές έννοιες του Deep Learning αναπτύχθηκαν χάρη στην έμπνευση από τον τρόπο λειτουργίας του εγκεφάλου, τα αντίστοιχα μοντέλα δε μοιάζουν σε τίποτα σε σχέση με αυτά των εγκεφάλων. Δεν υπάρχουν αποδείξεις άλλωστε πως ο εγκέφαλος λειτουργεί με αντίστοιχο τρόπο όπως τα μοντέλα Deep Learning. Αναλυτικότερα μπορούμε να ισχυριστούμε πως το Deep Learning είναι ένα μαθηματικό αντικείμενο για τη μάθηση παραστάσεων από δεδομένα αλλά και από τις σχέσεις μεταξύ αυτών.[9]



Εικόνα 2

Κεφάλαιο 4

4.1 Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (ANN), τα οποία καλούνται αλλιώς απλά νευρωνικά δίκτυα (NN), είναι υπολογιστικά συστήματα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα τα οποία πιο απλά μπορούμε να θέσουμε ως εγκεφάλους έμβιων οργανισμών.

Αναλυτικότερα, ένα τεχνητό νευρωνικό δίκτυο απαρτίζεται από ένα σύνολο συνδεδεμένων κόμβων οι οποίοι ονομάζονται νευρώνες. Κάθε τέτοια σύνδεση, δίνει τη δυνατότητα σε κάθε τέτοιο "νευρώνα" να μεταδώσει ένα σήμα σε άλλους "νευρώνες" με το οποίους είναι συνδεδεμένος. Αυτό άλλωστε αποτελεί και τη βασική ιδέα του παραλληλισμού του νευρωνικού δικτύου με τις συνάψεις ενός εγκεφάλου. Έτσι λοιπόν ένας τεχνητός νευρώνας λαμβάνει ένα σήμα, το επεξεργάζεται και στη συνέχεια σηματοδοτεί τους νευρώνες που συνδέονται άμεσα με αυτόν. Το "σήμα" σε μια σύνδεση είναι ένας πραγματικός αριθμός και η έξοδος κάθε νευρώνα υπολογίζεται από κάποια μη γραμμική συνάρτηση του αθροίσματος των εισόδων του. Οι συνδέσεις ονομάζονται ακμές. Οι νευρώνες και τα άκρα έχουν συνήθως ένα βάρος που προσαρμόζεται καθώς προχωρά η μάθηση. Το βάρος αυξάνει ή μειώνει την ισχύ του σήματος σε μια σύνδεση. Οι νευρώνες μπορεί να έχουν ένα κατώφλι τέτοιο ώστε ένα σήμα να αποστέλλεται μόνο εάν το αθροιστικό σήμα υπερβαίνει αυτό το κατώφλι. Τυπικά, οι νευρώνες συγκεντρώνονται σε στρώματα. Διαφορετικά επίπεδα μπορεί να εκτελούν διαφορετικούς μετασχηματισμούς στις εισόδους τους. Τα σήματα ταξιδεύουν από το πρώτο στρώμα (το επίπεδο εισόδου), στο τελευταίο στρώμα (το επίπεδο εξόδου), αφού διασχίσουν τα επίπεδα πολλές φορές.[10]

Συνοψιστικά μπορούμε να ισχυριστούμε πως ένα τεχνητό νευρωνικό δίκτυο αποτελεί στην πραγματικότητα ένα κύκλωμα διασυνδεδεμένων κόμβων , σκοπό του οποίου αποτελεί η πρόβλεψη των σωστών εξόδων δεδομένων των εισόδων. Ωστόσο, προκειμένου το δίκτυο να αποβεί όσο το δυνατόν πιο αποτελεσματικό στην αντιμετώπιση του οποιουδήποτε προβλήματος, η εκπαίδευσή του αποτελεί το σημαντικότερο πεδίο ανάλυσης.

Η διαδικασία της μάθησης, επιτελείται μέσω ορισμένων δειγμάτων (Training Data), αλλά και μέσω συγκεκριμένου αλγορίθμου εκπαίδευσης οπίσθιας διάδοσης(Backpropagation Algorithm), ο οποίος στοχεύει στη βελτίωση της απόδοσης του δικτύου. Ουσιαστικά πρόκειται για μια επαναληπτική διαδικασία κατά την οποία κάθε φορά οι ελεύθερες παράμετροι του νευρωνικού δικτύου μεταβάλλονται ούτως ώστε σταδιακά να μειωθεί το σφάλμα μεταξύ των επιθυμητών και των εξαγόμενων αποτελεσμάτων.[9]

4.2 Χαρακτηριστικά Νευρωνικών Δικτύων

4.2.1 Τρόποι Μάθησης Νευρωνικού Δικτύου

Οι τρόποι μάθησης διακρίνονται σε:

- Supervised Learning (Εποπτευόμενη Μάθηση)

Η εποπτευόμενη μάθηση (SL) είναι η εργασία μηχανικής μάθησης για την εκμάθηση μιας συνάρτησης από ένα νευρωνικό δίκτυο που αντιστοιχίζει μια είσοδο σε μια έξοδο με βάση παραδείγματα ζευγών εισόδου-εξόδου. Συνάγει ουσιαστικά μια συνάρτηση από δεδομένα εκπαίδευσης.[11]

- Unsupervised Learning (Μάθηση Χωρίς Επίβλεψη)

Η μάθηση χωρίς επίβλεψη είναι μια τεχνική μηχανικής μάθησης κατά την οποία το μοντέλο εργάζεται μόνο του προκειμένου να ανακαλύψει μοτίβα και πληροφορίες χρησιμοποιώντας δεδομένα τα οποία δεν είναι κατηγοριοποιημένα.[86]

- Reinforcement Learning (Ενισχυτική Μάθηση)

Η ενισχυτική μάθηση (RL) είναι ένας τομέας μηχανικής μάθησης που ασχολείται με το πώς οι ευφυείς πράκτορες πρέπει να αναλαμβάνουν ενέργειες σε ένα περιβάλλον προκειμένου να μεγιστοποιήσουν την έννοια της αθροιστικής ανταμοιβής.[87]

4.2.2 Ικανότητα Γενίκευσης

Επιπλέον, αξίζει να σημειωθεί πως κάθε νευρωνικό δίκτυο θα πρέπει να διαθέτει την ικανότητα της γενίκευσης, δεδομένου του ότι θα πρέπει να υπολογίζει σωστά τα αποτελέσματα ακόμα και μετά τη διαδικασία της εκπαίδευσης. Όταν δηλαδή "τροφοδοτηθεί" με διαφορετικά δεδομένα εισόδου (test data) από αυτά που χρησιμοποιήθηκαν κατά την εκπαίδευσή του(train data) τα αποτελέσματα θα πρέπει να είναι ανάλογα.[9]

4.3 Πλεονεκτήματα και Μειονεκτήματα Νευρωνικών Δικτύων

4.3.1 Πλεονεκτήματα

- Αποθήκευση πληροφοριών σε ολόκληρο το δίκτυο:

Η πληροφορία αποθηκεύεται στο σύνολο του δικτύου και όχι σε βάση δεδομένων, ενώ η εξαφάνιση μέρους των πληροφοριών σε ορισμένα σημεία δεν εμποδίζει το δίκτυο από την εύρυθμη λειτουργία του.

- Δυνατότητα εργασίας με ελλιπείς γνώσεις:

Μετά την εκπαίδευση του νευρωνικού δικτύου, είναι εφικτό να παραχθούν αποτελέσματα ακόμη και με ελλιπείς πληροφορίες. Η απώλεια απόδοσης εδώ εξαρτάται από τη σημασία των πληροφοριών που απουσιάζουν

- Ανοχή σφαλμάτων:

Καταστροφή ενός ή περισσότερων νευρώνων του νευρωνικού δικτύου δεν το εμποδίζει να παράγει έξοδο. Αυτό το χαρακτηριστικό καθιστά το δίκτυο αποτελεσματικότερο.

- Δυνατότητα παράλληλης επεξεργασίας:

Τα τεχνητά νευρωνικά δίκτυα διαθέτουν υπολογιστική ισχύ με αποτέλεσμα να είναι ικανά να εκτελούν περισσότερες από μία διεργασίες ταυτόχρονα.

4.3.2 Μειονεκτήματα

- Εξάρτηση υλικού:

Τα τεχνητά νευρωνικά δίκτυα απαιτούν επεξεργαστές με μεγάλη επεξεργαστική ισχύ, ανάλογα με τη δομή τους. Για το λόγο αυτό, η υλοποίησή τους είναι εξαρτημένη από τον εξοπλισμό που χρησιμοποιούμε.

- Προσδιορισμός σωστής δομής δικτύου:

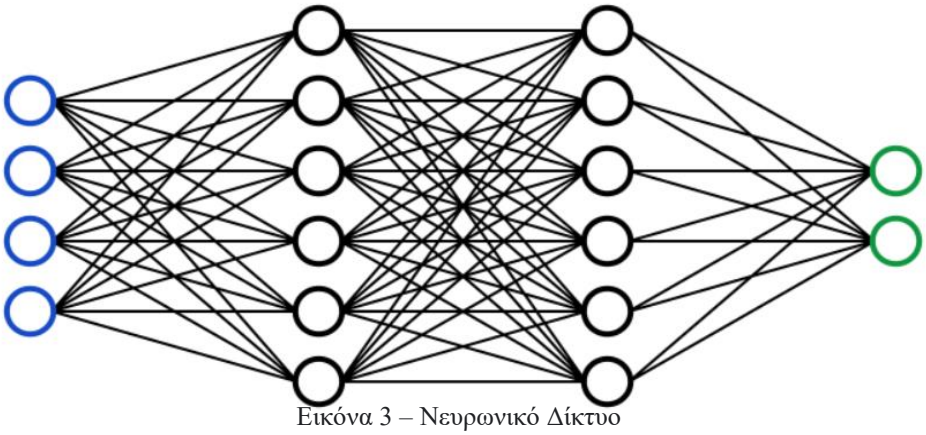
Δεν υπάρχει κανένας ειδικός κανόνας για τον προσδιορισμό της δομής του τεχνητού νευρικού δικτύου.

- Δυσκολία υπόδειξης του προβλήματος στο νευρωνικό δίκτυο.

Τα νευρωνικά δίκτυα είναι ικανά να λειτουργήσουν μόνο με αριθμητικές πληροφορίες. Συνεπώς οποιοδήποτε πρόβλημα πρέπει αρχικά να μεταφραστεί σε αριθμητικές τιμές, τις οποίες θα χρησιμοποιήσει στη συνέχεια το νευρωνικό δίκτυο. Ο τρόπος με τον οποίο θα χρησιμοποιηθεί η πληροφορία σε αυτήν την περίπτωση, επηρεάζει άμεσα την απόδοση του δικτύου. Συνεπώς συμπεραίνουμε πως κάτι τέτοιο έγκειται στην ικανότητα του χρήστη.

- Η διάρκεια της διαδικασίας εκπαίδευσης του δικτύου είναι άγνωστη

Το νευρωνικό δίκτυο θα σταματήσει μόλις η τιμή του σφάλματος φτάσει σε ένα επιθυμητό επίπεδο. Κάτι τέτοιο καθορίζεται από το χρήστη, με αποτέλεσμα η επιλεγόμενη τιμή να μη δίνει πάντα τα βέλτιστα αποτελέσματα ως προς την προσέγγιση των στόχων.[12]



Εικόνα 3 – Νευρωνικό Δίκτυο

4.4 Ανατομία Νευρωνικού Δικτύου (Επιμέρους Στοιχεία)

Στη συνέχεια θα αναλυθούν επιμέρους στοιχεία των νευρωνικών δικτύων, πέρα από τους νευρώνες, η λειτουργία των οποίων αναλύθηκε επαρκώς προηγουμένως(Κεφάλαιο 4.1).

4.4.1 Layers (Στρώματα Νευρωνικού Δικτύου)

Οι στρώσεις ενός νευρωνικού δικτύου αποτελούν ίσως το σημαντικότερο στοιχείο του. Ουσιαστικά κάθε νευρωνικό δίκτυο αποτελείται από τρεις τύπους στρώσεων/επιπέδων:

- Επίπεδο Εισόδου:
Δέχεται τα αρχικά δεδομένα με τα οποία τροφοδοτείται το νευρωνικό δίκτυο, κάποιους πίνακες δηλαδή(tensors).

- Κρυφά Επίπεδα:
Ενδιάμεσα επίπεδα μεταξύ του επιπέδου εισόδου και εξόδου όπου γίνονται όλοι οι επιμέρους υπολογισμοί.

- Επίπεδο Εξόδου:
Επίπεδο όπου παράγονται τα αποτελέσματα για τις εισόδους που δόθηκαν.[13]

Αναλόγως με τη φύση του εξεταζόμενου προβλήματος και των δεδομένων τα οποία το χαρακτηρίζουν, χρησιμοποιούμε διαφορετικού τύπου στρώσεις (layers). Για απλούστερα δεδομένα χρησιμοποιούμε δισδιάστατους πίνακες τα οποία ονομάζονται fully connected layers, για αλληλουχίες δεδομένων χρησιμοποιούμε τρισδιάστατους πίνακες η αλλιώς reccurent layers ενώ για εικόνες χρησιμοποιούμε τετραδιάστατους πίνακες η αλλιώς convolutional layers.[9]

Στην περίπτωσή μας, χρησιμοποιούνται δισδιάστατοι πίνακες (fully connected layers) αν και αυτό θα αναλυθεί εκτενέστερα σε επόμενο κεφάλαιο.

4.4.2 Bias-weights

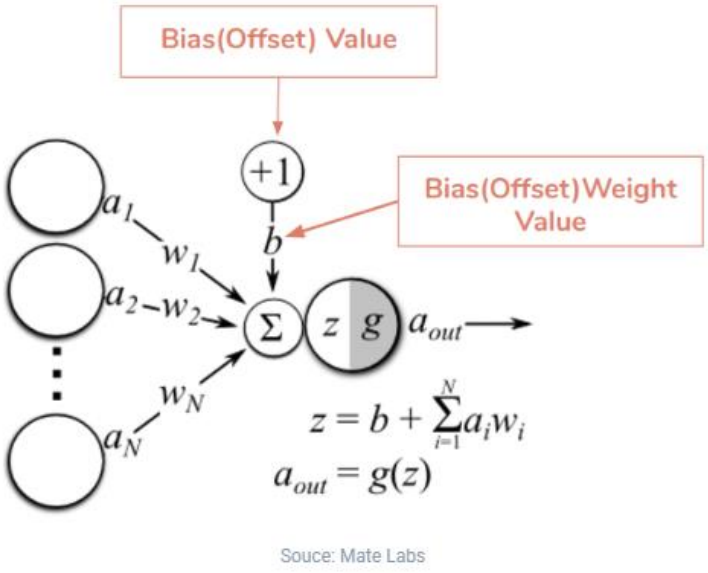
Τα βάρη και τα biases (κοινώς αναφέρονται ως W και b) αποτελούν παραμέτρους εκμάθησης των νευρωνικών δικτύων. Αναλυτικότερα, γνωρίζουμε πως οι νευρώνες είναι οι βασικές μονάδες ενός νευρωνικού δικτύου και κάθε ένας από αυτούς ο οποίος βρίσκεται σε ένα συγκεκριμένο στρώμα συνδέεται με κάποιους άλλους ή με όλους τους υπόλοιπους νευρώνες στο επόμενο στρώμα. Έτσι, όταν οι είσοδοι μεταδίδονται μεταξύ των νευρώνων ομοίως εφαρμόζονται σε αυτούς τα βάρη και τα biases. Παρατίθεται η σχέση εισόδου-εξόδου για κάθε «νευρώνα»:

$$Y = \sum(W * input) + b \qquad (1)$$

Όπου: Y έξοδος του εκάστοτε «νευρώνα», W το εκάστοτε βάρος, input η ανάλογη είσοδος του εκάστοτε «νευρώνα» και b το bias.

Τα βάρη ελέγχουν το σήμα(ή αλλιώς την ισχύ της σύνδεσης) μεταξύ δύο νευρώνων. Με άλλα λόγια φέρουν την ευθύνη της απόφασης σχετικά με το πόσο θα επηρεάσει η είσοδος την αντίστοιχη έξοδο.

Τα biases, τα οποία είναι σταθερά σε αντίθεση με τα βάρη, είναι μια πρόσθετη είσοδος στο επόμενο επίπεδο που πάντα θα έχει τιμή 1 ή 0.Παραμένουν ανεπηρέαστα σε σχέση με το προηγούμενο επίπεδο, (δεν έχουν εισερχόμενες συνδέσεις) ωστόσο διαθέτουν εξερχόμενες. Εν ολίγοις, ακόμα και αν όλες οι είσοδοι είναι μηδενικές χάρη στα biases κάποιοι νευρώνες θα ενεργοποιούνται(ανάλογα με τις ανάγκες του εκάστοτε προβλήματος).[14]



Εικόνα 4 – Σχήμα Εισόδου- Εξόδου Νευρώνα

4.4.3 Συναρτήσεις Ενεργοποίησης (Activation Functions)

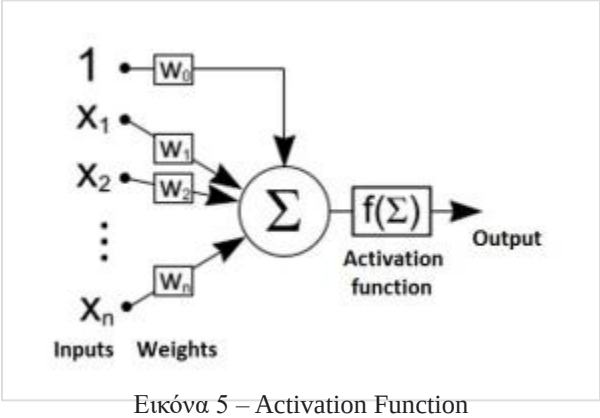
4.4.3.1 Ανάλυση

Οι συναρτήσεις ενεργοποίησης χρησιμοποιούνται προκειμένου να μετατρέπουν την ενεργοποίηση ενός νευρώνα σε σήμα εξόδου ενώ υπάρχουν αρκετές, οι οποίες χρησιμοποιούνται κάθε φορά ανάλογα με τη φύση του προβλήματος που επιλύεται και θα αναλυθούν εν συνεχεία.

Κάθε μοντέλο νευρώνα, αποτελείται από έναν πυρήνα επεξεργασίας στον οποίο καταλήγουν συνδέσεις εισόδων ενώ η έξοδος είναι μία και ενιαία. Η ροή σήματος των εισόδων νευρώνων, xi, θεωρείται μονής κατεύθυνσης και το σήμα εξόδου δίνεται από τη σχέση:

$$O = f(\sum) \qquad (2)$$

όπως φαίνεται και στο παρακάτω σχήμα:[15]



Εικόνα 5 – Activation Function

4.4.3.2 Γνωστές Συναρτήσεις Ενεργοποίησης

- ReLu function:

Εφαρμόζει τη λειτουργία ενεργοποίησης διορθωμένης γραμμικής μονάδας και επιστρέφει για κάθε στοιχείο το μέγιστο ανάμεσα σε αυτό και το μηδέν. Επιπλέον με κατάλληλη τροποποίηση ορισμένων παραμέτρων, επιτρέπει τη χρήση μη μηδενικών ορίων αλλά και τη μεταβολή της μέγιστης τιμής ενεργοποίησης

$$\text{ReLu}(x) = \max(0, x) \quad (3)$$

- Sigmoid function:

Πρόκειται για τη σιγμοειδή συνάρτηση διέγερσης η οποία για μικρές τιμές επιστρέφει μια τιμή κοντά στο μηδέν ενώ για μεγάλες τιμές το αποτέλεσμα της συνάρτησης πλησιάζει το ένα. Είναι ουσιαστικά ισοδύναμη με μια SoftMax δύο στοιχείων όπου το δεύτερο στοιχείο θεωρείται μηδέν. Ουσιαστικά επιστρέφει πάντα μία τιμή μεταξύ μηδέν και ένα.

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}} \quad (4)$$

- SoftMax function:

Η SoftMax μετατρέπει ένα διάνυσμα τιμών σε κατανομή πιθανότητας. Τα στοιχεία του διανύσματος εξόδου βρίσκονται στην περιοχή από μηδέν ως ένα και το συνολικό τους άθροισμα ισούται με ένα.Κάθε διάνυσμα αντιμετωπίζεται ανεξάρτητα, ενώ κυρίως χρησιμοποιείται ως συνάρτηση ενεργοποίησης στο τελευταίο επίπεδο νευρωνικών δικτύων ταξινόμησης καθώς το αποτέλεσμά της ερμηνεύεται ως κατανομή πιθανότητας. Οι τιμές εισόδου είναι οι πιθανότητες καταγραφής της προκύπτουσας πιθανότητας.

$$\text{SoftMax}(x) = \frac{e^x}{\Sigma e^x} \quad (5)$$

- Tanh function:

Όσο μεγαλύτερη είναι η είσοδος, τόσο πιο κοντά στο ένα θα είναι η τιμή εξόδου. Αντίστοιχα όσο πιο μικρή είναι η τιμή εξόδου τόσο πιο κοντά στο μείον ένα θα είναι και η έξοδος.[16]

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (6)$$

Τέλος σωστό κρίνεται να αναφερθεί πως μολονότι η επιλογή της σωστής συνάρτησης διέγερσης διαδραματίζει σημαντικό ρόλο στην αποτελεσματικότητα ενός νευρωνικού δικτύου, άλλες παράμετροι όπως ο αλγόριθμος εκπαίδευσης ή ακόμα και το μέγεθος του δικτύου είναι πιο ουσιαστικής σημασίας και έχουν μεγαλύτερο αντίκτυπο.[15]

4.4.4 Συνάρτηση Απώλειας (Loss Function)

Η συνάρτηση απώλειας επιφορτίζεται με το ρόλο της ανατροφοδότησης του μοντέλου του εκάστοτε νευρωνικού δικτύου. ΄Ουσιαστικά, σκοπός της είναι η μέτρηση της διαφοράς μεταξύ των εξαγόμενων αποτελεσμάτων και των πραγματικών στόχων που διαθέτουμε. Συμπεραίνουμε λοιπόν πως εξίσου σημαντικό ρόλο παίζει και στην αποτελεσματικότητα του νευρωνικού δικτύου αφού, η επιλογή της υπονοεί και μια συγκεκριμένη τιμή σφάλματος.[17]

4.4.5 Αλγόριθμοι Εκπαίδευσης (Training Algorithms)

Από την ανάπτυξη της μεθόδου οπίσθιας διάδοσης και μετά, αρκετοί τροποποιημένοι αλλά και νέοι αλγόριθμοι έχουν κάνει την εμφάνισή τους σε ότι αφορά στην εκπαίδευση των νευρωνικών δικτύων ενώ πολλοί από αυτούς έχουν πολύ γρήγορο ρυθμό σύγκλισης εφόσον το δίκτυο έχει ένα λογικό μέγεθος. Ωστόσο σε περιπτώσεις όπου τα νευρωνικά δίκτυα γίνονται μεγαλύτερης κλίμακας τότε και οι επιλογές των αλγορίθμων εκπαίδευσης μειώνονται δραματικά. Από την άλλη δεδομένης της έρευνας που διενεργείται συνεχώς, όλο και περισσότερα πολυστρωματικά δίκτυα με μεγάλο αριθμό βαρών γνωρίζουν ολοένα και περισσότερες επιλογές αναφορικά με την εκπαίδευσή τους(μέθοδοι στοχαστικής βελτιστοποίησης).

Οποιαδήποτε γραμμική ή μη γραμμική μέθοδος βελτιστοποίησης, είτε τοπική είτε καθολική, μπορεί να εφαρμοστεί στο πλαίσιο βελτιστοποίησης των βαρών κάποιου δικτύου. Προφανώς, τοπικές αναζητήσεις περιορίζονται θεμελιωδώς σε τοπικές λύσεις, ενώ οι καθολικές επιχειρούν να αποφύγουν αυτόν τον περιορισμό. Ουσιστικός στόχος ωστόσο, είναι η ελαχιστοποίση της συνάρτησης λάθους που αναλύθηκε παραπάνω. Η απόδοση της εκπαίδευσης ποικίλλει αναλόγως και έχει να κάνει τόσο με τη συνάρτηση σφάλματος για ένα δεδομένο πρόβλημα όσο και με τη διαμόρφωση του δικτύου. Βασιζόμενοι λοιπόν στο γεγονός πως για πολλά παραδείγματα νευρωνικών δικτύων έχουμε πλήρη γνώση των πληροφοριών της καμπύλης λάθους αλλά και της διαμόρφωσής τους, γνωρίζουμε επίσης και τους πιο δημοφιλείς αλγορίθμους εκπαίδευσης οι οποία στην πραγματικότητα βασίζονται σε παραλλαγές του αλγορίθμου οπίσθιας διάδοσης. Συνοπτικά αναφέρονται η αντίστροφη διάδοση με χρήση της προσέγγισης συζυγούς-κλίσης(backpropagation using the conjugate-gradient approach), η κλίμακα συζυγούς-κλίσης και το στοχαστικό του αντίστοιχο(scaled conjugate-gradient and its stochastic counterpart) και ο αλγόριθμος Levenberg-Marquadt, ο οποίος χρησιμοποιείται στην περίπτωσή μας.

Σε πολλές μελέτες, παρουσιάζονται πειράματα με νευρωνικά δίκτυα μικρής κλίμακας και ως εκ τούτου γίνεται συχνή αναφορά στη μέθοδο Levenberg-Marquadt η οποία χρησιμοποιείται και στην εν λόγω εργασία και πρόκειται να αναλυθεί εκτενέστερα παρακάτω. Γενικά, είναι ένας αλγόριθμος εκπαίδευσης εξαιρετικά γρήγορος για μικρά δίκτυα, με αποτέλεσμα να αποτελεί μακράν την καλύτερη επιλογή σε θέματα αποτελεσματικότητας. Αντιθέτως για μεγαλύτερης κλίμακας δίκτυα τα οποία μπορεί και να απαρτίζονται από χιλιάδες νευρώνες η προαναφερθείσα μέθοδος απαιτεί τεράστιο υπολογιστικό κόστος και χρόνο εξ αιτίας της πολυπλοκότητάς της.[18]

4.5 Προβλήματα Νευρωνικών Δικτύων και Αντιμετώπιση

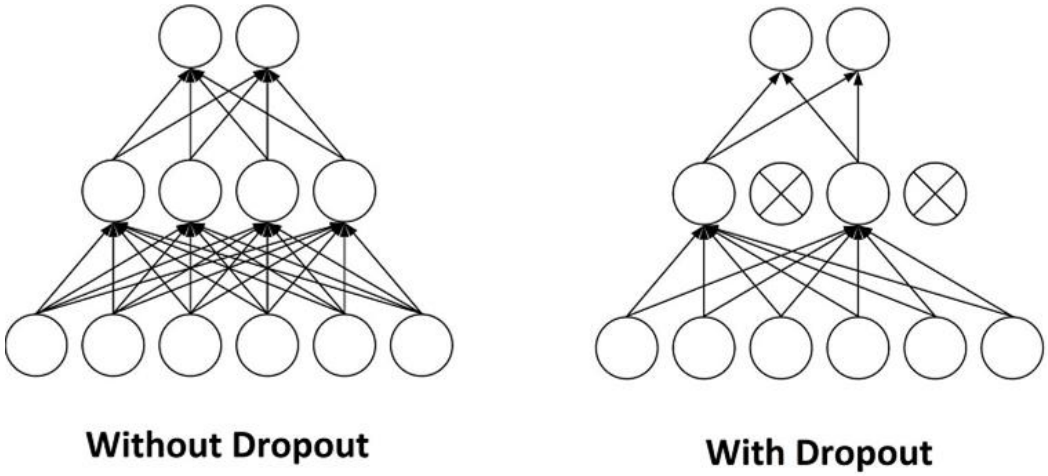
4.5.1 Overfitting

Εκτός όσων αναλύθηκαν παραπάνω, ένας ακόμα στόχος του αλγόριθμου εκμάθησης είναι η κατανόηση συγκεκριμένων μοτίβων και σχέσεων μεταξύ των δεδομένων εισόδου και εξόδου. Από την άλλη όμως υπάρχει ο κίνδυνος να φτάσουμε σε σημείο το μοντέλο να καταλήξει να "θυμάται" τις σχέσεις αυτές οπότε και παρατηρείται το φαινόμενο του overfitting. Κάτι τέτοιο μπορεί να αυξάνει την ακρίβεια ,ωστόσο κάνει το μοντέλο πολύ αναξιόπιστο σε περίπτωση που χρειαστεί να διαχειριστεί διαφορετικά Datasets.[18]

Το overfitting, ή αλλιώς κορεσμός παρατηρείται όταν το αποτέλεσμα μίας ανάλυσης είναι προσαρμοσμένο σε ένα συγκεκριμένα σύνολο δεδομένων (Dataset) με αποτέλεσμα το μοντέλο να μην μπορεί να προσαρμοστεί σε νέα δεδομένα ή να προβλέψει νέες μετρήσεις.[9]

4.5.2 Dropout

Απάντηση στον κορεσμό αποτελεί μια τεχνική που ονομάζεται dropout, για την οποία έχει αποδειχθεί πως βελτιώνει σημαντικά την απόδοση των νευρωνικών δικτύων. Η τεχνική dropout λειτουργεί ορίζοντας τα εξερχόμενα άκρα των κρυφών μονάδων(νευρώνες που αποτελούν κρυφά στρώματα) στο μηδέν σε κάθε ενημέρωση της φάσης εκπαίδευσης. Η πιθανότητα βάση της οποίας συντελείται κάτι τέτοιο εξαρτάται από το χρήστη. Κάθε παράδειγμα εκπαίδευσης μπορεί έτσι να θεωρηθεί ως παροχή διαβαθμίσεων για μια διαφορετική, τυχαία δειγματοληπτική αρχιτεκτονική, έτσι ώστε το τελικό νευρωνικό δίκτυο να αντιπροσωπεύει αποτελεσματικά ένα τεράστιο σύνολο νευρωνικών δικτύων, με καλή ικανότητα γενίκευσης.[19]



Εικόνα 6

4.6 Data Set

Με τον όρο Data Set ή σύνολο δεδομένων εννοούμε το πλήθος των δεδομένων που χρησιμοποιούμε για την εκπαίδευση και το έλεγχο της αξιοπιστίας ενός νευρωνικού δικτύου.

4.6.1 Διάκριση Δεδομένων Dataset

Ανάλογα με τη λειτουργία που επιτελούν τα δεδομένα διαχωρίζονται σε τρεις επιμέρους κατηγορίες:

- Δεδομένα εκπαίδευσης - Training Data:

Τα δεδομένα αυτά επηρεάζουν στην ουσία τις παραμέτρους του αλγορίθμου μηχανικής μάθησης(πχ βάρη). Ο χρήστης τροφοδοτεί τα δεδομένα εισόδου του αλγόριθμου, τα οποία αντιστοιχούν σε μια αναμενόμενη έξοδο. Το μοντέλο αξιολογεί τα δεδομένα επανειλημμένα προκειμένου να «κατανοήσει» τις συσχετίσεις μεταξύ τους και στη συνέχεια προσαρμόζεται για να εξυπηρετήσει τον επιδιωκόμενο σκοπό του.

- Δεδομένα Ελέγχου - Testing Data:

Μετά το πέρας της εκπαίδευσης του μοντέλου, τα δεδομένα δοκιμών επικυρώνουν εκ νέου την ακρίβεια των προβλέψεών του. Επίσης σε περίπτωση που τα training και validation data έχουν ετικέτες, τα testing data θα πρέπει να είναι χωρίς ετικέτα. Εν ολίγοις, παρέχουν έναν τελικό, πραγματικό έλεγχο προκειμένου να επιβεβαιωθεί πως ο αλγόριθμος εκπαιδεύτηκε αποτελεσματικά.

- Δεδομένα επικύρωσης - Validation Data:

Κατά τη διάρκεια της εκπαίδευσης, τροφοδοτούνται στο μοντέλο νέα δεδομένα με τα οποία δεν είχε έρθει σε επαφή νωρίτερα. Τα δεδομένα αυτά, παρέχουν την πρώτη δοκιμή έναντι μη ορατών δεδομένων, επιτρέποντας την αξιολόγηση του μοντέλου σε σχέση με την πραγματοποίηση προβλέψεων με βάση απολύτως νέα δεδομένα. Η χρήση τους δεν είναι αναγκαστική ωστόσο η συνεισφορά τους έγκειται στο γεγονός πως είναι υπεύθυνα για τη ρύθμιση των υπερπαραμέρων του μοντέλου οι οποίες επηρεάζουν τον τρόπο με τον οποίο το μοντέλο αξιολογεί τα δεδομένα.

Καθώς είναι προφανές πως καθένα από αυτά τα τρία σύνολα δεδομένων έχει τη θέση του στη δημιουργία και την εκπαίδευση μοντέλων μηχανικής μάθησης, εύκολα διακρίνεται μεταξύ τους κάποιου είδους επικάλυψη. Η διαφορά μεταξύ των δεδομένων εκπαίδευσης και των δεδομένων δοκιμής είναι σαφής: το ένα εκπαιδεύει ένα μοντέλο ενώ το άλλο επιβεβαιώνει ότι λειτουργεί σωστά. Ωστόσο, μπορεί να προκύψει σύγχυση μεταξύ των λειτουργικών ομοιοτήτων και διαφορών μεταξύ των προαναφερθέντων τύπων δεδομένων και των δεδομένων επικύρωσης.

4.6.2 Διαφορές ανάμεσα σε Training Data και Validation Data

Οι αλγόριθμοι μηχανικής μάθησης απαιτούν δεδομένα εκπαίδευσης προκειμένου να επιτύχουν ένα συγκεκριμένο στόχο. Έτσι, κατά τη διάρκεια μιας επαναληπτικής διαδικασίας, το εκάστοτε μοντέλο θα αναλύσει αυτό το σύνολο δεδομένων εκπαίδευσης και θα προσπαθήσει να κατανοήσει τις συσχετίσεις εντός αυτού. Εφόσον φτάσουμε στο πέρας της εκπαίδευσης, το μοντέλο μπορεί σταδιακά να απομνημονεύσει τις εισόδους και τις εξόδους του συνόλου των

δεδομένων. Αυτό ωστόσο αποτελεί πρόβλημα στην περίπτωση ελέγχου του δικτύου με διαφορετικά δεδομένα.

Εξ αιτίας αυτής της πιθανότητας λοιπόν είναι χρήσιμα τα δεδομένα επικύρωσης, τα οποία παρέχουν ένα πρώτο έλεγχο σχετικά με την αποτελεσματικότητα του μοντέλου υπό ρεαλιστικές συνθήκες, Οι αλγόριθμοι μηχανικής μάθησης μπορούν να αξιολογήσουν ταυτόχρονα τόσο τα δεδομένα εκπαίδευσης όσο και τα δεδομένα επικύρωσης.

Επιπλέον, τα δεδομένα επικύρωσης αποτελούν ένα εντελώς ξεχωριστό τμήμα δεδομένων παρόλο που είναι δυνατόν να προέρχονται από κατάτμηση των δεδομένων εκπαίδευσης. Φυσικά τα δεδομένα εκπαίδευσης και τα δεδομένα επικύρωσης παραμένουν αυστηρώς διαχωρισμένα καθ'όλη τη διάρκεια της εκπαίδευσης. Χάρη στην ακρίβεια των προβλέψεων μετά το στάδιο της επικύρωσης ο χρήστης έχει τη δυνατότητα να προσαρμόσει εκ νέου τις υπερπαραμέτρους του δικτύου όπως το ρυθμό εκμάθησης, τα κρυφά στρώματα ή τις συναρτήσεις ενεργοποίησης των νευρώνων. Κατ' αυτό τον τρόπο αποτρέπεται η υπερπροσαρμογή του μοντέλου στα αρχικά δεδομένα και συνεπώς η ακρίβεια του είναι εγγυημένη. Έτσι μπορούμε να είμαστε αρκετά σίγουροι πως το ίδιο μοντέλο θα έχει ανάλογη ακρίβεια ακόμα και με τα δεδομένα ελέγχου.

4.6.3 Διαφορές ανάμεσα σε Test Data και Validation Data

Βάση όσων προαναφέρθηκαν είναι προφανές πως τα δεδομένα επικύρωσης και τα δεδομένα ελέγχου εξυπηρετούν τον ίδιο στόχο. Επιβεβαιώνουν δηλαδή πως το μοντέλο μπορεί να παράξει ρεαλιστικές προβλέψεις. Ωστόσο, υπάρχουν ορισμένες πρακτικές διαφορές μεταξύ των δεδομένων επικύρωσης και των δεδομένων ελέγχου.
Αρχικά, το σύνολο των δεδομένων επικύρωσης είναι σημασμένο (labeled) με αποτέλεσμα να είναι εφικτή η εξαγωγή μετρήσεων που αποσκοπούν στη βελτιστοποίηση του μοντέλου. Υπό αυτή την έννοια, τα δεδομένα επικύρωσης εμφανίζονται ως μέρος της διαδικασίας εκπαίδευσης του μοντέλου. Αντίθετα, το μοντέλο λειτουργεί ως μαύρο κουτί όταν τροφοδοτείται με τα δεδομένα ελέγχου. Συνεπώς, τα δεδομένα επικύρωσης συντονίζουν επί της ουσίας το μοντέλο ενώ τα δεδομένα ελέγχου επιβεβαιώνουν πως λειτουργεί.

Από την άλλη υφίσταται κάποια σημασιολογική ασάφεια μεταξύ των δεδομένων επικύρωσης και των δεδομένων ελέγχου. Υπάρχουν αναφορές όπου τα δεδομένα ελέγχου αποκαλούνται σύνολα δεδομένων επικύρωσης. Τέλος, εφόσον υπάρχουν τρία σύνολα δεδομένων , τα δεδομένα επικύρωσης λειτουργούν ως μέσο ρύθμισης και βελτίωσης του αλγορίθμου μηχανικής μάθησης.[20]

4.6.4 Πλήθος Δεδομένων

Οι αλγόριθμοι μηχανικής μάθησης έχουν ανάγκη την πρόσληψη μιας λογικής ποσότητας δεδομένων προκειμένου να εξάγουν ακριβή αποτελέσματα. Ωστόσο, η έννοια της λογικής ποσότητας φαίνεται συγκεχυμένη καθώς εξαρτάται από την πολυπλοκότητα του ίδιου του αλγορίθμου αλλά και από τη φύση του προβλήματος. Η αξιοπιστία των δεδομένων και η κοινή χρήση τους είναι πολύ κρίσιμης σημασίας στη μηχανική μάθηση για τη βελτίωση της ακρίβειας των αποτελεσμάτων. Σε περιπτώσεις που τα δεδομένα είναι αρκετά πρέπει να γίνονται κατάλληλες περικοπές ενώ όταν είναι λιγότερα από όσα θα έπρεπε είναι σωστό να γίνεται συλλογή νέων δεδομένων για παράδειγμα μέσω κάποιας πειραματικής διαδικασίας ή μέσω

άλλης διαδικασίας προσομοίωσης.[21]

4.6.5 Επεξεργασία των Δεδομένων

Πολλοί αλγόριθμοι μηχανικής εκμάθησης αποδίδουν καλύτερα όταν οι αριθμητικές μεταβλητές εισόδου κλιμακώνονται σε ένα τυπικό εύρος. Οι δύο πιο δημοφιλείς τεχνικές για την κλιμάκωση των αριθμητικών δεδομένων πριν από την εκχώρησή τους στο μοντέλο μηχανικής μάθησης είναι η κανονικοποίηση(normalization) και η τυποποίηση(standardization). Η κανονικοποίηση κλιμακώνει κάθε μεταβλητή εισόδου ξεχωριστά στο εύρος 0-1, το οποίο είναι το εύρος τιμών κινητής υποδιαστολής όπου έχουμε τη μεγαλύτερη ακρίβεια. Η τυποποίηση κλιμακώνει κάθε μεταβλητή εισόδου χωριστά αφαιρώντας τον μέσο όρο (που ονομάζεται κεντράρισμα) και διαιρώντας με την τυπική απόκλιση για να μετατοπιστεί η κατανομή ώστε να έχει μέσο όρο μηδέν και τυπική απόκλιση ίση με ένα.

4.6.5.1 Data Normalization

Η κανονικοποίηση απαιτεί να γνωρίζουμε ή να βρισκόμαστε σε θέση να εκτιμήσουμε με ακρίβεια τις ελάχιστες και μέγιστες παρατηρήσιμες τιμές. Συνήθως η εκτίμηση αυτών των τιμών γίνεται από τα διαθέσιμα δεδομένα που διαθέτουμε[22].Τα χαρακτηριστικά συχνά κανονικοποιούνται ώστε να βρίσκονται σε ένα σταθερό εύρος (συνήθως από μηδέν ως ένα), διαιρώντας όλες τις τιμές με τη μέγιστη τιμή που συναντάται ή αφαιρώντας την ελάχιστη τιμή και διαιρώντας με το εύρος μεταξύ της μέγιστης και της ελάχιστης τιμής[23]. Μια τιμή κανονικοποιείται ως εξής:

$$y = \frac{x - min}{max - min} \qquad (7)$$

Οι ελάχιστες και μέγιστες τιμές αφορούν την τιμή x που κανονικοποιείται.

Ένας παράδειγμα τρόπου κανονικοποίησης δεδομένων είναι το αντικείμενο του Scikit-Learn που ονομάζεται MinMaxScaler. Μέσω του MinMaxScaler κάθε στοιχείο εισόδου κλιμακώνεται έτσι ώστε να βρίσκεται σε δεδομένο εύρος(μηδέν ως ένα). Αυτός ο μετασχηματισμός χρησιμοποιείται συχνά ως εναλλακτική λύση στη μηδενική μέση, μοναδιαία κλίμακα διακύμανσης[24].

4.6.5.2 Data Standardization

Η τυποποίηση ενός συνόλου δεδομένων περιλαμβάνει τη διαβάθμιση της κατανομής των τιμών έτσι ώστε ο μέσος όρος των παρατηρούμενων τιμών να είναι μηδέν και η τυπική απόκλιση να είναι ίση με ένα. Κάτι τέτοιο μπορεί να θεωρηθεί αφαίρεση της μέσης τιμής ή ακόμα και κεντράρισμα των δεδομένων. Όπως η κανονικοποίηση, η τυποποίηση μπορεί να είναι χρήσιμη, ακόμη και απαραίτητη σε ορισμένους αλγόριθμους μηχανικής εκμάθησης, ιδίως όταν τα δεδομένα έχουν τιμές εισόδου με διαφορετικές κλίμακες. Η τυποποίηση προϋποθέτει ότι τα δεδομένα ταιριάζουν σε μια κατανομή Gauss (καμπύλη καμπάνας) με ανάλογη μέση τιμή και τυπική απόκλιση. Από την άλλη, υπάρχει η επιλογή να γίνει τυποποίηση στα δεδομένα ακόμα και αν αυτές οι προϋποθέσεις δεν ικανοποιούνται, ωστόσο ενδέχεται τα αποτελέσματα να μην είναι αξιόπιστα.[22]

Μια άλλη τεχνική […] είναι ο υπολογισμός του στατιστικού μέσου όρου και της τυπικής απόκλισης των τιμών των χαρακτηριστικών, η αφαίρεση του μέσου όρου από κάθε τιμή και η διαίρεση του αποτελέσματος με την τυπική απόκλιση. Αυτή η διαδικασία ονομάζεται τυποποίηση μιας στατιστικής μεταβλητής και καταλήγει σε ένα σύνολο τιμών των οποίων ο μέσος όρος είναι μηδέν και η τυπική απόκλιση είναι ένα.[23]

Η τυποποίηση απαιτεί να γνώση της μέσης και τυπικής απόκλισης των παρατηρήσιμων τιμών, κάτι το οποίο μπορεί να εξαχθεί αρκετές φορές μόνο από τα δεδομένα εκπαίδευσης και όχι από ολόκληρο το σύνολο δεδομένων. Η αφαίρεση του μέσου όρου από τα δεδομένα ονομάζεται κεντράρισμα, ενώ η διαίρεση με την τυπική απόκλιση ονομάζεται κλιμάκωση. Ως εκ τούτου, η μέθοδος ονομάζεται και «κεντρική κλιμάκωση».[22] Μια τιμή τυποποιείται ως εξής:

$$y = \frac{x - mean}{standard\_deviation} \qquad (8)$$

Όπου ο μέσος όρος(mean) υπολογίζεται ως εξής:

$$mean = \frac{sum(x)}{count(x)} \qquad (9)$$

Και η τυπική απόκλιση(standrad deviation) υπολογίζεται ως εξής:

$$standard\ deviation = \sqrt{\Sigma \frac{(x - mean)^2}{count(x)}} \qquad (10)$$

Ένας παράδειγμα τρόπου τυποποίησης δεδομένων είναι το αντικείμενο του Scikit-Learn που ονομάζεται StandardScaler. Μέσω του StandardScaler κάθε στοιχείο εισόδου τυποποιείται καθώς αφαιρείται η μέση τιμή του και κλιμακώνεται η διακύμανση μονάδας.[25]

Τέλος κρίνεται σωστό να αναφερθεί πως δεν υπάρχει ουσιαστικά βέλτιστη μέθοδος σε σχέση με την επεξεργασία των δεδομένων που θα χρησιμοποιήσει οποιοδήποτε νευρωνικό δίκτυο. Ουσιαστικά, επιλέγεται μετά από μία διαδικασία δοκιμών και πειραματισμού συνδιαστικά και με την επιλογή των υπόλοιπων στοιχείων που το απαρτίζουν.

4.7 Είδη Νευρωνικών Δικτύων

Στη συνέχεια παρατίθενται οι τρεις βασικοί τύποι νευρωνικών δικτύων:

- Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks (ANN)) (Εικόνα 7)
- Νευρωνικά Δίκτυα Συνέλιξης(Convolution Neural Networks (CNN)) (Εικόνα 8)
- Επαναλαμβανόμενα νευρωνικά δίκτυα(Recurrent Neural Networks (RNN) (Εικόνα 9)

4.7.1 Artificial Neural Networks (ANN)

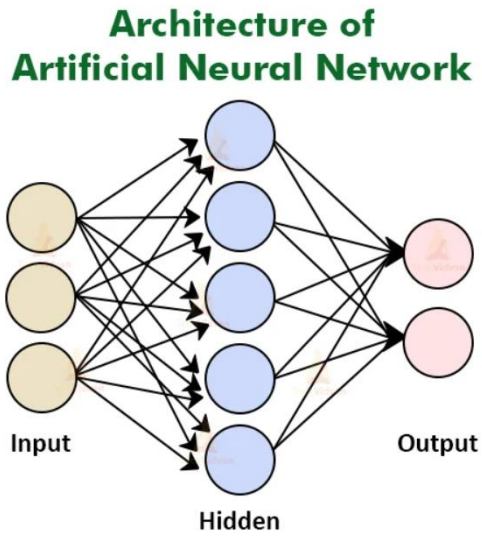
Οι περισσότεροι άνθρωποι θεωρούν τα τεχνητά νευρωνικά δίκτυα ως συνώνυμα με τα νευρωνικά δίκτυα. Ωστόσο, δεν είναι ένα και το αυτό. Αντιθέτως, ένα τεχνητό νευρωνικό δίκτυο αποτελεί μόνο έναν τύπο νευρωνικού δικτύου. Το ANN είναι ουσιαστικά ένα feedforward νευρωνικό δίκτυο λόγω του τρόπου με τον οποίο οι πληροφορίες ταξιδεύουν μέσω αυτού — από το ένα επίπεδο στο άλλο χωρίς να "αγγίζουν " τον ίδιο κόμβο δύο φορές.

Αυτός ο τύπος νευρωνικού δικτύου αναγνωρίζει μοτίβα σε ακατέργαστα δεδομένα, βοηθώντας στην επίλυση πολύπλοκων διαδικασιών. Επίσης δεδομένου ότι διαμορφώνεται ανάλογα με τον τρόπο που λειτουργούν οι νευρώνες του ανθρώπινου εγκεφάλου, βελτιώνεται με κάθε νέα είσοδο που λαμβάνει. Με λίγα λόγια, το ANN βελτιώνεται συνεχώς.

Αναφορικά με τη σύνθεση αυτού του τύπου δικτύου, βασίζεται σε τρία ή περισσότερα διασυνδεδεμένα στρώματα κόμβων. Όλα αυτά τα επίπεδα είναι υπεύθυνα για την εισαγωγή, την επεξεργασία και την έξοδο δεδομένων. Ένα τέτοιο πολυστωματικό σύστημα είναι αυτό που βοηθά αυτό το νευρωνικό δίκτυο να κατανοήσει και να μάθει περίπλοκους υπολογισμούς.

Λόγω της κλιμακωτής σύνθεσής του, το ANN χρησιμοποιείται στην τεχνολογία που εστιάζει στην επίλυση σύνθετων προβλημάτων, όπως προβλήματα αναγνώρισης προτύπων. Άλλα παραδείγματα χρήσης ενός ANN είναι η προγνωστική ανάλυση για επιχειρηματικούς σκοπούς, η πρόγνωση δεδομένων στον βιομηχανικό ποιοτικό έλεγχο ή ακόμα και μια απλή πρόγνωση καιρού.

Η σημαντικότητα των τεχνητών νευρωνικών δικτύων έγκειται στην προσαρμοστική τους φύση. Κάτι τέτοιο σημαίνει πως το νευρωνικό δίκτυο τροποποιεί τον εαυτό του μετά την αρχική εκπαίδευση εκμεταλλευόμενο της πληροφορίες που επεξεργάζεται. Επίσης λόγω της ισχύος τους τα ANN μπορούν να εκτελούν πολλαπλές εργασίες ταυτόχρονα. Μάλιστα είναι αρκετά ανθεκτικά στον σταδιακό εκφυλισμό αφού σε περίπτωση σφάλματος σε έναν ή σε μερικούς κόμβους στο ANN, δεν εμποδίζεται η διαδικασία εξαγωγής ορθών αποτελεσμάτων. Τέλος τα ANN διακρίνονται για την κατανεμημένη αποθήκευση πληροφοριών. Σε αντίθεση με την αποθήκευση πληροφοριών στον παραδοσιακό προγραμματισμό, το ANN αποθηκεύει πληροφορίες σε ολόκληρο το δίκτυο — όχι σε μια συγκεκριμένη βάση δεδομένων. Αυτός άλλωστε είναι ένας ακόμα λόγος για τον οποίο είναι ανεκτικά σε σφάλματα. Αν κάποιες πληροφορίες λείπουν δεν μειώνεται η απόδοση ολόκληρου του συστήματος, αλλά αντιμετωπίζονται ως ακραίες τιμές και το μοντέλο συνεχίζει κανονικά.[26]



Εικόνα 7
Τεχνητό Νευρωνικό Δίκτυο

4.7.2 Convolutional Neural Networks (CNN)

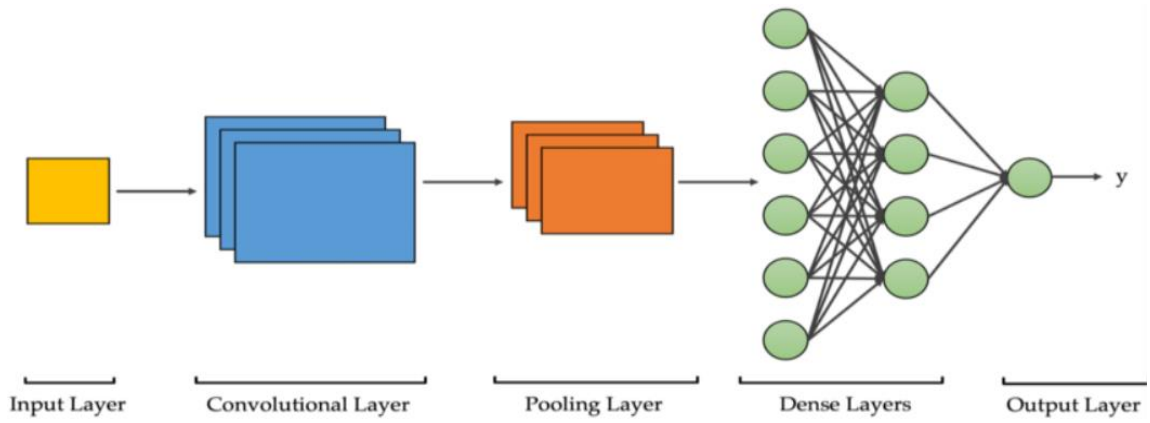
Τα νευρωνικά δίκτυα συνέλιξης είναι γνωστά κυρίως για το ρόλο τους στην αναγνώριση, ανάλυση και ταξινόμηση εικόνων και βίντεο. Πριν την ανακάλυψή τους, τέτοιες εργασίες γίνονταν με πλήρη ευθύνη του ανθρώπινου παράγοντα. Τα CNN ωστόσο βοήθησαν στην κλιμάκωση των διαδικασιών αυτών χρησιμοποιώντας αρχές γραμμικής άλγεβρας για τον εντοπισμό μοτίβων σε εικόνες. Τα CNN βασίζονται σε τρία κύρια επίπεδα τα οποία είναι:

- Συνελικτικό στρώμα (Convolutional layer)
- Στρώμα συγκέντρωσης (Pooling layer)

- Πλήρως συνδεδεμένο στρώμα (Fully-connected layer)

Σε κάθε επίπεδο, η ικανότητα του CNN στην ερμηνεία μιας εικόνας αυξάνεται. Αυτό σημαίνει ότι τα πρώτα επίπεδα επικεντρώνονται στην κατανόηση απλών χαρακτηριστικών , όπως οι άκρες και τα χρώματά της. Στη συνέχεια, το δίκτυο είναι σε θέση να αναγνωρίσει πολύπλοκα χαρακτηριστικά, όπως σχήματα αντικειμένων. Τέλος, το βαθύτερο στρώμα είναι σε θέση να αναγνωρίσει το αντικείμενο στόχο.

Γενικά, αυτός ο τύπος νευρωνικού δικτύου χρησιμοποιείται στη μηχανική όραση η οποία έχει πολλές εφαρμογές, από την αναγνώριση προσώπων σε ποικίλες διαδυκτιακές εφαρμογές έως και την εξέλιξη της βιοϊατρικής μέσω της διευκόλυνσης των επιστημόνων στην αναγνώριση παθήσεων(πχ αναγνώριση όγκων).[26]



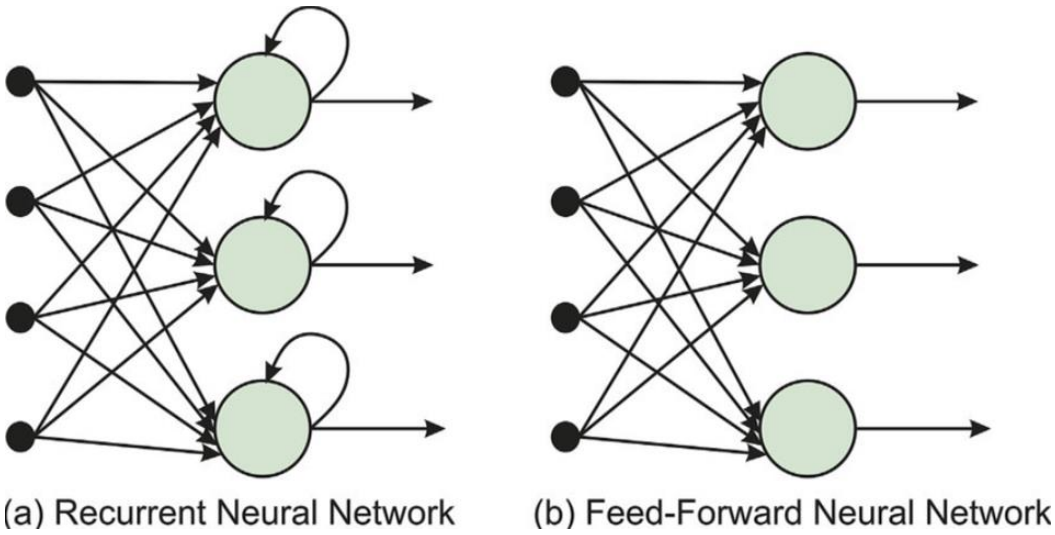
Εικόνα 8 – Convolutional Neural Network

4.7.3 Recurrent Neural Networks (RNN)

Τα RNN είναι μοναδικά λόγω της ικανότητάς τους να επεξεργάζονται τόσο προηγούμενα δεδομένα όσο και δεδομένα εισόδου(διαθέτουν ικανότητα απομνημόνευσης)και αναπτύχθηκαν για να ξεπεραστούν οι αδυναμίες των feed-forward νευρωνικών δικτύων. Πρακτική εφαρμογή τέτοιου τύπου νευρωνικού δικτύου αποτελεί η Siri, προϊόν της Apple.

Όπως το ANN και το CNN, το RNN μαθαίνει επίσης με δεδομένα εκπαίδευσης. Από εκεί και πέρα, η διαδικασία της εκπαίδευσής του δε σταματά εκεί. Αντιθέτως χρησιμοποιεί δεδομένα από προηγούμενες εισροές προκειμένου να λάβει αποφάσεις. Με λίγα λόγια, αυτή η αρχιτεκτονική είναι έτσι δομημένη ώστε να έχει ικανότητα μνήμης. Με αυτά τα δεδομένα γίνεται εύκολα κατανοητή η ικανότητα των RNN να παρέχουν ιδιαίτερα ακριβείς προβλέψεις. Οι εφαρμογές RNN περιλαμβάνουν παραδείγματα όπως η αναγνώριση ομιλίας και η αυτοματοποιημένη μετάφραση.

Τέλος κρίνεται σωστό να αναφερθεί πως το RNN αποτελεί το μοναδικό τύπο δικτύου με μνήμη και διπλή επεξεργασία δεδομένων. Μάλιστα έχει τη δυνατότητα να χαρτογραφήσει πολλές εισόδους και εξόδους.[26]



Εικόνα 9 – Recurrent Neural Network vs Feed-Forward Neural Network

4.7.4 Πίνακας διαφορών ANN, CNN, RNN

Στη συνέχεια παρατίθεται πίνακας με ορισμένες βασικές διαφορές και ομοιότητες ανάμεσα στους τρεις βασικούς τύπους νευρωνικών δικτύων που αναλύθηκαν πρωτύτερα.[27]

	ANN	CNN	RNN
Δεδομένα	Αριθμητικά δεδομένα	Εικόνες	Ακολουθιακά Δεδομένα (πχ Δεδομένα ήχου)
Επαναλαμβανόμενη σύνδεση νευρώνων	ΟΧΙ	ΟΧΙ	ΝΑΙ
Χωρική σχέση στρωμάτων	ΟΧΙ	ΝΑΙ	ΟΧΙ
Ελαχιστοποίηση λάθους	ΝΑΙ	ΝΑΙ	ΝΑΙ
Αρχιτεκτονική	Κλασική feed-forward /	Δομή που βασίζεται σε	Η πληροφορία δε ρέει

	η πληροφορία ρέει προς μία κατεύθυνση	πολλαπλά στρώματα που είναι ομαδοποιημένα / κάθε ομάδα επιτελεί ξεχωριστό ρόλο	μονοσήμαντα / υπάρχει η ικανότητα της απομνημόνευσης
Βασικό Χαρακτηριστικό	Ικανότητα απόδοσης αποτελεσμάτων με ελλειπή δεδομένα	Ικανότητα αναγνώρισης οπτικών δεδομένων	Ικανότητα απομνημόνευσης και αυτόνομης μάθησης
Βασικό Ελάττωμα	Ανάγκη για ισχυρό hardware	Ανάγκη για μεγάλο όγκο δεδομένων	Χρονοβόρα διαδικασία εκπαίδευσης

Πίνακας 1

Τέλος, δεδομένου το γεγονότος πως παραπάνω αναφέρθηκε αρκετές φορές ο όρος feed-forward κρίνεται σκόπιμο να οριστεί κατάλληλα. Συνεπώς ως feed-forward ορίζεται οποιοδήποτε νευρωνικό δίκτυο στο οποίο οι πληροφορίες ρέουν μόνο προς μία κατεύθυνση.[26]

4.8 Ανάπτυξη και Υλοποίηση Νευρωνικού Δικτύου

Η ανάπτυξη οποιουδήποτε νευρωνικού δικτύου απαιτεί τέσσερα βασικά βήματα τα οποία πρόκειται να αναλυθούν παρακάτω. Επιγραμματικά πρόκειται για τον ορισμό του μοντέλου, τη μεταγλώττιση του μοντέλου, την εκπαίδευση του μοντέλου και τον έλεγχο αυτού.

4.8.1 Ορισμός μοντέλου(Model Definition)

Πρόκειται για τη διαδικασία επιλογής του τύπου και κατ'επέκταση της αρχιτεκτονικής του μοντέλου. Αναλυτικότερα, επιλέγεται ο τρόπος διασύνδεσης μεταξύ των στρωμάτων του δικτύου αλλά και το πλήθος των κόμβων που απαρτίζουν το κάθε στρώμα όπως επίσης και ο ίδιος ο αριθμός των στρωμάτων.[9]

4.8.2 Μεταγλώττιση Μοντέλου (Model Compilation)

Αφότου το μοντέλο έχει οριστεί προχωράμε στο compilation. Πρόκειται για τη διαδικασία όπου ορίζονται κάποιες υπερπαράμετροι του νευρωνικού δικτύου όπως το loss function, το batch size ή ακόμα και κάποιος optimizer(έννοιες οι οποίες θα αναλυθούν εκτενέστερα στο επόμενο κεφάλαιο). Ακόμα ορίζονται metrcs τα οποία δίνουν τη δυνατότητα στο χρήστη να παρατηρεί την πρόοδο του μοντέλου[28].

4.8.3 Εκπαίδευση Μοντέλου (Model Training)

Η εκπαίδευση του νευρωνικού δικτύου αποτελεί μία επαναληπτική διαδικασία κατά την οποία τα δεδομένα εκπαίδευσης εισέρχονται στο δίκτυο περνώντας εν συνεχεία από τα βαθύτερα κρυφά στρώματα του δικτύου. Εν τέλει υπολογίζεται μια τιμή η οποία χρησιμοποιείται από έναν αλγόριθμο οπίσθιας διάδοσης προκειμένου να παραμετροποιηθούν οι τιμές των βαρών του δικτύου. Η διαδικασία αυτή συνεχίζεται ανάλογα με τις επιλογές που έχει κάνει ο χρήστης. Είναι

δηλαδή δυνατόν να επιλέξει απριόρι ακριβώς τον αριθμό των επαναλήψεων ή να αυτοματοποιήσει τον τερματισμό της διαδικασίας μόλις κατακτηθεί ο στόχος που έχει θέσει.[28]

4.8.4 Αξιολόγηση του Μοντέλου (Model Evaluation)

Η αξιολόγηση του μοντέλου αποτελεί το τελευταίο στάδιο στάδιο της ανάπτυξής του. Αφότου έχουν ολοκληρωθεί οι προαναφερθείσες διαδικασίες τροφοδοτείται πλέον με τα δεδομένα ελέγχου προκειμένου να παρατηρηθεί η συμπεριφορά του. Αξιολογείται δηλαδή η αποτελεσματικότητα και η ακρίβειά του.[28]

Κεφάλαιο 5

5.1 Χρήσιμα Εργαλεία

5.1.1 Matlab

Η MATLAB ® είναι μια πλατφόρμα προγραμματισμού σχεδιασμένη ειδικά για μηχανικούς και επιστήμονες, η οποία εξυπηρετεί στην ανάλυση και στο σχεδιασμό συστημάτων και προϊόντων που μεταμορφώνουν τον κόσμο μας. Ο πυρήνας της MATLAB είναι η γλώσσα MATLAB, η οποία είναι βασισμένη σε πίνακες, γεγονός το οποίο επιτρέπει μια πιο φυσική έκφραση των υπολογιστικών μαθηματικών. Συνεπώς διευκολύνονται με αυτόν τον τρόπο διεργασίες, όπως η ανάλυση δεδομένων, η ανάπτυξη αλγορίθμων και η δημιουργία μοντέλων και εφαρμογών.

Εκατομμύρια μηχανικοί και επιστήμονες παγκοσμίως χρησιμοποιούν τη MATLAB για εφαρμογές στη βιομηχανία και τον ακαδημαϊκό κόσμο, συμπεριλαμβανομένης της βαθιάς μάθησης και της μηχανικής μάθησης, της επεξεργασίας σήματος και των επικοινωνιών, της επεξεργασίας εικόνας και βίντεο, συστημάτων ελέγχου, δοκιμών και μετρήσεων, όπως επίσης και υπολογιστικής βιολογίας.

Το MATLAB® προσφέρει εξειδικευμένες εργαλειοθήκες για μηχανική μάθηση, νευρωνικά δίκτυα, βαθιά μάθηση, ακόμα και για εφαρμογές αυτοματοποιημένης οδήγησης. Επιτρέπει τη γρήγορη και εύκολη δημιουργία αλλά και την οπτικοποίηση νευρωνικών δικτύων ενώ καλύπτει και το κομμάτι της ενσωμάτωσης ενός νευρωνικού δικτύου σε άλλα διαφορετικά συστήματα.[29]

Εικόνα 10 - Matlab

5.1.2 Python

Η Python είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, υψηλού επιπέδου, η οποία προτιμάται ιδιαίτερα για ταχεία ανάπτυξη εφαρμογών, ενώ επίσης χρησιμοποιείται και ως γλώσσα δέσμης ενεργειών ή συνδετικού στοιχείου για τη σύνδεση υπαρχόντων εφαρμογών μεταξύ τους. Η απλή, εύκολη στην εκμάθηση σύνταξη της Python δίνει έμφαση στην αναγνωσιμότητα και ως εκ τούτου μειώνει το κόστος συντήρησης του εκάστοτε προγράμματος. Μάλιστα, υποστηρίζει πακέτα τα οποία ενθαρρύνουν τη σπονδυλότητα του προγράμματος και την επαναχρησιμοποίηση κώδικα. Τέλος, είναι διαθέσιμη χωρίς χρέωση και διανέμεται

ελεύθερα.

Επιπλέον παρέχει αυξημένη παραγωγικότητα. Δεδομένου ότι δεν υπάρχει βήμα μεταγλώττισης, ο κύκλος επεξεργασίας-δοκιμής-εντοπισμού σφαλμάτων είναι απίστευτα γρήγορος. Ο εντοπισμός σφαλμάτων σε προγράμματα Python είναι εύκολος: ένα σφάλμα ή κακή είσοδος δεν πρόκειται να προκαλέσει ποτέ σφάλμα τμηματοποίησης. Αντίθετα, όταν ο διερμηνέας ανακαλύπτει ένα σφάλμα, δημιουργεί μια εξαίρεση. Όταν το πρόγραμμα δεν εντοπίζει την εξαίρεση, ο διερμηνέας εκτυπώνει ένα ίχνος στοίβας. Ένα πρόγραμμα εντοπισμού σφαλμάτων σε επίπεδο πηγής, επιτρέπει την επιθεώρηση τοπικών και καθολικών μεταβλητών, την αξιολόγηση αυθαίρετων εκφράσεων, τον ορισμό σημείων διακοπής, τη μετάβαση στον κώδικα μια γραμμή κάθε φορά και ούτω καθεξής. Το πρόγραμμα εντοπισμού σφαλμάτων είναι γραμμένο στην ίδια την Python, μαρτυρώντας έτσι την ενδοσκοπική δύναμή της.[30]

Εικόνα 11 – Python

5.1.3 Πλεονεκτήματα της Python σε σχέση με τη Matlab

Παρακάτω παρατίθενται οι λόγοι για τους οποίους οι ιδιότητες της Python είναι πλεονεκτικές σε σχέση με αυτές της Matlab παρά τα πολυάριθμα συγκρίσιμα χαρακτηριστικά τους.

- Ο κώδικας Python είναι πιο συμπαγής και πιο ευανάγνωστος από τον κώδικα Matlab:

  1. Σε αντίθεση με τη Matlab, η οποία χρησιμοποιεί την εντολή end για να υποδείξει το τέλος ενός μπλοκ, στην Python το μέγεθος του μπλοκ καθορίζεται με βάση τις εσοχές.
  2. Η Python χρησιμοποιεί αγκύλες για ευρετηρίαση και παρενθέσεις για συναρτήσεις και μεθόδους,ενώ η Matlab χρησιμοποιεί παρενθέσεις και για τα δύο, καθιστώντας πιο δύσκολη τη διαφοροποίηση μεταξύ τους.
  3. Η καλύτερη αναγνωσιμότητα της Python οδηγεί σε λιγότερα σφάλματα και στον ταχύτερο εντοπισμό αυτών.

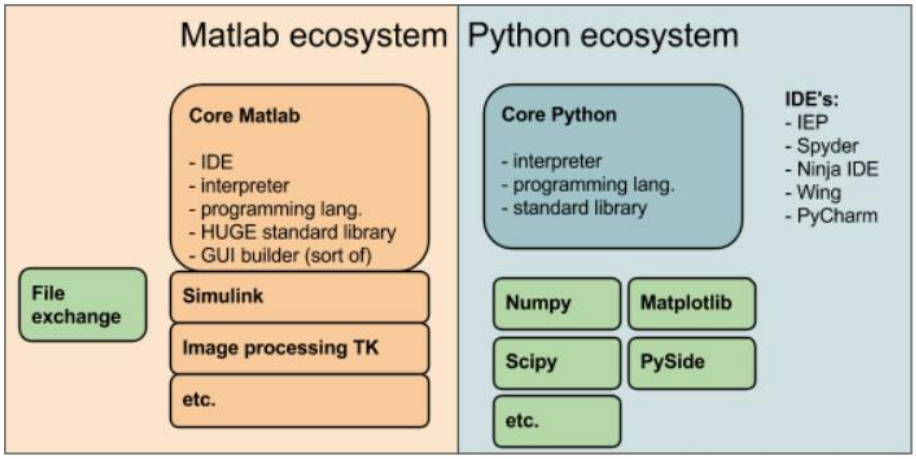
- Ενώ οι περισσότερες γλώσσες προγραμματισμού, συμπεριλαμβανομένης της Python, χρησιμοποιούν μηδενική ευρετηρίαση(zero-based indexing), η Matlab χρησιμοποιεί μοναδιαία ευρετηρίαση(one-based indexing), καθιστώντας πιο περίπλοκη την κατανόησή της για τους χρήστες.
- Ο αντικειμενοστραφής προγραμματισμός (OOP) στην Python αποτελεί ένα εξαιρετικά ευέλικτο εργαλείο ενώ το σχήμα OOP της Matlab είναι πολύπλοκο και προκαλεί σύγχυση.

- Η Python διανέμεται δωρεάν και αποτελεί πλατφόρμα ανοιχτού κώδικα:

    1. Ενώ η Python αποτελεί προγραμματισμό ανοιχτού κώδικα, μεγάλο μέρος της Matlab είναι κλειστό.
    2. Το δυναμικό της Python ενθαρρύνει τους χρήστες να εισάγουν προτάσεις για το λογισμικό, ενώ όσων αφορά στη Matlab δεν προσφέρεται τέτοια αλληλεπίδραση.

- Η Python προσφέρει ένα ευρύτερο σύνολο επιλογών σε πακέτα γραφικών, βιβλιοθήκες και σύνολα εργαλείων

- Η Python παρέχει αρκετές βιβλιοθήκες σε σχέση με τη μηχανική μάθηση αντίθετα με τη Matlab.[31]



Εικόνα 12 – Python Ecosystem & Matlab Ecosystem

Είναι εύλογο λοιπόν το γεγονός πως στην παρούσα εργασία επιλέγεται η Python για επίλυση του αντίστροφου προβλήματος αναγνώρισης ρωγμών σε ιδανικό ελαστικό υλικό. Πιο συγκεκριμένα, γίνεται χρήση ορισμένων πακέτων τα οποία αφορούν στη δημιουργία και στην εκπαίδευση μοντέλων νευρωνικών δικτύων όπως το Keras και το TensorFlow, τα οποία θα αναλυθούν στη συνέχεια.

5.1.4 TensorFlow

Το TensorFlow είναι μια end-to-end πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση. Διαθέτει ένα ολοκληρωμένο και ευέλικτο οικοσύστημα εργαλείων, βιβλιοθηκών και λοιπών πόρων που επιτρέπει στους χρήστες να προωθούν την τελευταία λέξη της τεχνολογίας στη μηχανική μάθηση ή πιο απλά επιτρέπει την εύκολη ανάπτυξη και δημιουργία εφαρμογών που υποστηρίζονται από μηχανική μάθηση.

Το TensorFlow καθιστά αρκετά εύκολη τη δημιουργία μοντέλων μηχανικής μάθησης προσφέροντας πολλαπλά επίπεδα αφαίρεσης(levels of abstraction), δίνοντας έτσι τη δυνατότητα επιλογής του καταλληλότερου μοντέλου για κάθε ανάγκη. Προσφέρει ακόμα τη δυνατότητα δημιουργίας και εκπαίδευσης αυτών των μοντέλων χρησιμοποιώντας το υψηλού επιπέδου Keras

API. Σε περίπτωση ανάγκης για περισσότερη ευελιξία, υπάρχει ακόμα η δυνατότητα χρήσης του Distribution Strategy API για κατανεμημένη εκπαίδευση σε διαφορετικές διαμορφώσεις υλικού χωρίς την ανάγκη αλλαγής του ορισμού του μοντέλου.

Επιπλέον, η εν λόγω πλατφόρμα παρέχει έναν άμεσο δρόμο προς την παραγωγή. Είτε πρόκειται για servers, είτε απλά για τον ιστό (Web) το TensorFlow επιτρέπει την εκπαίδευση και ανάπτυξη του κάθε μοντέλου αρκετά εύκολα, ανεξάρτητα από τη γλώσσα προγραμματισμού ή την πλατφόρμα που χρησιμοποιείται σε οποιαδήποτε περίπτωση. Σε περίπτωση διοχέτευσης μιας πλήρους εφαρμογής μηχανικής μάθησης υπάρχει το TensorFlow Extended, σε περίπτωση εκτέλεσης ενός μοντέλου μηχανικής μάθησης κρίνεται κατάλληλο το TensorFlow Lite ενώ π.χ. σε περιβάλλον JavaScript χρησιμοποιείται το TensorFlow.js, τα οποία ωστόσο δεν πρόκειται να αναλυθούν περαιτέρω.

Επιπροσθέτως, η ταχύτητα και η απόδοση αποτελούν βασικά στοιχεία των μοντέλων νευρωνικών δικτύων που χτίζονται μέσω του TensorFlow. Επίσης υποστηρίζει ένα οικοσύστημα ισχυρών πρόσθετων βιβλιοθηκών και μοντέλων για πειραματισμό, συμπεριλαμβανομένων των Ragged Tensors, TensorFlow Probability, Tensor2Tensor και BERT.

Τέλος κρίνεται σκόπιμο να αναφερθεί πως αναπτύχθηκε από την ομάδα Google Brain σε συνεργασία με τον ερευνητικό οργανισμό ‘Machine Intelligence Research Organization for Machine Learning and Deep Neural Networks. Ακόμα, είναι cross-platform καθώς μπορεί να εκτελείται σε πληθώρα hardware(επεξεργαστές, κάρτες γραφικών κ.λπ.) ενώ πολλές είναι και οι γνωστές εταιρίες που το χρησιμοποιούν όπως η Booking και η Coca-Cola.[32]

Εικόνα 13 - TensorFlow

5.1.5 Keras

Πρόκειται για ένα API βαθιάς εκμάθησης γραμμένο σε Python, το οποίο τρέχει πάνω(On Top) από την πλατφόρμα μηχανικής εκμάθησης TensorFlow και βασικό στόχο της ανάπτυξής του αποτελεί η δυνατότητα εξαγωγής αποτελεσμάτων με ταχύτητα και ακρίβεια. Άλλωστε το κλειδί για μια επιτυχημένη έρευνα είναι η γρήγορη μετάβαση από την ιδέα σε ένα απτό αποτέλεσμα. Το Keras, παρέχει βασικά αφαιρετικά επίπεδα(abstraction levels) και δομικά στοιχεία για την ανάπτυξη λύσεων μηχανικής εκμάθησης με υψηλή ταχύτητα επαναλήψεων ενώ γενικά αποτελεί μια πλήρως επεκτάσιμη Cross-Platform διεπαφή. Συνάμα, μπορεί να εκτελεστεί σε TPU ή ακόμα και σε μεγάλα συμπλέγματα GPUs, ενώ τα μοντέλα του μπορούν να εξαχθούν προκειμένου να εκτελεστούν σε εφαρμογές Web ή Mobile.

Στα θετικά του συγκαταλέγονται η απλότητα που το διακρίνει, η ευελιξία του και η ισχύς του. Ακολούθως ο γνωστικός φόρτος του εκάστοτε χρήστη μειώνεται με αποτέλεσμα το ουσιαστικό βάρος να δίνεται σε μέρη του προβλήματος που έχουν περισσότερη σημασία. Οι βασικές δομές δεδομένων του Keras είναι επίπεδα και μοντέλα ενώ ο πιο διαδεδομένος τύπος μοντέλου είναι το Sequential Model που αποτελεί μια γραμμική στοίβα στρωμάτων.

Ιστορικά στοιχεία:
Κέρας σημαίνει κέρατο στα ελληνικά. Είναι μια αναφορά σε μια λογοτεχνική εικόνα από την αρχαία ελληνική και λατινική λογοτεχνία, που βρέθηκε για πρώτη φορά στην Οδύσσεια. Εκεί, τα πνεύματα των ονείρων (Όνειροι) χωρίζονται μεταξύ εκείνων που εξαπατούν τους ονειροπόλους με ψεύτικα οράματα και φτάνουν στη Γη μέσω μιας πύλης από ελεφαντόδοντο και σε αυτούς που ανακοινώνουν μελλοντικά γεγονότα που θα συμβούν και φτάνουν μέσα από μια πύλη κέρατος. Είναι ένα παιχνίδι με τις λέξεις κέρας (κέρατο) / κραίνω (εκπληρώνω) και ἐλέφας (ελεφαντόδοντο) / ἐλεφαίρομαι (απατώ).[33]

Εικόνα 14 – Keras

5.1.6 Keras συνδιαστικά με TensorFlow (Keras over TensorFlow)

Ακόμα, κρίνεται αναγκαίο να αποσαφηνιστεί περαιτέρω ο τρόπος με τον οποίο διαφοροποιούνται το Keras σε σχέση με το TensorFlow. Όπως είναι γνωστό, τα πλαίσια μηχανικής μάθησης λειτουργούν πολυεπίπεδα. Στο χαμηλότερο επίπεδο (Backend) τοποθετείται το TensorFlow, χάρη στο οποίο εκτελούνται όλοι οι μαθηματικοί υπολογισμοί που επιτελούνται στο παρασκήνιο. Από την άλλη, το Keras βρίσκεται στην κορυφή με αποτέλεσμα να χρησιμοποιεί τα αποτελέσματα του Backend περιβάλλοντος ούτως ώστε να συμβάλει άμεσα στη δημιουργία μοντέλων νευρωνικών δικτύων αλλά και των παραμέτρων αυτών. Με λίγα λόγια δίνει στο χρήστη τη δυνατότητα να δημιουργεί νευρωνικά δίκτυα αλλά και να προσαρμόζει πλήρως τον τρόπο με τον οποίο γίνεται η διαδικασία εκμάθησης χωρίς να έρχεται σε επαφή με περίπλοκες μαθηματικές πράξεις.

Επιπλέον, κρίνεται ορθό να αναφερθεί πως το Keras μπορεί να χρησιμοποιήσει και άλλες Backend πλατφόρμες όπως το Theano, αποτελώντας μια τυπική απλοποιημένη διεπαφή προγραμματισμού(interface).

Όσων αφορά στην επιλογή του Keras συνδιαστικά με το TensorFlow για την υλοποίηση του πειράματος της συγκεκριμένης εργασίας οι λόγοι είναι αρκετοί. Αρχικά προσφέρεται στο χρήστη ένα εύχρηστο περιβάλλον, το οποίο ουδεμία σχέση έχει με το δύσχρηστο TensorFlow. Συνεπώς ο

χρόνος ανάπτυξης του μοντέλου μηχανικής μάθησης μειώνεται δραστικά. Επιπλέον, δίνεται η δυνατότητα της εύκολης μετατροπής του εκάστοτε μοντέλου προκειμένου να μπορεί να χρησιμοποιηθεί ως πρόσθετο σε άλλες εφαρμογές(Android-IOS-Web Apps κ.λπ.) ενώ από άποψη hardware τα μοντέλα του Keras έχουν την ικανότητα να εκπαιδεύονται σε διαφορετικές πλατφόρμες όπως NVIDIA GPU, OPENCL-enabled GPUs και Google TPUs. Συνεπώς , οι περιορισμοί είναι ελάχιστοι και η επιλογή του Keras αποτέλεσε μονόδρομο.[9]

Εικόνα 15 – Keras over TensorFlow

5.2 Ανάπτυξη εγχειριδίου

Στο επόμενο μέρος του συγκεκριμένου κεφαλαίου, πρόκειται να αναλυθεί εκτενέστερα ο κώδικας του αρχικού προβλήματος παράλληλα με το νέο κώδικα. Θα παρουσιαστούν σημεία που ομοιάζουν ωστόσο έμφαση θα δοθεί στον τρόπο με τον οποίο μεταβήκαμε από Matlab σε Python ιδίως όσων αφορά σε διαδικασίες που συντελέστηκαν με εντελώς διαφορετικό τρόπο. Τέλος ειδική μνεία θα γίνει στον αλγόριθμο που χρησιμοποιήθηκε για την εκπαίδευση του νευρωνικού δικτύου ο οποίος παρόλο που παρέμεινε όμοιος με την αρχική υλοποίηση εφαρμόστηκε με ένα νέο ιδιαίτερο τρόπο, διαφορετικό από τους συνηθισμένους σε ότι αφορά στο Keras.

5.2.1 Δεδομένα Νευρωνικού Δικτύου

5.2.1.1 Ανάλυση Δεδομένων (Dataset Analysis)

Όπως έχει ήδη αναφερθεί, το πρόβλημα που αντιμετωπίζεται στην παρούσα εργασία έχει να κάνει με την εύρεση παραμέτρων που χαρακτηρίζουν μια ελλειψοειδή οπή, η οποία προσεγγίζει μια ρωγμή, από μετρήσεις μετακινήσεων στην εξωτερική πλευρά ενός στατικά φορτισμένου δίσκου. Στην εν λόγω υλοποίηση το πρόβλημα είναι συγκεντρωτικό καθώς στα δεδομένα εκπαίδευσης και δοκιμής συγκεντρώνονται όλες οι περιπτώσεις που θα δημιουργήσουν σφάλμα στο υλικό που μελετάται. Αναλυτικότερα, τα δεδομένα εισόδου αποτελούν μετρήσεις μετακινήσεων των δομικών στοιχείων της πλάκας ενώ τα δεδομένα εξόδου είναι συντεταγμένες x,y του κέντρου της οπής. Τα δεδομένα για την εκπαίδευση του μοντέλου αναγνώρισης νευρωνικών δικτύων και τη δοκιμή του, παράγονται από μια τεχνική υπολογιστικής μηχανικής που βασίζεται σε οριακά στοιχεία. Η μεθοδολογία
ισχύει γενικά για τον εντοπισμό σφαλμάτων, την ανίχνευση ελαττωμάτων και τον ποιοτικό

έλεγχο κατασκευών ή δομικών στοιχείων .Τα δεδομένα αυτά, θα μπορούσαν να είναι αποτελέσματα πειράματος ωστόσο στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε software το οποίο βασίζεται στον κώδικα QUADPLEH. Με παρόμοιο τρόπο θα μπορούσε να γίνει συλλογή δεδομένων και για τεχνικές αναγνώρισης, οι οποίες υιοθετούνται ευρέως στη βιομηχανία για εργασίες ποιοτικού ελέγχου.[1]

Πιο συγκεκριμένα, το dataset που χρησιμοποιήθηκε, όπως αναφέρεται και σε προηγούμενο κεφάλαιο αποτελείται από τα δεδομένα εκπαίδευσης (training data) και τα δεδομένα ελέγχου(test data). Στην περίπτωσή μας, τα δεδομένα εκπαίδευσης όπως και τα δεδομένα ελέγχου χωρίζονται δε δεδομένα εισόδου και σε δεδομένα εξόδου. Σε ότι αφορά στα δεδομένα εισόδου εκπαίδευσης διαθέτουμε 81 παραδείγματα τιμών μετακίνησης τα οποία αντιστοιχίζονται σε 82 εισόδους. Οι έξοδοι είναι 2 και βάση των στοιχείων που διαθέτουμε γίνεται κατανοητό πως ο πίνακας που αντιστοιχεί στην έξοδο θα έχει διαστάσεις 2x82 προκειμένου να καλυφθεί κάθε περίπτωση αναγνώρισης σφάλματος. Όσων αφορά στα δεδομένα ελέγχου, οι είσοδοι και οι έξοδοι προφανώς παραμένουν ισάριθμες ενώ αλλάζει ο αριθμός των παραδειγμάτων που ελέγχονται καθώς αυτά είναι 64. Αναλόγως εξάγεται και ο πίνακας που αντιστοιχεί στην έξοδο του νευρωνικού δικτύου με διαστάσεις 2x64.

Από τη βιβλιογραφία γνωρίζουμε πως μεγάλος όγκος δεδομένων εκπαίδευσης σημαίνει εξίσου μεγάλη απόδοση του του νευρωνικού δικτύου. Ωστόσο, με δεδομένο τον μικρό αριθμό δεδομένων εκπαίδευσης που έχουμε στην παρούσα εργασία, η ακρίβεια των αποτελεσμάτων έγκειται κυρίως σε υπερπαραμέτρους του δικτύου(αλγόριθμος εκπαίδευσης) που θα αναλυθούν εν συνεχεία.

5.2.1.2 Τεχνική Ανάλυση Δεδομένων

Κατά την υλοποίηση σε Matlab τα δεδομένα εισήχθησαν με τη μορφή *.m file. Ένα *.m file είναι ένα απλό αρχείο κειμένου όπου τοποθετούνται εντολές Matlab ή δεδομένα με τη μορφή πινάκων ή διανυσμάτων. Όταν εκτελείται το αρχείο, η Matlab το διαβάζει στοιχείο προς στοιχείο.[34]

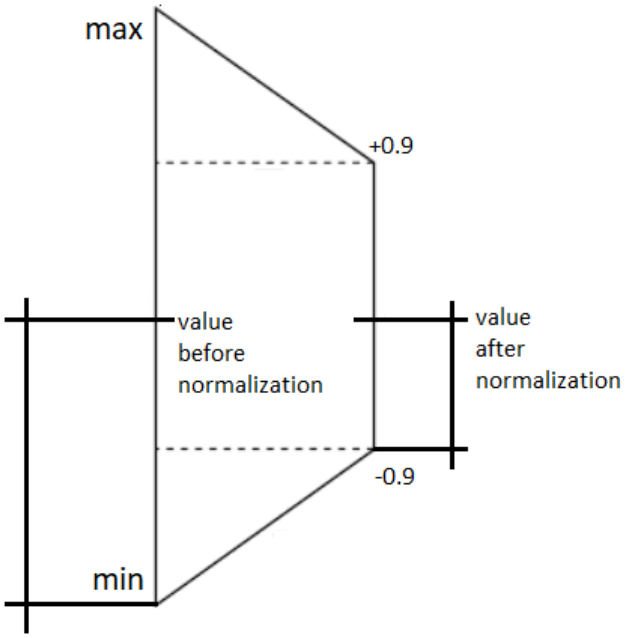
Αντιθέτως κατά την υλοποίηση σε Python τα δεδομένα εισήχθησαν σε μορφή *.csv. Τα *.csv files είναι αρχεία τιμών διαχωρισμένων με κόμματα. Ουσιαστικά, αποτελούν οριοθετημένα αρχεία κειμένου που χρησιμοποιούν το κόμμα για να διαχωρίσουν τις τιμές που περιλαμβάνουν. Κάθε γραμμή του αρχείου είναι μια εγγραφή δεδομένων και κάθε εγγραφή αποτελείται από ένα ή περισσότερα πεδία. Η χρήση του κόμματος ως διαχωριστικού πεδίου είναι η πηγή του ονόματος για αυτήν τη μορφή αρχείου. Ένα αρχείο CSV συνήθως αποθηκεύει δεδομένα πίνακα (αριθμούς και κείμενο) σε απλό κείμενο, οπότε κάθε γραμμή θα έχει τον ίδιο αριθμό πεδίων[35]

Γενικότερα, τα αρχεία ASCII (όπως το CSV) απαιτούν μετατροπή από και προς τη δυαδική μορφή στη μνήμη, γεγονός που τα καθιστά σχετικά αργά. Ωστόσο, στην περίπτωσή μας εφόσον ο όγκος των δεδομένων είναι αρκετά μικρός η διαφορά είναι ανεπαίσθητη. Επίσης, χρησιμοποιήθηκαν *.csv αρχεία καθώς ήταν αναγκαία η ανταλλαγή δεδομένων με λογισμικό που μπορεί να διαβάσει *.csv αλλά δεν μπορεί να διαβάσει *.mat files.

5.2.1.3 Κανονικοποίηση Δεδομένων (Data Normalization)

Επιπροσθέτως, κρίνεται απαραίτητο να αναφερθεί πως προτού τροφοδοτηθούν στο νευρωνικό δίκτυο τα εν λόγω δεδομένα υφίστανται κάποια επεξεργασία. Εν ολίγοις, τα δεδομένα κανονικοποιούνται σε ένα στάδιο προεπεξεργασίας κατά το οποίο προετοιμάζονται με τέτοιο τρόπο ώστε να αφομοιωθούν αποτελεσματικά από το νευρωνικό δίκτυο. Η διαδικασία αυτή άλλες φορές είναι χρήσιμη και άλλες καθίσταται εντελώς αναγκαία.

Η κανονικοποίηση λοιπόν, είναι η διαδικασία μεταβολής των δεδομένων ώστε να βρίσκονται σε συγκεκριμένο εύρος, όπως για παράδειγμα μεταξύ 0 και 1 ή στην περίπτωσή μας μεταξύ -1 και 1. Απαιτείται όταν υπάρχουν μεγάλες διαφορές στο εύρος των διαφορετικών στοιχείων των δεδομένων και γίνεται χρήσιμη όταν το σύνολο των δεδομένων δεν περιέχει ακραίες τιμές. Τέλος, Το θεωρητικό υπόβαθρο της κανονικοποίησης μπορεί να γίνει εύκολα κατανοητό από την παρακάτω εικόνα.



Εικόνα 16 – Σχήμα Κανονικοποίησης

Η εξήγηση έγκειται στην τριγωνομετρία και παρουσιάζεται στη συνέχεια:

$$\frac{ValueAftrerNormalization - (-0.9)}{0.9 - (-0.9)} = \frac{ValueBeforeNormalization - min}{max - min}$$

$$\frac{ValueAftrerNormalization + 0.9}{1.8} = \frac{ValueBeforeNormalization - min}{max - min}$$

$$ValueAfterNormalization = \left(\frac{ValueBeforeNormalization - min}{max - min} * 1.8\right) - 0.9 \qquad (11)$$

Τέλος, όσων αφορά στα πλεονεκτήματα που προσφέρει η κανονικοποίηση, αυτά έχουν να κάνουν με την επιτάχυνση της διαδικασίας εκπαίδευσης του νευρωνικού δικτύου αλλά και την ταχύτερη σύγκλιση της καμπύλης της συνάρτησης λάθους(loss function). Ειδικά για τα νευρωνικά δίκτυα η κανονικοποίηση καθίσταται απαραίτητη διότι όταν εισάγονται μη κανονικοποιημένες είσοδοι σε συναρτήσεις ενεργοποίησης η διαδικασία εκμάθησης γίνεται αρκετά δύσκολη ή μπορεί ακόμα και να μην καταφέρουμε να εξάγουμε επαρκή αποτελέσματα.[36] Φυσικά αυτό θα αναπτυχθεί εκτενέστερα στη συνέχεια όπου πρόκειται να αναλυθεί η αρχιτεκτονική του μοντέλου που δημιουργήθηκε για τους σκοπούς αυτής της εργασίας.

5.2.1.4 Διαφοροποίηση κώδικα Python ως προς την Επεξεργασία Δεδομένων

Σε αυτό το σημείο κρίνεται σωστό να αναφερθεί πως κατά την υλοποίηση σε Python έγιναν κάποια παραπάνω βήματα όσων αφορά στο επίπεδο επεξεργασίας των δεδομένων. Αρχικά χρησιμοποιήθηκε η συνάρτηση του TensorFlow, from_tensor_slices η οποία χρησιμοποιώντας τα δεδομένα εκπαίδευσης δημιουργεί ένα νέο σύνολο δεδομένων του οποίου τα στοιχεία είναι “φέτες”(slices) των tensors που εισήχθησαν σε αυτή. Αυτοί οι tensors κόβονται κατά μήκος της πρώτης τους διάστασης. Αυτή η λειτουργία, διατηρεί τη δομή των τανυστών εισόδου, αφαιρώντας την πρώτη διάσταση κάθε τανυστή και χρησιμοποιώντας την ως διάσταση δεδομένων. Προφανώς, όλοι οι τανυστές εισόδου πρέπει να έχουν το ίδιο μέγεθος στις πρώτες τους διαστάσεις.

Εν συνεχεία χρησιμοποιήθηκε η συνάρτηση shuffle του TensorFlow η οποία ανακατεύει τυχαία τα στοιχεία του συνόλου δεδομένων για το οποίο έγινε λόγος νωρίτερα. Αυτό το σύνολο δεδομένων γεμίζει ένα buffer με στοιχεία συγκεκριμένου μεγέθους και έπειτα, δημιουργεί τυχαία δείγματα στοιχείων από αυτό το buffer, αντικαθιστώντας τα επιλεγμένα στοιχεία με νέα στοιχεία. Προκειμένου αυτή η διαδικασία να αποδόσει τα μέγιστα, απαιτείται μέγεθος buffer μεγαλύτερο ή ίσο με το πλήρες μέγεθος του συνόλου δεδομένων. Στην περίπτωσή μας επιλέχθηκε μέγεθος ίσο με το πλήρες μέγεθος του συνόλου των δεδομένων. Ύστερα, χρησιμοποιήθηκε η συνάρτηση batch του TensorFlow η οποία συνδυάζει διαδοχικά στοιχεία αυτού του συνόλου δεδομένων σε παρτίδες.

Τέλος, χρησιμοποιήθηκε η συνάρτηση prefech του TensorFlow η οποία δημιουργεί ένα σύνολο δεδομένων προανακτώντας στοιχεία από το ίδιο. Αυτό, επιτρέπει την προετοιμασία μεταγενέστερων στοιχείων κατά την επεξεργασία του εκάστοτε τρέχοντος στοιχείου και κατά συνέπεια βελτιώνεται ο χρόνος εκπαίδευσης του νευρωνικού δικτύου , με το κόστος χρήσης πρόσθετης μνήμης να μειώνεται για την αποθήκευση στοιχείων προανάκτησης.[37]

5.2.2 Περαιτέρω Ανάλυση Stohastic Gradient Descent-Backpropagation και αλγορίθμου

Levenberg -Marquardt

Πριν παρατεθεί η αντίστοιχη επίλυση του προβλήματος αναγνώρισης που εξετάζουμε σε κώδικα Python κρίνεται απαραίτητο να αποσαφηνιστούν ορισμένες έννοιες που δεν αναλύθηκαν αρκετά στο Κεφάλαιο 4.

Αρχικά πρόκειται να παρουσιαστούν οι διαφορές ανάμεσα σε Backpropagation και Gradient Descent. Γενικά, υπάρχει μεγάλη σύγχυση σχετικά με τον αλγόριθμο που χρησιμοποιείται για την εκπαίδευση των μοντέλων των νευρωνικών δικτύων βαθιάς μάθησης. Άλλωστε είναι σχετικά σύνηθες να ακούμε πως νευρωνικά δίκτυα "μαθαίνουν" χρησιμοποιώντας "back-propagation of error" ή "stochastic gradient descent". Μερικές φορές, οποιοσδήποτε από αυτούς τους αλγόριθμους χρησιμοποιείται ως συντομογραφία σε σχέση με τον τρόπο εκμάθησης ενός νευρωνικού δικτύου, αν και σε πολλές περιπτώσεις, υπάρχει βαθιά σύγχυση ως προς το τι είναι αυτοί οι αλγόριθμοι, πώς συσχετίζονται και πώς μπορούν να λειτουργήσουν μαζί.

5.2.2.1 Stohastic Gradient Descent και Backpropagation

Το Gradient Descent είναι ένας αλγόριθμος βελτιστοποίησης, ο οποίος βρίσκει το σύνολο των μεταβλητών εισόδου για μια συνάρτηση οδηγώντας έτσι, σε μια ελάχιστη τιμή της συνάρτησης αυτής.

Αναλυτικότερα, η παράγωγος πρώτης τάξης μιας συνάρτησης υπολογίζει την κλίση ή την καμπυλότητα μιας συνάρτησης σε ένα δεδομένο σημείο. Μια θετική παράγωγος υποδηλώνει ότι η συνάρτηση έχει κλίση προς τα πάνω και μια αρνητική παράγωγος υποδηλώνει ότι η συνάρτηση έχει κλίση προς τα κάτω. Εφόσον η συνάρτηση λαμβάνει πολλές μεταβλητές εισόδου, πρόκειται για διάνυσμα μεταβλητών. Στο διανυσματικό λογισμό, το διάνυσμα των παραγώγων πρώτης τάξης (μερικές παράγωγοι) αναφέρεται γενικά ως η κλίση της συνάρτησης(gradient).

Όπως ήδη ανφέρθηκε, ο αλγόριθμος Gradient Descent απαιτεί τον υπολογισμό της κλίσης της συνάρτησης σε σχέση με συγκεκριμένες τιμές εισόδου. Όταν η κλίση είναι ανηφορική, το αρνητικό της κλίσης κάθε μεταβλητής εισόδου ακολουθείται κατηφορικά ούτως ώστε να προκύψουν νέες τιμές για κάθε μεταβλητή, κάτι το οποίο σταδιακά οδηγεί στο ελάχιστο της εκάστοτε συνάρτησης. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να εντοπιστεί το ελάχιστο της συνάρτησης , να αξιολογηθεί ένας μέγιστος αριθμός υποψήφιων λύσεων ή κάποια άλλη συνθήκη διακοπής. Επιπλέον, ένα μέγεθος βήματος (step size) απαιτείται για την κλίμακα της διαβάθμισης και τον έλεγχο του πόσο θα αλλάξει η κάθε μεταβλητή εισόδου σε σχέση με το gradient. Το μέγεθος βήματος αναφέρεται επίσης και ως learning rate ή alpha και αποτελεί υπερπαράμετρο του νευρωνικού δικτύου.

Το Gradient Descent μπορεί να προσαρμοστεί προκειμένου να ελαχιστοποιήσει τη συνάρτηση απώλειας(Loss Function) ενός προγνωστικού μοντέλου νευρωνικού δικτύου σε ένα σύνολο δεδομένων εκπαίδευσης, όπως ένα μοντέλο ταξινόμησης(classification model) ή παλινδρόμησης(regression model). Αυτή η προσαρμογή ονομάζεται Stohastic Gradient Descent.

Η συνάρτηση για την οποία γίνεται λόγος στην προηγούμενη παράγραφο μπορεί να είναι μια συνάρτηση όπως αυτή του μέσου τετραγωνικού σφάλματος ενώ οι ίδιες οι παράμετροι του μοντέλου νευρωνικού δικτύου(πχ βάρη) λαμβάνονται υπόψιν ως μεταβλητές για τη συνάρτηση-στόχο.

Ο αλγόριθμος αναφέρεται ως «στοχαστικός» επειδή οι διαβαθμίσεις της συνάρτησης-στόχου σε

σχέση με τις μεταβλητές εισόδου είναι θορυβώδεις (π.χ. μια πιθανολογική προσέγγιση). Αυτό σημαίνει ότι η αξιολόγηση της κλίσης μπορεί να έχει στατιστικό θόρυβο ο οποίος μπορεί να κρύβει το πραγματικό υποκείμενο σήμα κλίσης. Κάτι τέτοιο προκαλείται λόγω της αραιότητας και του θορύβου στο σύνολο δεδομένων εκπαίδευσης.

Μια πρόκληση όταν χρησιμοποιείται ο αλγόριθμος Stohastic Gradient Descent είναι ο τρόπος υπολογισμού του gradient(κλίσης) για νευρώνες που ανήκουν σε κρυφά επίπεδα του δικτύου, όπως για παράδειγμα για νευρώνες που βρίσκονται ένα ή περισσότερα βήματα μακριά από το επίπεδο εξόδου. Κάτι τέτοιο, απαιτεί μια συγκεκριμένη τεχνική η οποία ονομάζεται κανόνας αλυσίδας και έναν αποτελεσματικό αλγόριθμο ο οποίος εφαρμόζει τον κανόνα της αλυσίδας και μπορεί να χρησιμοποιηθεί για τον υπολογισμό των κλίσεων για οποιαδήποτε παράμετρο στο νευρωνικό δίκτυο. Αυτός ο αλγόριθμος ονομάζεται backpropagation.

Τέλος, μπορούμε να πούμε πως ως backpropagation ορίζεται ένας αλγόριθμος αυτόματου διαφορικού υπολογισμού ο οποίος χρησιμοποιείται για τον υπολογισμό των gradients των παραμέτρων των νευρωνικών δικτύων[41].Ο όρος backpropagation συχνά παρεξηγείται με τέτοιο τρόπο ώστε να εννοείται ολόκληρος ο αλγόριθμος εκμάθησης για πολυεπίπεδα νευρωνικά δίκτυα. Στην πραγματικότητα, η αντίστροφη διάδοση αναφέρεται μόνο στη μέθοδο για τον υπολογισμό του gradient, ενώ ένας άλλος αλγόριθμος, όπως ο stohastic gradient descent, χρησιμοποιείται προκειμένου να συντελεστεί η διαδικασία μάθησης χρησιμοποιώντας αυτό το gradient. Μαζί, ο αλγόριθμος gradient descent και ο αλγόριθμος backpropagation αποτελούν μια πληρέστερη περιγραφή του γενικού αλγορίθμου που χρησιμοποιείται για την εκπαίδευση ενός νευρωνικού δικτύου,αναφορικά με τους αλγορίθμους βελτιστοποίησης(optimization algorithm) και υπολογισμού κλίσης(gradient calculation algorithm).[42]

5.2.2.2 Levenberg-Marquardt

Ο αλγόριθμος Levenberg-Marquardt (LM) είναι ένας πολύ γνωστός αλγόριθμος μη γραμμικής βελτιστοποίησης για προσέγγιση συναρτήσεων. Ο Levenberg [44] ήταν ο αρχικός δημιουργός αυτής της μεθόδου για συναρτήσεις μη γραμμικών ελαχίστων τετραγώνων με βάση τη διαπίστωση ότι η απλή μέθοδος gradient descent και η επανάληψη Gauss-Newton είναι συμπληρωματικές μέθοδοι. Ο Marquardt [45] επέκτεινε την αρχική μέθοδο κλιμακώνοντας κάθε στοιχείο του gradient σύμφωνα με την καμπυλότητα, συμπεριλαμβάνοντας τη διαγώνιο του Hessian matrix.[43]

Ο Hessian Matrix είναι ένας τετράγωνος πίνακας δευτερευουσών μερικών παραγώγων μιας βαθμωτής συνάρτησης. Χρησιμοποιείται γενικά στη γραμμική άλγεβρα και ειδικότερα για τον προσδιορισμό σημείων τοπικών μέγιστων ή ελάχιστων.[46]

Υπάρχουν πολλές εργασίες στην ανοιχτή βιβλιογραφία σχετικά με αυτόν τον αλγόριθμο όπως επίσης και εφαρμογές αυτού. Ο Roweis [47] δίνει μια εξαιρετική διαισθητική περιγραφή του αλγορίθμου. Ουσιαστικά συνόψισε το γεγονός πως ο αλγόριθμος Levenberg-Marquardt λειτουργεί εξαιρετικά καλά στην πράξη και έχει γίνει ένα εικονικό πρότυπο για τη βελτιστοποίηση του μεσαίου μεγέθους μη γραμμικών μοντέλων. Η Mathworks δημοσίευσε μια τεχνική σημείωση [48] όπου συνέκρινε το Levenberg-Marquardt με άλλους κοινώς χρησιμοποιούμενους αλγόριθμους βελτιστοποίησης χρησιμοποιώντας έναν αριθμό συνόλων

δεδομένων δοκιμής. Κατέληξε στο συμπέρασμα πως ο εν λόγω αλγόριθμος έχει γενικά την ταχύτερη σύγκλιση με χαμηλότερα μέσα τετραγωνικά σφάλματα από οποιονδήποτε από τους άλλους αλγόριθμους που δοκιμάστηκαν σε πολλές περιπτώσεις αναφορικά με προβλήματα προσέγγισης συναρτήσεων για δίκτυα που περιέχουν σχετικά μικρό αριθμό βαρών. Κάτι τέτοιο κρίνεται ιδιαίτερα συμφέρον όταν απαιτείται πολύ ακριβής εκπαίδευση. Ωστόσο, το πλεονέκτημα του Levenberg-Marquardt μπορεί να μειώνεται καθώς αυξάνεται ο αριθμός των βαρών στο δίκτυο ενώ η απόδοσή του δεν είναι ιδιαίτερα καλή όσων αφορά σε προβλήματα αναγνώρισης προτύπων. Επιπροσθέτως οι απαιτήσεις αποθήκευσης μνήμης του LM είναι μεγαλύτερες από άλλους δοκιμασμένους αλγόριθμους.[49]

Γενικότερα, Η μέθοδος Levenberg-Marquardt απαιτεί μια αντιστροφή πίνακα που είναι υπολογιστικά ακριβή και καταναλώνει σημαντική ποσότητα μνήμης, περιορίζοντας την εφαρμογή της σε μοντέλα μεσαίου μεγέθους. Ωστόσο αξίζει να σημειωθεί πως το σύγχρονο υπολογιστικό υλικό, με σημαντική υπολογιστική ικανότητα και μνήμη, έχει επεκτείνει το εύρος των μοντέλων όπου μπορεί να χρησιμοποιηθεί ο συγκεκριμένος αλγόριθμος.[43]

5.2.2.3 Levenberg-Marquardt για MSE

Για την εκπαίδευση τόσο του μοντέλου νευρωνικού δικτύου που αναπτύχθηκε σε Matlab όσο και για εκείνου του μοντέλου που αναπτύχθηκε σε Python με τη βοήθεια Keras και TensorFlow, χρησιμοποιήθηκε ο αλγόριθμος οπίσθιας διάδοσης Levenberg-Marquardt ενώ ως μέτρο απόδοσης του μοντέλου και στις δύο περιπτώσεις χρησιμοποιήθηκε το μέσο τετραγωνικό σφάλμα(MSE) κατά τη φάση της εκπαίδευσης. Ο κύριος στόχος ήταν η ελαχιστοποίηση της συνάρτησης απόδοσης η οποία ορίζεται ως:

$$MSE = \frac{1}{n} \cdot \sum_{k=1}^{1} e_k^2 = \frac{1}{n} \sum_{k=1}^{1} (t_k - y_k)^2$$

(13)

όπου n δηλώνεται ο αριθμός των δειγμάτων, e_k είναι το σφάλμα του νευρωνικού δικτύου, t_k είναι οι τιμές των στόχων και y_k είναι οι τιμές εξόδων του δικτύου. Τα βάρη του δικτύου μεταβάλλονται βάση της εξίσωσης:

$$w_{k+1} = w_k - (J_k^T J_k + \lambda I)^{-1} \cdot J_k e_k \quad (14)$$

Η οποία βασίζεται στην προσέγγιση του Hessian Matrix:

$$H = JJ^T + \lambda I \qquad (15)$$

όπου το J δηλώνει το Jacobian matrix, το I δηλώνει πίνακα ταυτότητας(identity matrix) και το λ είναι πάντα θετικό και αναφέρεται στο ρυθμό εκμάθησης. Ο όρος $J_k e_k$ αντιπροσωπεύει το gradient.[50]

Ο αλγόριθμος Levenberg-Marquardt επιλέχθηκε στη συγκεκριμένη περίπτωση καθώς είναι

σταθερός, γρήγορος και αξιόπιστος.

Μέχρι στιγμής, πέρα από τη διαφορά στον τρόπο φόρτωσης τον δεδομένων στην εκάστοτε υλοποίηση δεν υπάρχει περαιτέρω διαφοροποίηση ανάμεσα στον κώδικα σε Matlab και σε αυτόν σε python.Ωστόσο το γεγονός αυτό μεταβάλλεται στη συνέχεια καθώς πρόκειται να παρουσιαστεί ο τρόπος με τον οποίο διαρθρώνεται και δομείται το κάθε μοντέλο νευρωνικού δικτύου ενώ ιδιαίτερη έμφαση θα δοθεί στον τρόπο με τον οποίο επιτελείται η κάθε μια διαδικασία εκπαίδευσης.

5.2.3 Δημιουργία Νευρωνικού Δικτύου Matlab

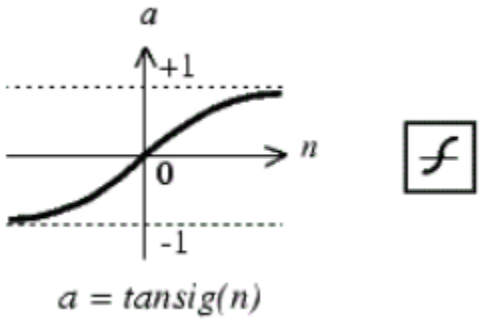
Όσων αφορά στην υλοποίηση σε Matlab για τη δημιουργία του μοντέλου του νευρωνικού δικτύου χρησιμοποιείται η συνάρτηση newff η οποία ουσιαστικά δημιουργεί ένα νευρωνικό δίκτυο οπίσθιας διάδοσης-ευθείας τροφοδοσίας(feed-forward neural network). Ο πλήρης ορισμός της είναι : newff(PR,[S1 S2...SNi],{TF1 TF2...TFNi},BTF,BLF,PF) και ως :

- PR ορίζεται ένας πίνακας διαστάσεων Rx2 ο οποίος περιέχει τις ελάχιστες και μέγιστες τιμές εισόδου για R στοιχεία εισόδου,
- Si ορίζεται το μέγεθος κάθε στρώματος του νευρωνικού δικτύου(αριθμός νευρώνων κάθε στρώματος) ενώ Ni είναι ο αριθμός των στρωμάτων του νευρωνικού δικτύου,
- Tfi ορίζεται η εκάστοτε συνάρτηση ενεργοποίησης για κάθε στρώμα του νευρωνικού δικτύου,
- BTF ορίζεται ο backpropagation αλγόριθμος εκπαίδευσης νευρωνικού δικτύου,
- BLF ορίζεται η συνάρτηση βελτιστοποίησης του νευρωνικού δικτύου(Backpropagation weight/bias learning function)
- PF ορίζεται η συνάρτηση απόδοσης του νευρωνικού δικτύου[38]

Όσων αφορά στη συγκεκριμένη εργασία, η τιμές του PR κυμαίνονται από -0.9 ως 0.9 όπως άλλωστε αναλύθηκε και νωρίτερα, τα S1, S2 είναι 8 και 2 αντίστοιχα και συναρτήσεις ενεργοποίησης είναι οι tansig και purlin αντίστοιχα. Ο αλγόριθμος εκπαίδευσης του νευρωνικού δικτύου είναι ο Levenberg-Marquardt, η συνάρτηση βελτιστοποίησης είναι η συνάρτηση gradient descent ενώ εν κατακλείδι η συνάρτηση μέσω της οποίας αποδίδεται η απόδοση του δικτύου είναι η συνάρτηση mean squared error.

Tansig (Hyperbolic tangent sigmoid transfer function)

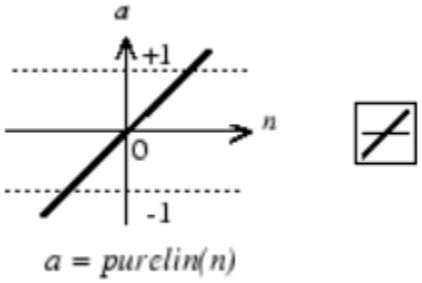
Η tansig δέχεται σαν είσοδο Ν έναν πίνακα διανυσμάτων εισόδου και επιστρέφει έναν πίνακα Α των στοιχείων του Ν, συμπιεσμένων μεταξύ -1 και 1.[39]



Εικόνα 17 – Tansig

Purelin (Linear transfer function)

Η purelin δέχεται έναν πίνακα Ν διανυσμάτων εισόδου και επιστρέφει έναν πίνακα Α ίσων διαστάσεων.[40]



Εικόνα 18 – Purelin

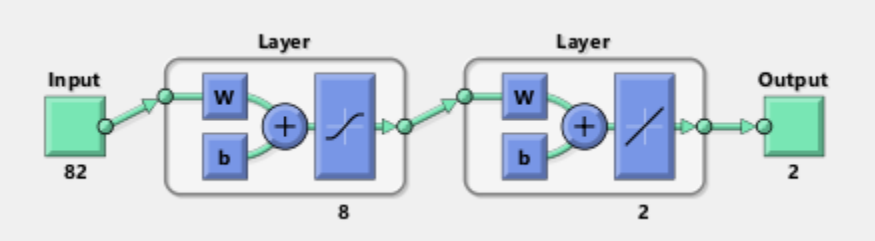
Mean Squared Error Function

Στις στατιστική, το μέσο τετράγωνο σφάλμα (MSE) μιας διαδικασίας για την εκτίμηση μιας μη παρατηρούμενης ποσότητας, μετρά τον μέσο όρο των τετραγώνων των σφαλμάτων—δηλαδή τη μέση τετραγωνική διαφορά μεταξύ των εκτιμώμενων τιμών και της πραγματικής αξίας. Το γεγονός ότι το MSE είναι σχεδόν πάντα αυστηρά θετικό (και όχι μηδενικό) οφείλεται στην τυχαιότητα ή στο ότι ο εκτιμητής δεν λαμβάνει υπόψη πληροφορίες που θα μπορούσαν να παράγουν μια πιο ακριβή εκτίμηση.[88]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(12)

Όπου MSE είναι το μέσο τετραγωνικό σφάλμα, n ο αριθμός των data points, y_i οι τιμές-στόχοι, και $\hat{y}_i$ οι τιμές που προβλέπει το νευρωνικό δίκτυο.
Σχήμα Μοντέλου Νευρωνικού Δικτύου (Matlab):



Εικόνα 19 – Νευρωνικό Δίκτυο Matlab

5.2.4 Δημιουργία Νευρωνικού Δικτύου Python

Για τη δημιουργία του μοντέλου νευρωνικού δικτύου κατά την υλοποίηση σε Python χρησιμοποιήθηκε το Sequential Model του Keras. Το Sequential model API, είναι ένας τρόπος δημιουργίας μοντέλων βαθιάς μάθησης όπου δημιουργείται ένα instance της κλάσης Sequential ενώ δημιουργούνται και προστίθενται σε αυτό layers. Αναλυτικότερα, Το Sequential model API είναι εξαιρετικό για την ανάπτυξη μοντέλων βαθιάς μάθησης στις περισσότερες περιπτώσεις, ωστόσο έχει επίσης ορισμένους περιορισμούς. Για παράδειγμα, δεν είναι απλός ο ορισμός μοντέλων που μπορεί να έχουν πολλές διαφορετικές πηγές εισόδου, ωστόσο στην περίπτωσή μας κάτι τέτοιο δεν αποτελεί τροχοπέδη οπότε και το Sequential model API καλύπτει πλήρως τις ανάγκες μας.

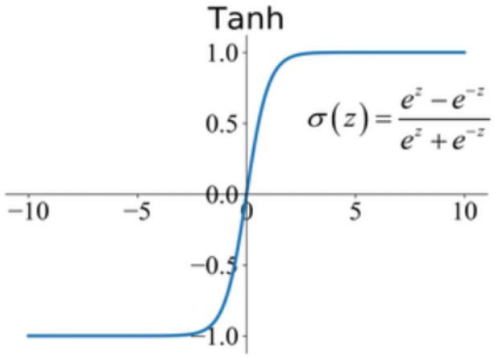
Για τον πλήρη ορισμό του μοντέλου είναι απαραίτητος και ο ορισμός instances των layers που το απαρτίζουν, τα οποία συνδέονται μεταξύ τους. Από αυτά τα layers το πρώτο λειτουργεί ως είσοδος στο μοντέλο, το τελευταίο λειτουργεί ως έξοδος ενώ τα ενδιάμεσα αποτελούν τα hidden layers του μοντέλου.[51]

Αναλυτικότερα, όσων αφορά στη δημιουργία layers χρησιμοποιείται η κλάση Dense. Η Dense υλοποιεί τη λειτουργία output = activation (dot(input, kernel)), όπου ως activation ορίζεται η συνάρτηση ενεργοποίησης , ως kernel ορίζεται ένας πίνακας βαρών και ως input ορίζεται ένας πίνακας που αποτελούν δεδομένα εισόδου.[52] Ο όρος bias δεν αναφέρεται καθώς στην περίπτωσή μας δεν χρησιμοποιείται.

Πιο συγκεκριμένα όσων αφορά στην υλοποίηση σε κώδικα Python, το layer εισόδου έχει input_shape 82, δηλαδή διαθέτει 82 εισόδους και το layer εξόδου διαθέτει 2 κόμβους οπότε κατανοούμε πως οι έξοδοί μας είναι 2, όπως ακριβώς συμβαίνει και κατά την υλοποίηση σε Matlab. Όσων αφορά στα hidden layers συνολικά είναι δύο και πάλι όπως κατά την υλοποίηση σε matlab. Το πρώτο hidden layer έχει 27 νευρώνες και η activation function του είναι η tanh ενώ το δεύτερο hidden layer έχει 9 νευρώνες και η activation function του είναι και πάλι η tanh. Ο αλγόριθμος εκπαίδευσης του νευρωνικού δικτύου είναι ο Levenberg-Marquardt, η συνάρτηση βελτιστοποίησης είναι η συνάρτηση gradient descent ενώ εν κατακλείδι η συνάρτηση μέσω της οποίας αποδίδεται η απόδοση του δικτύου είναι η συνάρτηση mean squared error.

Tanh activation function(hyperbolic tangent activation function)

Η Tanh μοιάζει πολύ με τη συνάρτηση ενεργοποίησης σιγμοειδούς(Sigmoid) και μάλιστα έχει το ίδιο σχήμα S. Παίρνει οποιαδήποτε πραγματική τιμή ως είσοδο και εξάγει τιμές στην περιοχή -1 έως 1. Όσο μεγαλύτερη είναι η είσοδος (πιο θετική), τόσο πιο κοντά η τιμή εξόδου θα είναι στο 1, ενώ όσο μικρότερη είναι η είσοδος (πιο αρνητική), τόσο πιο κοντά η έξοδος θα είναι -1.[53]



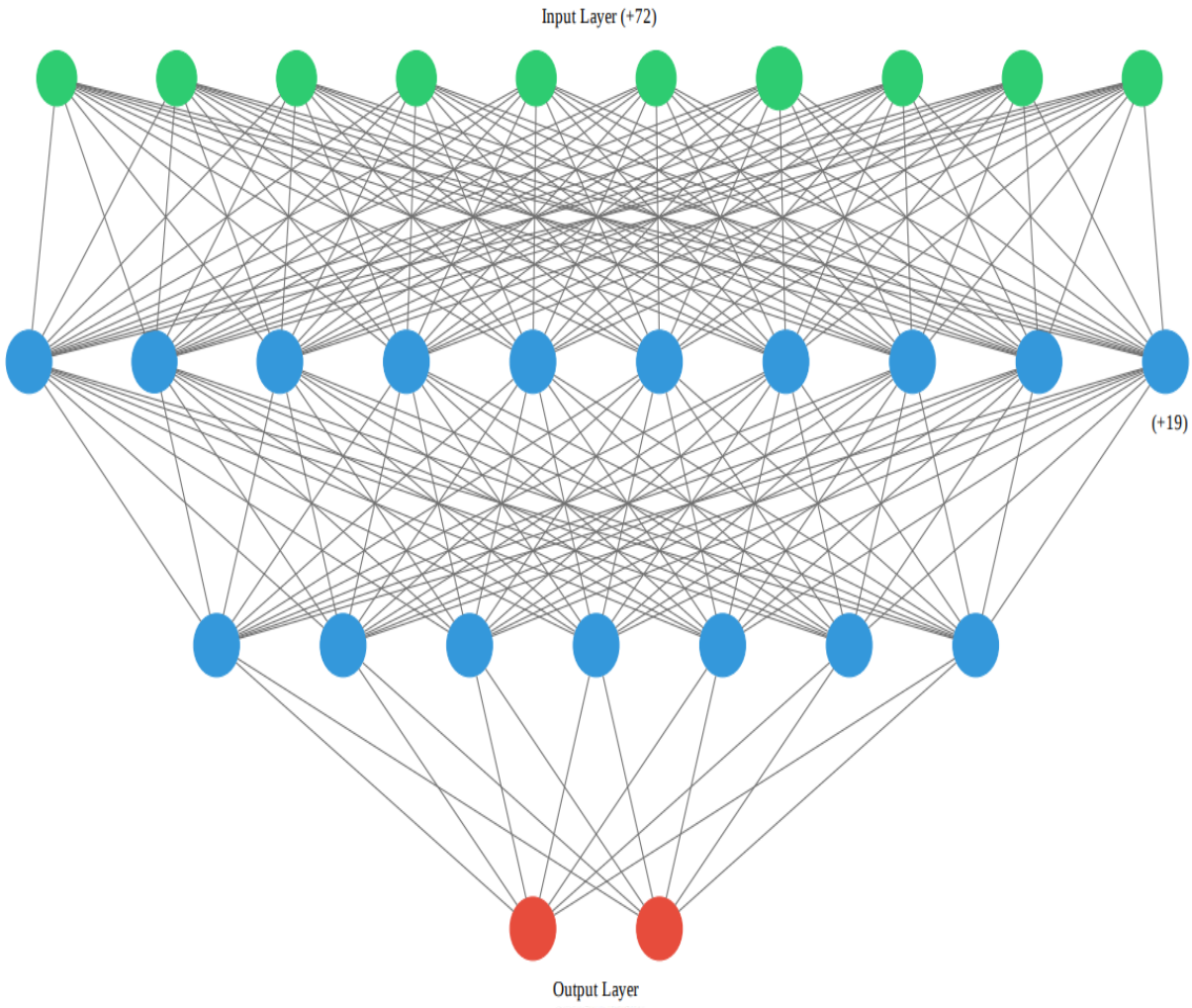
Εικόνα 20 – Tanh

Η συνάρτηση ενεργοποίησης Tanh υπολογίζεται ως εξής:

$$Tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (16)$$

Σχήμα Μοντέλου Νευρωνικού Δικτύου (Python):

Για την οπτικοποίηση του σχήματος του μοντέλου νευρωνικού δικτύου σε Python χρησιμοποιήθηκε το ANN Visualizer. Πρόκειται για μια βιβλιοθήκη, η οποία μας δίνει τη δυνατότητα να οπτικοποιήσουμε ένα Τεχνητό Νευρωνικό Δίκτυο χρησιμοποιώντας ελάχιστες γραμμές κώδικα. Χρησιμοποιείται για εργασία με το Keras ενώ χρησιμοποιεί τη βιβλιοθήκη Graphviz της Python προκειμένου να δημιουργήσει ένα τακτοποιημένο και εμφανίσιμο γράφημα του νευρωνικού δικτύου που δημιουργούμε.[83] Από την άλλη το Graphviz αποτελεί μονάδα ανοιχτού κώδικα, που χρησιμοποιείται για τη δημιουργία αντικειμένων γραφήματος που μπορούν να ολοκληρωθούν χρησιμοποιώντας διαφορετικούς κόμβους και ακμές. Βασίζεται στη γλώσσα DOT του λογισμικού Graphviz και στην Python μας επιτρέπει να κατεβάσουμε τον πηγαίο κώδικα του γραφήματος σε γλώσσα DOT.[84]



Εικόνα 21 – Artificial Neural Network

5.2.5 Διαφοροποίηση κώδικα Python ως προς τον ορισμό του Μοντέλου Νευρωνικού Δικτύου

Σε αυτό το σημείο παρατηρείται και η πρώτη βασική διαφορά μεταξύ των δύο υλοποιήσεων η οποία έγκειται στον αριθμό των κόμβων των hidden layers αλλά και στις συναρτήσεις ενεργοποίησης αυτών.

5.2.5.1 Τοπολογικά Χαρακτηριστικά Νευρωνικού Δικτύου

Γενικότερα, η ουσία οποιασδήποτε διένεξης σε σχέση με τα τοπολογικά όρια των νευρωνικών δικτύων είναι πιθανότατα το θεώρημα του Kolmogorov, το οποίο δηλώνει πως κάθε συνεχής συνάρτηση που ορίζεται σε έναν κύβο διαστάσεων n μπορεί να παρασταθεί με αθροίσματα και υπερθέσεις συνεχών συναρτήσεων μιας μεταβλητής.[54] Ο Hecht-Nielsen [55] εισήγαγε αυτό το θεώρημα αργότερα στη νευρουπολογιστική αποδεικνύοντας ότι οποιαδήποτε συνεχής συνάρτηση μπορεί να αναπαρασταθεί από ένα νευρωνικό δίκτυο που έχει μόνο ένα κρυφό επίπεδο με ακριβώς 2n + 1 κόμβους, όπου n είναι ο αριθμός κόμβων εισόδου. Αρκετοί χρησιμοποιούν το σχήμα αυτό ως πανάκεια ωστόσο η ανάγκη καλύτερων μεθόδων ανακύπτει

λόγο της πολυπλοκότητας των προβλημάτων που τίθενται προς επίλυση. Άλλωστε κατά τον Hecht-Nielsen ο εν λόγω κανόνας εμπίμπτει μόνο σε συγκεκριμένες κατηγορίες νευρωνικών δικτύων.[55]

Ύστερα, όσων αφορά στη χρήση σιγμοειδών συναρτήσεων ως συναρτήσεων ενεργοποίησης σε μοντέλα νευρωνικών δικτύων έχει προταθεί[56] πως είναι ορθό να χρησιμοποιούνται δύο κρυφά επίπεδα(hidden layers) προκειμένου να αντισταθμίζεται η χαμένη απόδοση. Φυσικά μπορεί να χρησιμοποιηθεί και μόνο ένα κρυφό στρώμα ωστόσο ο αριθμός των κόμβων αυξάνεται σημαντικά.[57] Ο σκοπός της χρήσης ενός δεύτερου κρυφού στρώματος είναι να μειωθεί δραστικά το σύνολο των απαιτούμενων κόμβων. Ο Huang απέδειξε πως στην περίπτωση όπου έχουμε ένα μοντέλο νευρωνικού δικτύου με δύο κρυφά επίπεδα, m νευρώνες εισόδου και N δείγματα για εκμάθηση, ο αριθμός των κόμβων που είναι αρκετοί για να έχουμε αποτέλεσμα με το ελάχιστο δυνατό σφάλμα δίνεται από τη σχέση[58]:

$$2 \cdot \sqrt{(m+2) \cdot N} \qquad (17)$$

Συγκεκριμένα προτείνει ο επαρκής αριθμός κρυφών κόμβων στο πρώτο κρυφό στρώμα να είναι:

$$\sqrt{(m+2) \cdot N} + 2 \cdot \sqrt{\frac{N}{(m+2)}} \quad (18)$$

και στο δεύτερο:

$$m \cdot \sqrt{\frac{N}{(m+2)}} \quad (19)$$

Τέλος, εφόσον η βέλτιστη τοπολογία ενός νευρωνικού δικτύου κρίνεται από την ικανότητα γενίκευσης σε δεδομένα τα οποία είναι άγνωστα, μια πιο ακριβής δομή θα έχει σίγουρα το πολύ τόσους κόμβους όσους αυτοί που προτείνονται από τις προαναφερθείσες εξισώσεις. Συνοπτικά πρέπει να αναζητούνται τοπολογίες όπου ο αριθμός των κρυφών κόμβων θα είναι στη χειρότερη αυτός ο οποίος προτείνεται από τις εξισώσεις.[59]

Προφανώς ο αριθμός των κόμβων των κρυφών επιπέδων κατά την πρώτη υλοποίηση βρίσκεται μέσα στα επιτρεπτά όρια, ωστόσο η μέθοδος trial and error επιτρέπει τον πειραματισμό προκειμένου να βρεθούν νέα πιο αποτελεσματικά τοπολογικά δεδομένα. Ως εκ τούτου ορίστηκαν νέοι αριθμοί κόμβων κάθε κρυφού στρώματος.

Επιπλέον κρίνεται ορθό να αναφερθεί πως η έλλειψη συνάρτησης ενεργοποίησης στο layer εξόδου σημαίνει ότι η έξοδος του νευρωνικού δικτύου είναι απλώς ένας γραμμικός συνδυασμός των εισόδων από το προηγούμενο επίπεδο.[60]

5.2.5.2 Συναρτήσεις ενεργοποίησης

Όσων αφορά στις συναρτήσεις ενεργοποίησης, αποτελούν κρίσιμο μέρος του σχεδιασμού ενός νευρωνικού δικτύου καθώς η επιλογή τους ελέγχει πόσο καλά το μοντέλο του νευρωνικού

δικτύου μαθαίνει από τα δεδομένα εκπαίδευσης. Δεδομένου του ότι το Keras δεν προσφέρει την επιλογή των συναρτήσεων tansig και purlin υπήρξε η ανάγκη αντικαταστάσής τους. Η tanh κρίθηκε κατάλληλη καθώς είναι γνωστό πως είναι αρκετά καλή για δεδομένα τα οποία είναι κανονικοποιημένα στο διάστημα από -1 ως 1. Επιπλέον από τη βιβλιογραφία καθίσταται προφανές πως ένα νευρωνικό δίκτυο σχεδόν πάντα χρησιμοποιεί την ίδια συνάρτηση ενεργοποίησης σε όλα του τα κρυφά επίπεδα. Παραδοσιακά, η συνάρτηση σιγμοειδούς ενεργοποίησης ήταν η προεπιλεγμένη συνάρτηση ενεργοποίησης τη δεκαετία του 1990. Ωστόσο, από τα μέσα έως τα τέλη της δεκαετίας του 1990 έως τη δεκαετία του 2010, η συνάρτηση Tanh έγινε η προεπιλεγμένη λειτουργία ενεργοποίησης για κρυφά επίπεδα στις περισσότερες περιπτώσεις.[53] Επιπλέον, η tanh συνήθως αποδίδει καλύτερα σε σχέση με τη sigmoid.[42]

5.2.5.3 Εφαρμογή αλγορίθμου Levenberg-Marquardt

Σε ότι αφορά στην επιλογή μεθόδου υπολογισμού του gradient ο αλγόριθμος Levenberg-Marquardt εφαρμόστηκε όπως ακριβώς και κατά την υλοποίηση σε Matlab. Εδώ ωστόσο η διαφορά ανάμεσα στις δύο υλοποιήσεις έγκειται στον τρόπο της εφαρμογής του καθώς το Keras δεν διαθέτει by default τον αλγόριθμο αυτό. Συνεπώς χρησιμοποιήθηκε η βιβλιοθήκη “Tensorflow Levenberg-Marquardt” η οποία εξυπηρετεί τη χρήση του εν λόγω αλγορίθμου σε Sequential Keras Models[61]. Προκειμένου να γίνει ορθή εφαρμογή της βιβλιοθήκης αυτής δημιουργήθηκε ένα instance της κλάσης Model Wrapper το οποίο βασίζεται στο Sequential Keras Model το οποίο ορίστηκε νωρίτερα.

5.2.6 Περεταίρω Στοιχεία του Κώδικα Python

5.2.6.1 Compile Function

Στη συνέχεια χρησιμοποιείται η συνάρτηση compile του Keras η οποία ουσιαστικά διαμορφώνει το μοντέλο για εκπαίδευση.[28] Αναλυτικότερα, χάρη σε αυτή ορίζεται η συνάρτηση απώλειας(loss function), ο optimizer και τα metrics και είναι απολύτως απαραίτητη για την εκπαίδευση, για την οποία θα γίνει λόγος παρακάτω, εφόσον η συνάρτηση απώλειας και ο optimizer αποτελούν θεμελιώδη στοιχεία της εκπαιδευτικής διαδικασίας.

5.2.6.2 Loss Function

Στην περίπτωσή μας, ως Loss Function χρησιμοποιείται η MSE(Mean Squared Error) η οποία υπολογίζει τον μέσο όρο των τετραγώνων των σφαλμάτων μεταξύ στόχων και προβλέψεων.[62] Τέλος, σκοπός της συνάρτησης απώλειας είναι ο υπολογισμός της ποσότητας που ένα μοντέλο πρέπει να επιδιώξει να ελαχιστοποιήσει κατά τη διάρκεια της εκπαίδευσης[63], ενώ κρίνεται ορθό να αναφερθεί πως η υλοποίηση της μας παρέχεται από το Keras.

5.2.6.3 Optimizer (Stochastic Gradient Descent)

Ως optimizer χρησιμοποιήθηκε ο SGD (Stochastic Gradient Descent). Γενικά, ο optimizer είναι μια συνάρτηση που τροποποιεί τα χαρακτηριστικά του νευρωνικού δικτύου, όπως πχ τα βάρη. Έτσι, βοηθά στη μείωση της συνολικής απώλειας και στη βελτίωση της ακρίβειας του μοντέλου νευρωνικού δικτύου[64].

Το Stochastic Gradient Descent είναι ένας από τους πολλούς αλγορίθμους βελτιστοποίησης που μπορούν να χρησιμοποιηθούν για την εκπαίδευση μοντέλων νευρωνικών δικτύων.Ο SGD optimizer που μας παρέχει το Keras ανανεώνει τα βάρη, βάση του momentum το οποίο είναι float υπερπαράμετρος του μοντέλου του νευρωνικού δικτύου και επιταχύνει το gradient descent κατά την αντίστοιχη κατεύθυνση. Επίσης χάρη στο momentum μειώνονται οι έντονες αυξομοιώσεις της τιμής του loss. Η τιμή του momentum στην περίπτωσή μας είναι 0 και τα βάρη υπολογίζονται από τη σχέση:

$$w = w - learning\_rate \cdot g \quad (20)$$

όπου w είναι τα βάρη, learning_rate ο ρυθμός εκμάθησης και g το gradient.[65]

5.2.6.4 Ρυθμός Εκμάθησης (Learning Rate)

Στη μηχανική μάθηση, ο ρυθμός εκμάθησης είναι μια παράμετρος συντονισμού σε έναν αλγόριθμο βελτιστοποίησης που καθορίζει το μέγεθος του βήματος σε κάθε επανάληψη, ενώ κινείται προς την ελάχιστη συνάρτηση απώλειας.[66] Κατά την υλοποίηση σε Python, το learning rate τέθηκε ίσο με 1 εν αντιθέσει με την υλοποίηση σε Matlbam όπου είναι ίσο με 0.2 .

Όπως ήδη αναφέρθηκε ο ρυθμός εκμάθησης ελέγχει πόσο γρήγορα το μοντέλο νευρωνικού δικτύου προσαρμόζεται στο πρόβλημα. Μικρότερα ποσοστά απαιτούν περισσότερα epochs λόγω των μικρών αλλαγών που γίνονται στα βάρη σε κάθε ενημέρωση, ενώ αντιστοίχως μεγαλύτερα ποσοστά οδηγούν σε μικρότερο αριθμό επαναλήψεων. Ένας πολύ μεγάλος ρυθμός μάθησης μπορεί να προκαλέσει τη σύγκλιση του μοντέλου πολύ γρήγορα σε μια μη βέλτιστη λύση, ενώ ένας πολύ μικρός ρυθμός μάθησης μπορεί να προκαλέσει κόλλημα της διαδικασίας εκπαίδευσης[67].

Στην περίπτωσή μας τόσο κατά την υλοποίηση σε Python όσο και κατά την υλοποίηση σε Matlab οι τιμές που χρησιμοποιήθηκαν για το ρυθμό εκμάθησης είναι αποδεκτές εφόσον το αποτέλεσμα είναι επαρκές. Ωστόσο εκμεταλλευόμενοι την ισχύ που προσφέρει το Keras σε συνδιασμό με το TensorFlow κατέστη εφικτό να μειωθεί το περισσόρο δυνατόν ο αριθμός των επαναλήψεων κατά τη διαδικασία εκπαίδευσης άρα και να μειωθεί ο χρόνος που αυτή απαιτεί.

5.2.7 Διαδικασία Εκπαίδευσης

5.2.7.1 Υλοποίηση Matlab

Κατά την υλοποίηση σε Matlab χρειάστηκε ακόμα να οριστούν ορισμένες παράμετροι πριν την κλήση της συνάρτησης εκπαίδευσης. Πρόκειται για το ρυθμό εκπαίδευσης ο οποίος ορίστηκε ίσος με 0.2, το μέγιστο αριθμό επαναλήψεων ο οποίος τέθηκε ίσος με 5000 αλλά και το στόχο απόδοσης του οποίου η τιμή ορίστηκε στο 0.0001 . Δεδομένης της απόδοσης τιμής στον στόχο απόδοσης γίνεται κατανοητό πως εφόσον αυτός κατακτηθεί σταματά αυτομάτως και η διαδικασία εκπαίδευσης του νευρωνικού δικτύου.

Η συνάρτηση που κλήθηκε είναι η train η οποία παρέχεται από τη Matlab και εκπαιδεύει το μοντέλο του νευρωνικού δικτύου που ορίστηκε βάση του αλγορίθμου gradient descent όπως αναφέρθηκε προηγουμένως. Σαν εισόδους δέχεται το μοντέλο του νευρωνικού δικτύου που ορίστηκε νωρίτερα, τα δεδομένα εισόδου εκπαίδευσης και τα δεδομένα-στόχους εκπαίδευσης. Στον αντίποδα, επιστρέφει ένα νέο εκπαιδευμένο νευρωνικό δίκτυο αλλά και ένα αρχείο το οποίο αφορά στη διαδικασία της εκπαίδευσης.[68]

Κατά την εκπαίδευση σε Matlab αξιοσημείωτη είναι η ύπαρξη του nntraintool το οποίο αποτελεί μια γραφική διεπαφή η οποία δείχνει τη διαδικασία εκπαίδευσης ενώ επίσης προσφέρει τη δυνατότητα εξαγωγής σχετικών διαγραμμάτων με το πάτημα ενός κουμπιού. Μια τέτοια δυνατότητα δεν προσφέρεται από το Keras σε συνδιασμό με το το TensorFlow. Η ικανότητα της python όμως να δέχεται πολλές διαφορετικές βιβλιοθήκες, μας επιτρέπει να εξάγουμε εύκολα και γρήγορα τα ανάλογα αποτελέσματα.

5.2.7.2 Υλοποίηση Python

Κατά την υλοποίηση σε Python χρησιμοποιήθηκε η μέθοδος fit η οποία είναι μέρος του Keras Models API. Σαν εισόδους δέχεται ένα TensorFlow DataSet το οποίο περιέχει το σύνολο των δεδομένων εκπαίδευσης, τα μέγιστο αριθμό επαναλήψεων ο οποίος επιλέχθηκε να είναι ίσος με αυτόν κατά την υλοποίηση σε Matlab(5000) και μια callback συνάρτηση η οποία επενεργεί στη διαδικασία εκμάθησης του νευρωνικού δικτύου.

Στην περίπτωσή μας, δημιουργήθηκε μια νέα custom συνάρτηση η οποία παρακολουθώντας το μέγεθος loss κατά τη διαδικασία της εκπαίδευσης είναι ικανή να σταματήσει τη διαδικασία αυτή εφόσον η τιμή του loss γίνει ίση ή μικρότερη από 0.00001(μικρότερη από την υλοποίηση σε Matlab). Επιπλέον δεδομένου του ότι τα δεδομένα εισόδου τροφοδοτούνται στο δίκτυο με τη μορφή TensorFlow Dataset, batch_size(αριθμός των παραδειγμάτων εκπαίδευσης που χρησιμοποιούνται σε κάθε επανάληψη) δεν ορίζεται καθώς του αποδίδεται αυτόματα τιμή.[69]

Επιπροσθέτως, χάρη στη χρήση της βιβλιοθήκης “Tensorflow Levenberg-Marquardt” μας δίνεται η επιλογή να θέσουμε μια ακόμα παράμετρο προς αποφυγή overfitting κατά τη διαδικασία της εκπαίδευσης. Κατ'αυτόν τον τρόπο, επιβάλουμε το πέρας της διαδικασίας της εκπαίδευσης εφόσον η τιμή του loss δε μειωθεί για δέκα συνεχόμενα epochs. Κάτι τέτοιο μειώνει με τον τρόπο του τις επαναλήψεις και αποτρέπει απο overfitting καθώς εφόσον η διαδικασία δεν παρατείνεται επ άπειρο χωρίς αποτελέσματα είναι απίθανο να απομνημονευτούν τα δεδομένα εισόδου εκπαίδευσης. Ως overfitting ορίζεται το γεγονός πως ένα νευρωνικό δίκτυο μαθαίνει τόσο καλά τα δεδομένα εκπαίδευσης ώστε η απόδοσή του σε δεδομένα με τα οποία δεν έχει έρθει ξανά σε επαφή είναι πολύ κακή παρόλο που πολλές φορές η τιμή του loss φαίνεται ιδιαιτέρως μικρή.[70]

5.2.8 Πρόβλεψη Αποτελεσμάτων

5.2.8.1 Υλοποίηση σε Matlab

Για την πρόβλεψη των αποτελεσμάτων κατά την υλοποίηση σε Matlab χρησιμοποιείται συνάρτηση sim η οποία παρέχεται από την ίδια τη Matlab και δέχεται σαν εισόδους το

εκπαιδευμένο νευρωνικό δίκτυο ,για το οποίο έγινε λόγος πρωτύτερα, και τα δεδομένα ελέγχου εισόδου. Επιστρέφει τις προβλέψεις του νευρωνικού δικτύου για τα συγκεκριμένα δεδομένα εισόδου οι οποίες ωστόσο χρήζουν denormalization προκειμένου να λάβουμε τα τελικά αποτελέσματα και να μπορέσουμε να ελέγχσουμε την αξιοπιστία του του νευρωνικού μας δικτύου.

5.2.8.2 Υλοποίηση σε Python

Εδώ και πάλι η διαδικασία δε διαφέρει σχεδόν καθόλου με μοναδική παραλλαγή τη χρήση της μεθόδου predict που μας παρέχει το Keras API. Σαν είσοδο δέχεται και αυτή τα test δεδομένα εισόδου από το dataset μας και έξοδό της αποτελούν οι προβλέψεις των στόχων. Η διαδικασία denormalization κρίνεται και πάλι αναγκαία.

5.2.9 Data Denormalization

Η διαδικασία του denormalization είναι απαραίτητη εφόσον έχει προηγηθεί αυτή της κανονικοποίησης.[36] Πιο συγκεκριμένα, συντελείται η ακόλουθη διαδικασία προκειμένου τα δεδομένα μας(είναι κανονικοποιημένα ανάμεσα στο -0.9 και στο 0.9) να λάβουν τις νέες τιμές τους οι οποίες αντιστοιχούν στην πραγματικότητα:

$$\frac{ValueNormalized-(-0.9)}{0.9-(-0.9)}=\frac{ValueDenormalized-min}{max-min}$$

$$\frac{ValueNormalized+0.9}{1.8}=\frac{ValueDenormalized-min}{max-min}$$

$$ValueDenormalized=(\max-min)\cdot\frac{ValueNormalized+0.9}{1.8}+min$$

(21)

Όσων αφορά στη σχέση (21) προκύπτει από την τριγωνομετρία ομοίως με τη σχέση (11) του κεφαλαίου 5.2.1.3. Σχήμα δεν παρατίθεται εφόσον πρόκειται για την αντίστροφη διαδικασία της σχέσης που μοντελοποιείται στην εικόνα 16 του κεφαλαίου 5.2.1.3.

5.2.10 Διαδικασία Οπτικοποίησης Αποτελεσμάτων

5.2.10.1 Υλοποίηση Matlab

Για την οπτικοποίηση της θέσης των οπών που έχουν προβλεφθεί από το νευρωνικό δίκτυο κατά την υλοποίηση σε Matlab χρησιμοποιείται η συνάρτηση plot της Matlab η οποία έχει την ικανότητα να σχεδιάζει πολλαπλά ζεύγη συντεταγμένων x,y στο ίδιο σύστημα αξόνων[71].Επιπλέον χρησιμοποιείται η συνάρτηση postreg η οποία επεξεργάζεται το σύνολο δεδομένων δικτύου εκτελώντας μια γραμμική παλινδρόμηση(linear regression) μεταξύ κάθε στοιχείου της απόκρισης δικτύου και του αντίστοιχου στόχου. Σαν εισόδους δέχεται τα δεδομένα εισόδου(εκπαίδευσης ή ελέγχου) του νευρωνικού δικτύου και τα δεδομένα-στόχους, ενώ σαν

έξοδοι προκύπτουν τα M,B,R με M την κλήση της γραμμικής παλινδρόμησης,B το intercept της γραμμικής παλινδρόμησης και R το regression value.Όσο πιο κοντά είναι το R στο 1, τόσο καλύτερη είναι η συσχέτιση μεταξύ των δεδομένων που μελετάμε. R=1 σημαίνει τέλεια συσχέτιση.[72] Τα αποτελέσματα θα παρατεθούν στο επόμενο κεφάλαιο.

5.2.10.2 Υλοποίηση Python

Προκειμένου να σχεδιαστούν οι θέσεις των οπών οι οποίες έχουν προβλεφθεί από το μοντέλο νευρωνικού δικτύου σε Python χρησιμοποιήθηκε η συνάρτηση plt της βιβλιοθήκης matplotlib η οποία αποτελεί ολοκληρωμένη λύση για τη δημιουργία στατικών, κινούμενων και διαδραστικών απεικονίσεων στην Python.[73]Η εν λόγω συνάρτηση plot πως αντίστοιχα και η plot της matlab έχει την ικανότητα να σχεδιάζει πολλαπλά ζεύγη συντεταγμένων x,y στο ίδιο σύστημα αξόνων.

Όσων αφορά στη διαδικασία οπτικοποίησης του linear regression μεταξύ των εκάστοτε στοιχείων εισόδου του νευρωνικού δικτύου συνδιαστικά με τα δεδομένα-στόχους αυτού, χρησιμοποιήθηκε η μέθοδος regplot του πακέτου seaborn.

Όπως είναι γνωστό η οπτικοποίηση δεδομένων αποτελεί αναπόσπαστο μέρος της επιστημονικής διαδικασίας της μηχανικής μάθησης διότι, αποτελεσματικές απεικονίσεις επιτρέπουν στο χρήστη τόσο να κατανοήσει τα δικά του δεδομένα όσο και να κοινοποιήσει τις γνώσεις του σε τρίτους[74]. Τέτοιοι στόχοι μπορούν να προωθηθούν μέσα στο επιστημονικό οικοσύστημα Python, το οποίο μας επιτρέπει να χρησιμοποιούμε APIs όπως το matplotlib [75] το οποίο είναι πολύ καλά εδραιωμένο, καθώς βρίσκεται υπό συνεχή λειτουργία και ανάπτυξη για σχεδόν δύο δεκαετίες. Επιπλέον, κρίνεται εξαιρετικά ευέλικτο, προσφέροντας λεπτομερή έλεγχο της τοποθέτησης αντικειμένων στο εκάστοτε διάγραμμα. Μπορεί να χρησιμοποιηθεί διαδραστικά μέσω GUI εφαρμογών και παράγει γραφικά σε ένα ευρύ φάσμα στατικών μορφών. Το matplotlib αποτελεί API χαμηλού επιπέδου όπου κανείς μπορεί να κάνει ορισμένες κοινές εργασίες δυσκίνητες στην εκτέλεση.
Η βιβλιοθήκη seaborn προσφέρει μια διεπαφή στο matplotlib επιτρέποντας έτσι την ταχεία εξερεύνηση δεδομένων και τη δημιουργία πρωτότυπων οπτικοποιήσεων διατηρώντας παράλληλα μεγάλο μέρος της ευελιξίας και της σταθερότητας που είναι απαραίτητες για την παραγωγή γραφικών ποιότητας. Είναι γενικού τομέα και μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα συνόλων δεδομένων τα οποία βρίσκονται σε μορφή πίνακα ενώ συχνά ενσωματώνει και δομές δεδομένων panda όπως και στη δική μας περίπτωση.

Όταν δίνεται ένα σύνολο δεδομένων και μια προδιαγραφή του διαγράμματος που πρόκειται να δημιουργηθεί, το seaborn αντιστοιχίζει αυτόματα τις τιμές δεδομένων σε οπτικά χαρακτηριστικά όπως χρώμα, μέγεθος ή στυλ, υπολογίζοντας εσωτερικά τους στατιστικούς μετασχηματισμούς ενώ παράλληλα διακοσμεί το διάγραμμα με ενημερωτικές ετικέτες αξόνων. Το seaborn έχει σχεδιαστεί για να είναι χρήσιμο καθ' όλη τη διάρκεια του κύκλου ζωής ενός επιστημονικού έργου. Τέλος, με την παραγωγή ολοκληρωμένων γραφικών λύσεων από μία απλή κλήση συνάρτησης με ελάχιστα ορίσματα, το seaborn διευκολύνει την ταχεία δημιουργία πρωτότυπων διαγραμμάτων και αντίστοιχα τη διερευνητική ανάλυση δεδομένων.[76]

Όπως αναφέρθηκε νωρίτερα, η βιβλιοθήκη seaborn συνεργάζεται με το Pandas προκειμένου να λάβει δεδομένα τα οποία θα οπτικοποιηθούν. Το Pandas είναι ένα πακέτο Python ανοιχτού

κώδικα το οποίο χρησιμοποιείται ευρέως για εργασίες ανάλυσης δεδομένων και μηχανικής μάθησης. Είναι χτισμένο πάνω από ένα άλλο πακέτο(on top) που ονομάζεται Numpy, το οποίο παρέχει υποστήριξη για πολυδιάστατους πίνακες[80]. Επιπλέον, όσων αφορά στα δεδομένα εισόδου που χρησιμοποιούνται από τη μέθοδο regplot αυτά είναι τύπου Pandas DataFrame. Το Pandas DataFrame είναι μια δισδιάστατη δομή δεδομένων(τα δεδομένα ευθυγραμμίζονται όπως σε πίνακα σε γραμμές και στήλες) με δυνατότητα αλλαγής μεγέθους.[81]

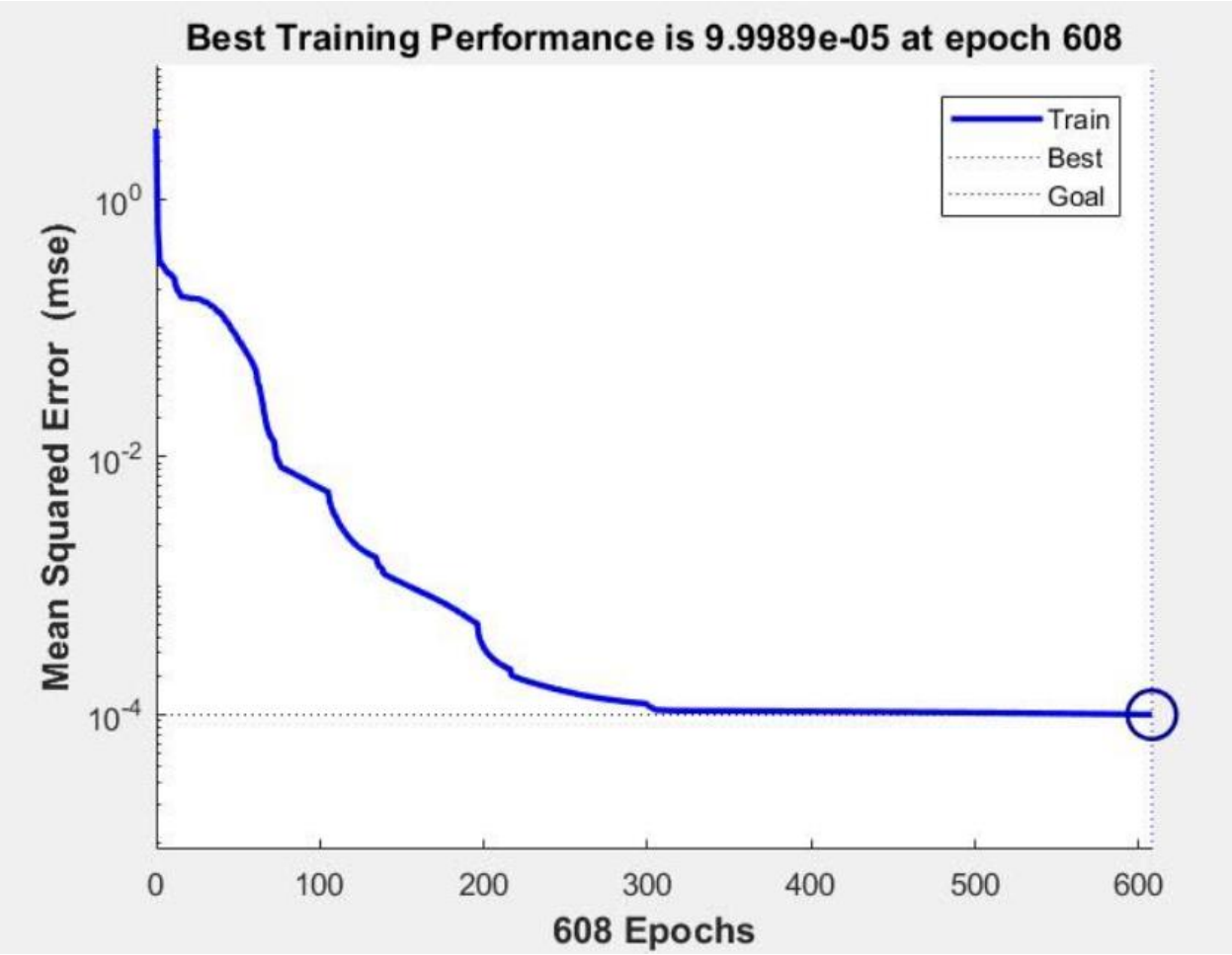
Όσων αφορά στα επιμέρους δεδομένα της οπτικοποίησης του linear regression κατά την υλοποίηση σε Python, χαρακτηριστικά όπως η εύρεση εξίσωσης γραμμής και ο αυτόματος υπολογισμός του R δεν είναι διαθέσιμα. Το 2015, ένας από τους κύριους προγραμματιστές του seaborn απάντησε σε ένα αίτημα χαρακτηριστικών που ζητούσε πρόσβαση στις στατιστικές τιμές που χρησιμοποιούνται για τη δημιουργία γραφημάτων λέγοντας: "Δεν είναι διαθέσιμο και δεν θα γίνει διαθέσιμο[82]. Ως εκ τούτου, δημιουργήθηκε custom συνάρτηση η οποία διαβάζοντας ουσιαστικά το δισδιάστατο διάγραμμα μοντελοποιεί τα σημεία της γραμμής του linear regression και κατ’ αυτόν τον τρόπο βρίσκει την εξίσωση που τη χαρακτηρίζει. Σε ότι αφορά στο R(συντελεστής προσδιορισμού), χρησιμοποιήθηκε η συνάρτηση r2_score του sklearn η οποία δέχεται σαν εισόδους τα δεδομένα-στόχους και αυτά τα οποία έχουν παραχθεί από τις προβλέψεις του νευρωνικού δικτύου. Η Scikit-learn είναι ίσως από τις πιο χρήσιμες βιβλιοθήκες για μηχανική μάθηση στην Python, καθώς περιέχει πολλά αποτελεσματικά εργαλεία για μηχανική μάθηση και στατιστική μοντελοποίηση, όπως ταξινόμηση και παλινδρόμηση(regression).

Κεφάλαιο 6

Στο συγκεκριμένο κεφάλαιο πρόκειται να παρουσιαστούν τα αποτελέσματα κάθε υλοποίησης όσων αφορά στην πρόβλεψη της θέσης του κέντρου των ρωγμών στην ελαστική δομή που θεωρείται για την εν λόγω εργασία. Αντίθετα με τα προηγούμενα κεφάλαια, το συγκεκριμένο πρόκειται να διαρθρωθεί βάση της κατηγορίας στην οποία ανήκει κάθε διάγραμμα και όχι βάση της κάθε υλοποίησης.

6.1 Διαγράμματα Loss Function

6.1.1 Υλοποίηση Matlab



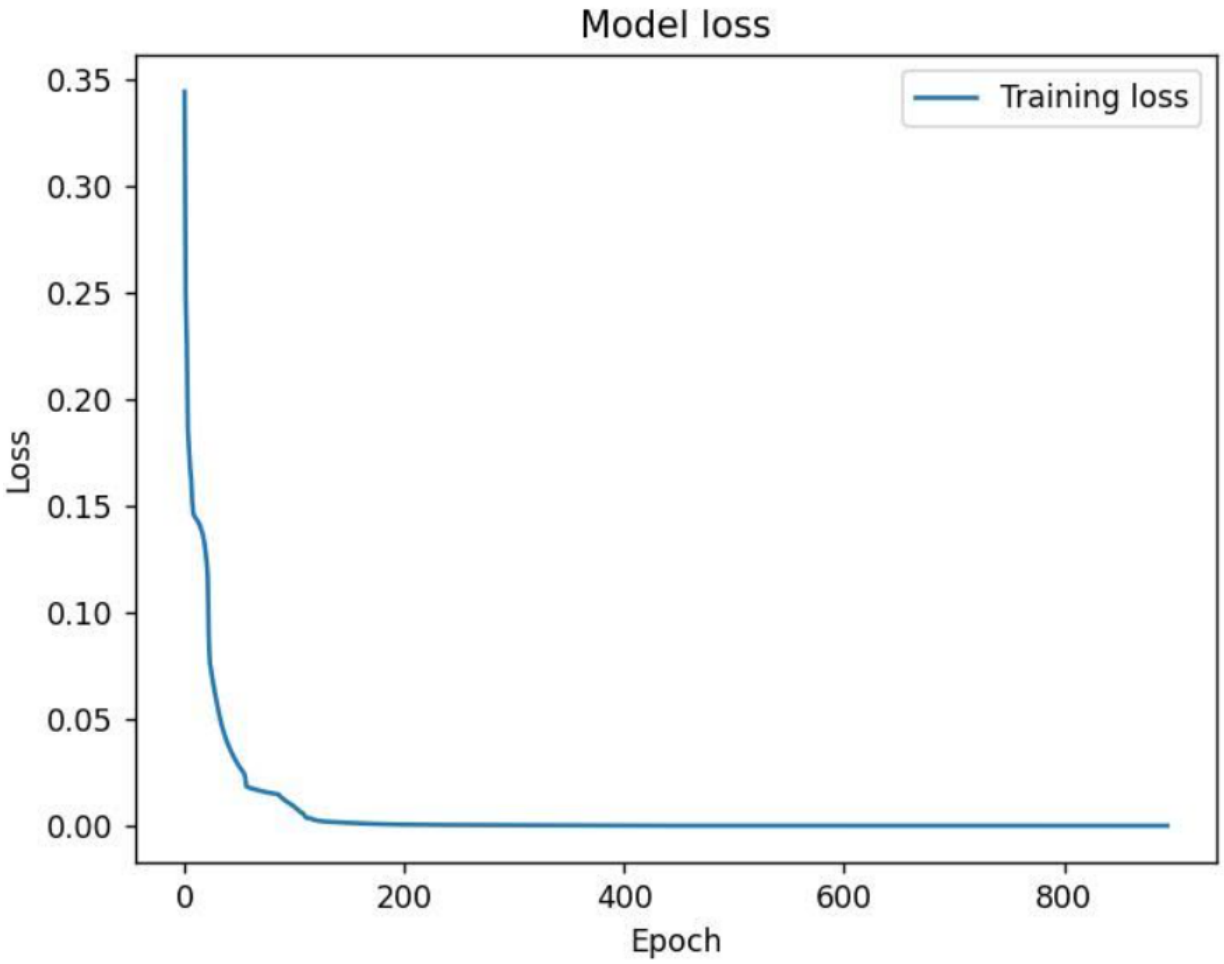
Εικόνα 22 – Matlab Διάγραμμα Loss Function

Loss: 9.9989*e^{-5}

Επαναλήψεις: 608

Χρόνος: 17’’

6.1.2 Υλοποίηση Python



Εικόνα 23 – Python Διάγραμμα Loss Function

Loss: 3.0457*e^{-5}

Επαναλήψεις: 895

Χρόνος: 30’’

6.1.3 Παρατηρήσεις

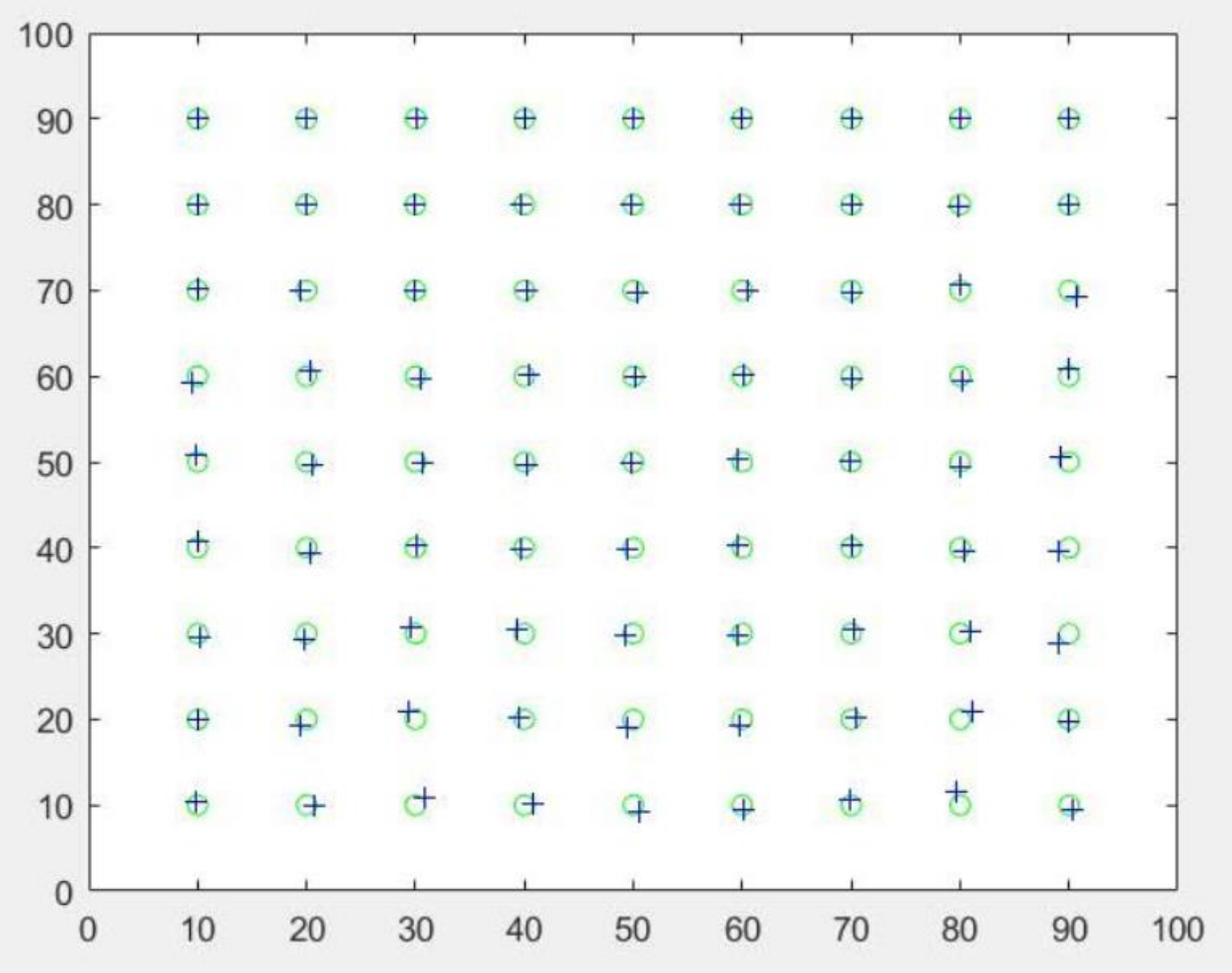
Είναι προφανές πως κατά την υλοποίηση σε Matlab, τόσο ο χρόνος της εκπαίδευσης του μοντέλου του νευρωνικού δικτύου όσο και οι επαναλήψεις έχουν μικρότερες τιμές σε σχέση με την υλοποίηση σε Python. Ωστόσο το γεγονός πως το σφάλμα ελαχιστοποιείται αρκετά περισσότερο κατά την υλοποίηση σε Python, μας επιτρέπει να προσπεράσουμε την προαναφερθείσα διαφορά σε ότι αφορά στις επαναλήψεις. Άλλωστε πρόκειται για ένα χαρακτηριστικό το οποίο επιβαρύνει χρονικά τη διαδικασία της εκπαίδευσης και στην περίπτωσή μας έχει να κάνει μόνο με μερικά δευτερόλεπτα. Στον αντίποδα, οι προβλέψεις του

νευρωνικού δικτύου βελτιώνονται αισθητά, όπως θα παρουσιαστεί στη συνέχεια, ιδίως αναφορικά τα άγνωστα στο νευρωνικό δίκτυο, δεδομένα ελέγχου .

6.2 Προβλέψεις αναφορικά με τα Training Data

6.2.1 Υλοποίηση Matlab

6.2.1.1 Θέσεις Οπών

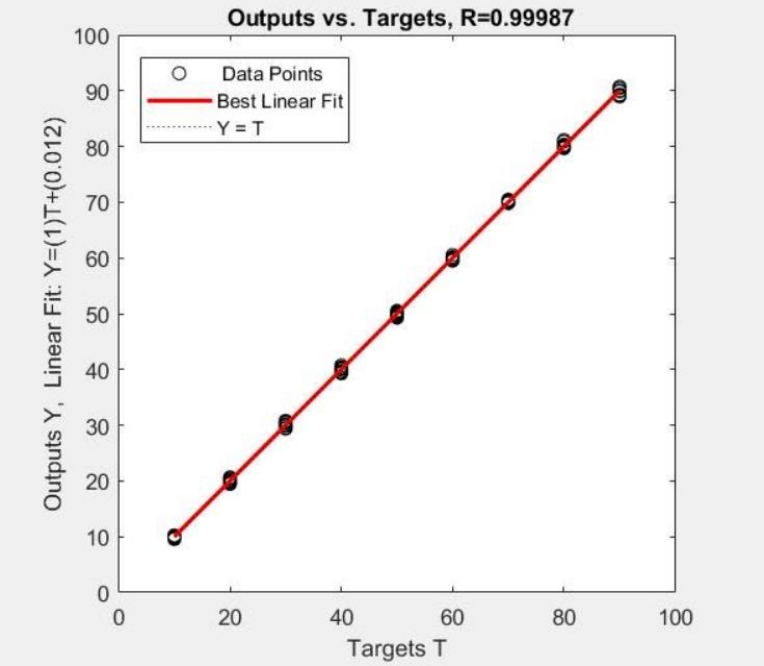


Εικόνα 24 – Matlab Θέσεις Οπών (training data)

6.2.1.2 Linear Regression Συντεταγμένων Κέντρου Ρωγμών

Τα παρακάτω διαγράμματα παρατίθενται προκειμένου να έχουμε μια πιο ολοκληρωμένη εικόνα σχετικά με εγκυρότητα των προβλέψεων του νευρωνικού δικτύου για κάθε έξοδο ξεχωριστά.

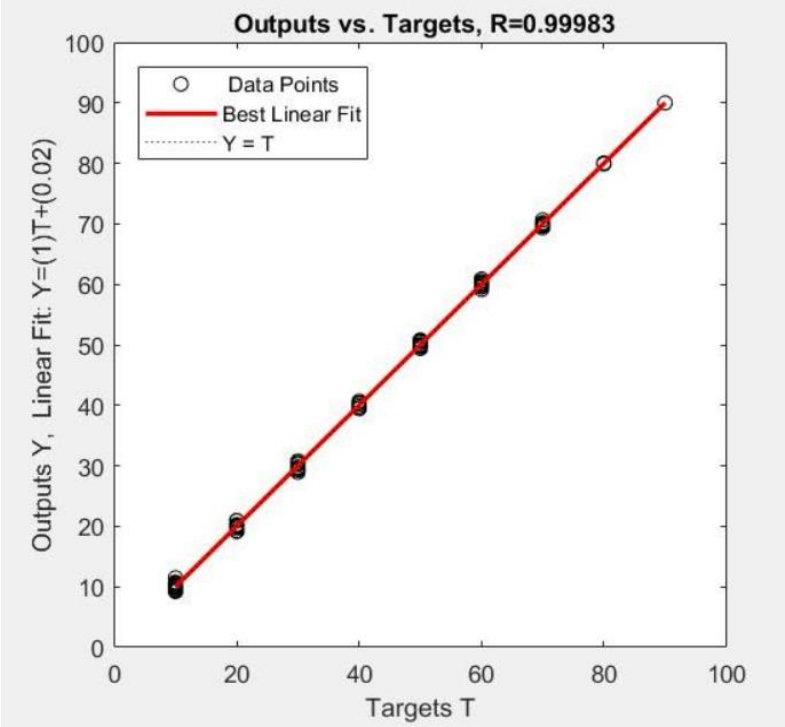
Συντεταγμένη X



Εικόνα 25 – Matlab X (train) Linear Regression

Y = 1*X + 0.012 R = 0.99987

Συντεταγμένη Y

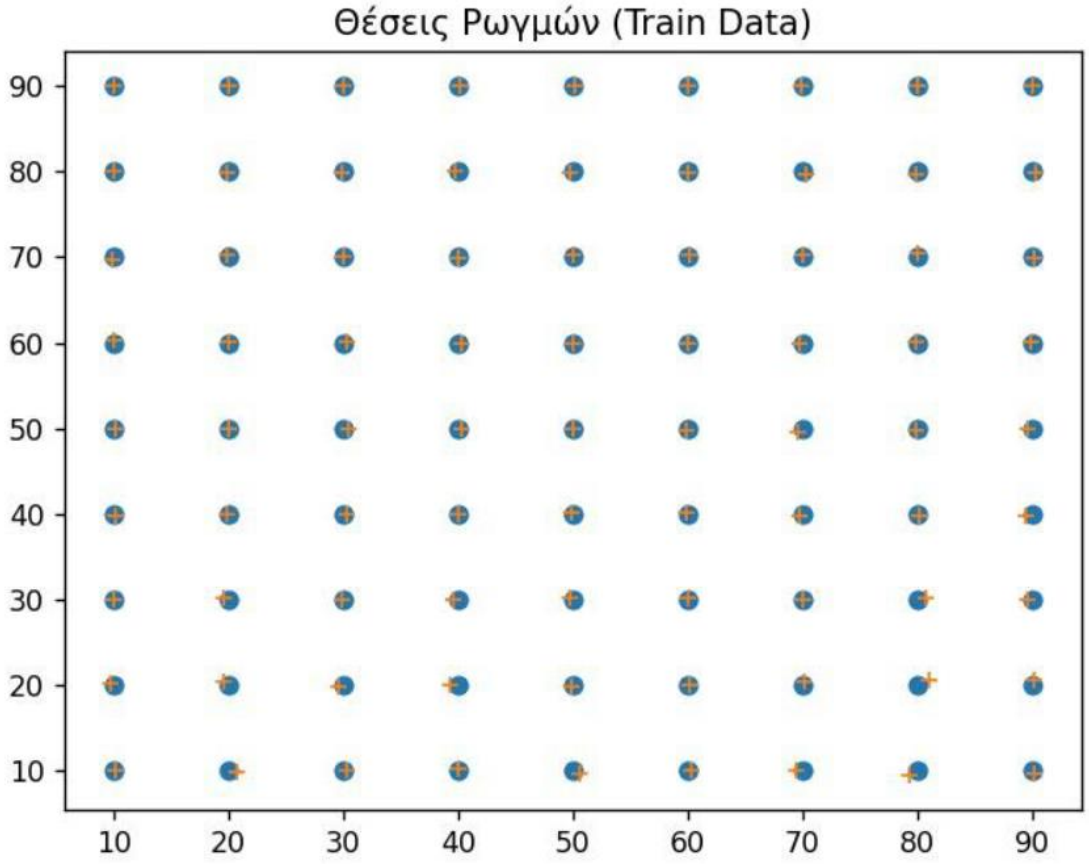


Εικόνα 26 – Matlab Y (train) Linear Regression

Y = 1 * X + 0.02 R = 0.99983

6.2.2 Υλοποίηση Python

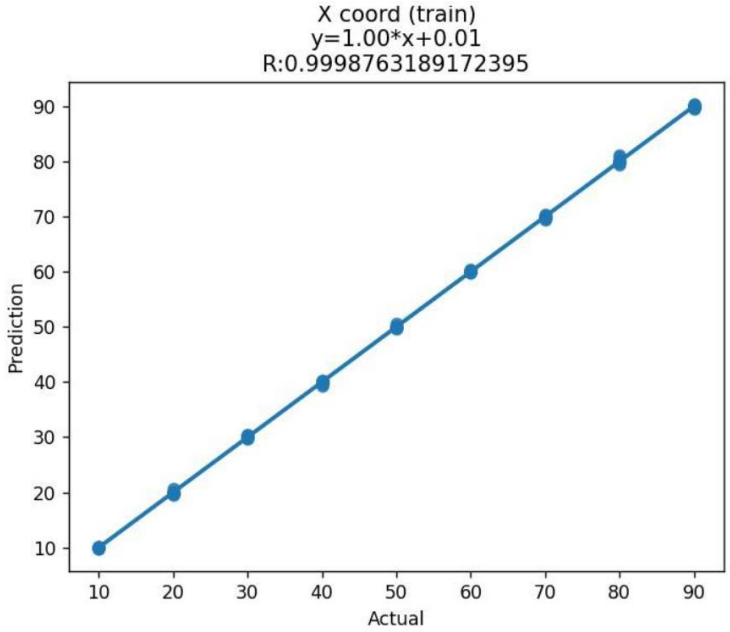
6.2.2.1 Θέσεις οπών



Εικόνα 27 – Python Θέσεις Οπών(training data)

6.2.2.2 Linear Regression Συντεταγμένων Κέντρου Ρωγμών

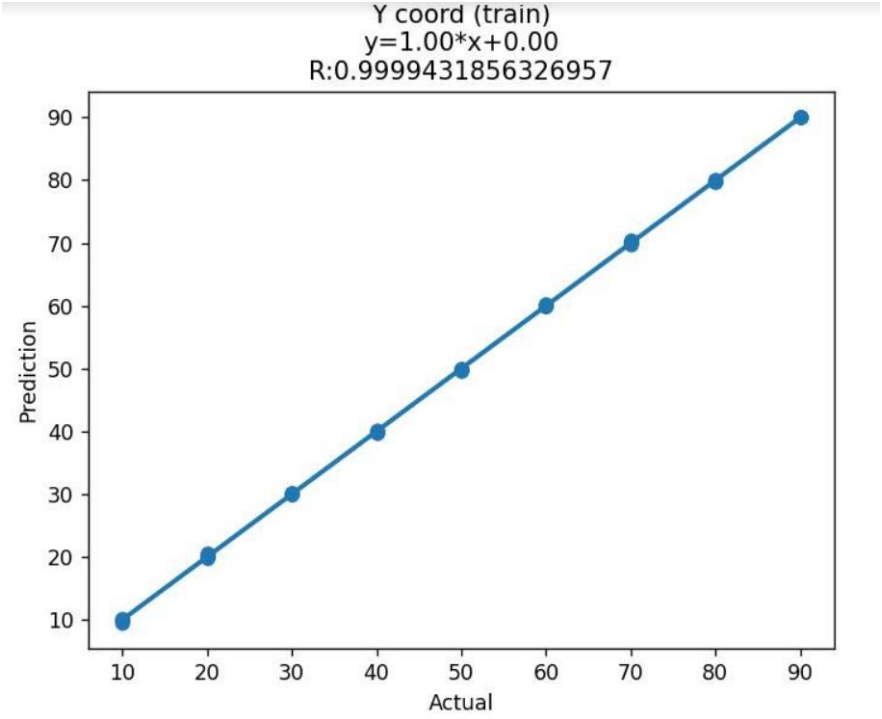
Συντεταγμένη X



Εικόνα 28 – Python X (train) Linear Regression

Y = 1*X + 0.01 R = 0.99987

Συντεταγμένη Y



Εικόνα 29 – Python Y (train) Linear Regression

Y = 1*X R = 0.99994

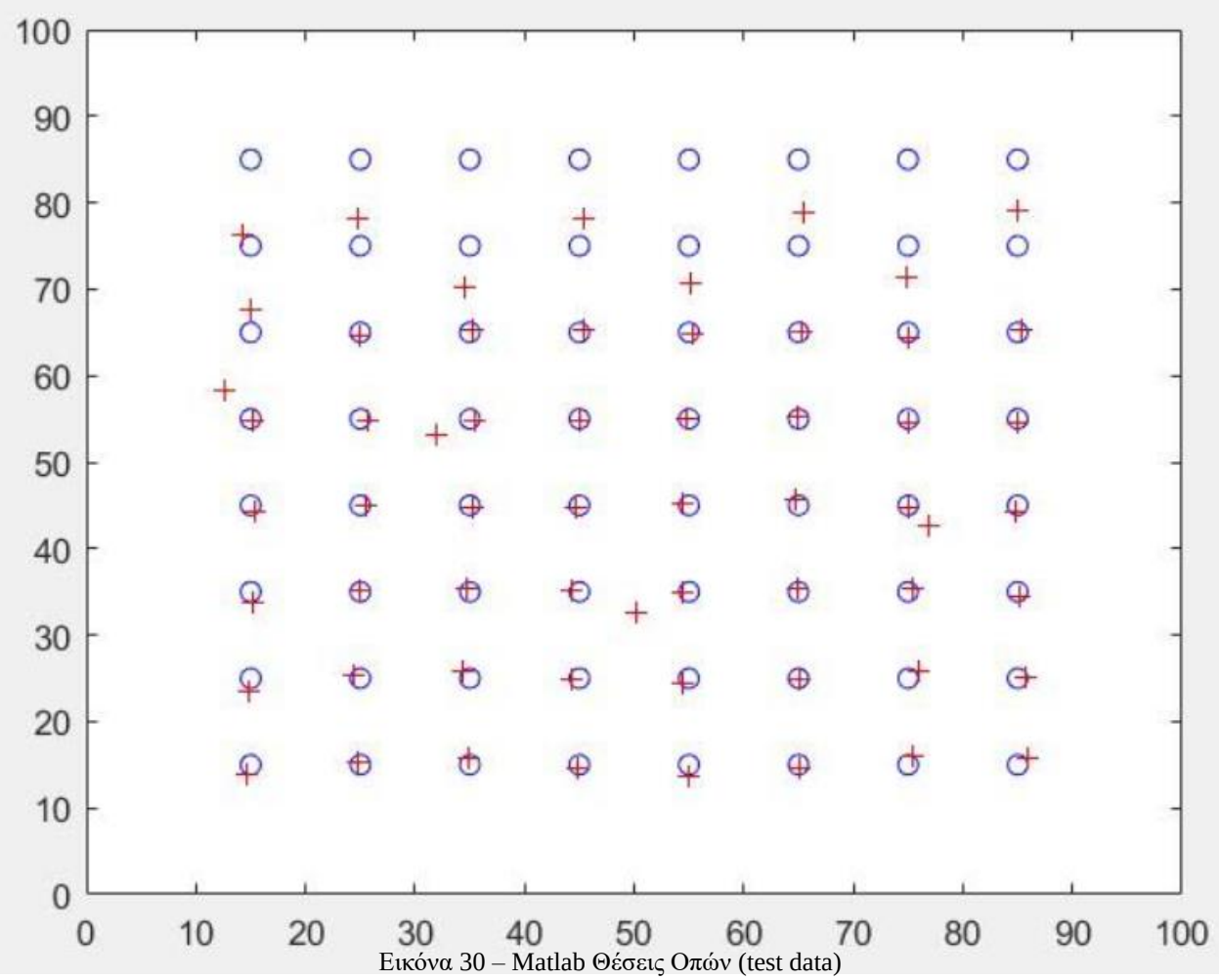
6.2.3 Παρατηρήσεις

Όπως είναι προφανές οι διαφορές σε ότι αφορά στις προβλέψεις που βασίζονται στα Training Data είναι σχεδόν ανεπαίσθητες. Αναφορικά με τη συντεταγμένη X τα αποτελέσματα είναι όμοια. Ωστόσο αναφορικά με τη συντεταγμένη Y κατά την υλοποίηση σε Python υπάρχει καλύτερη συσχέτιση μεταξύ των τιμών που έχουν προβλεφθεί και των τιμών στόχων.

6.3 Προβλέψεις αναφορικά με τα Test Data

6.3.1 Υλοποίηση Matlab

6.3.1.1 Θέσεις Οπών

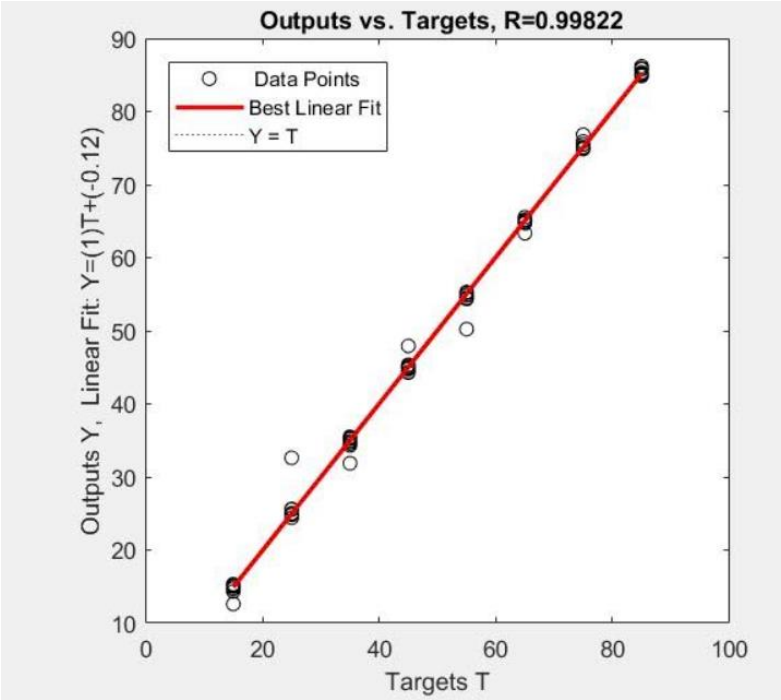


Εικόνα 30 – Matlab Θέσεις Οπών (test data)

6.3.1.2 Linear Regression Συντεταγμένων Κέντρου Ρωγμών

Τα παρακάτω διαγράμματα παρατίθενται προκειμένου να έχουμε μια πιο ολοκληρωμένη εικόνα σχετικά με εγκυρότητα των προβλέψεων του νευρωνικού δικτύου για κάθε έξοδο ξεχωριστά.

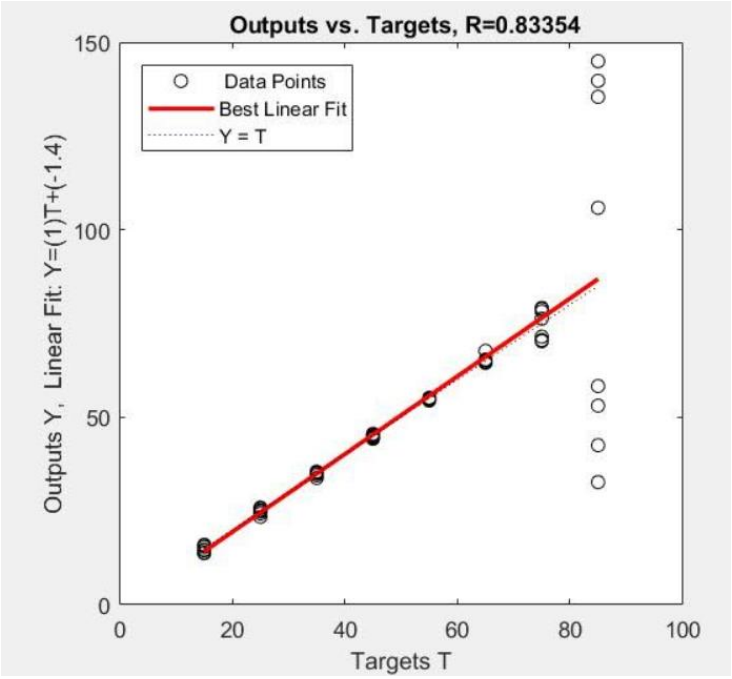
Συντεταγμένη X



Εικόνα 31 – Matlab X (test) Linear Regression

Y = 1*X – 0.12 R = 0.99822

Συντεταγμένη Y

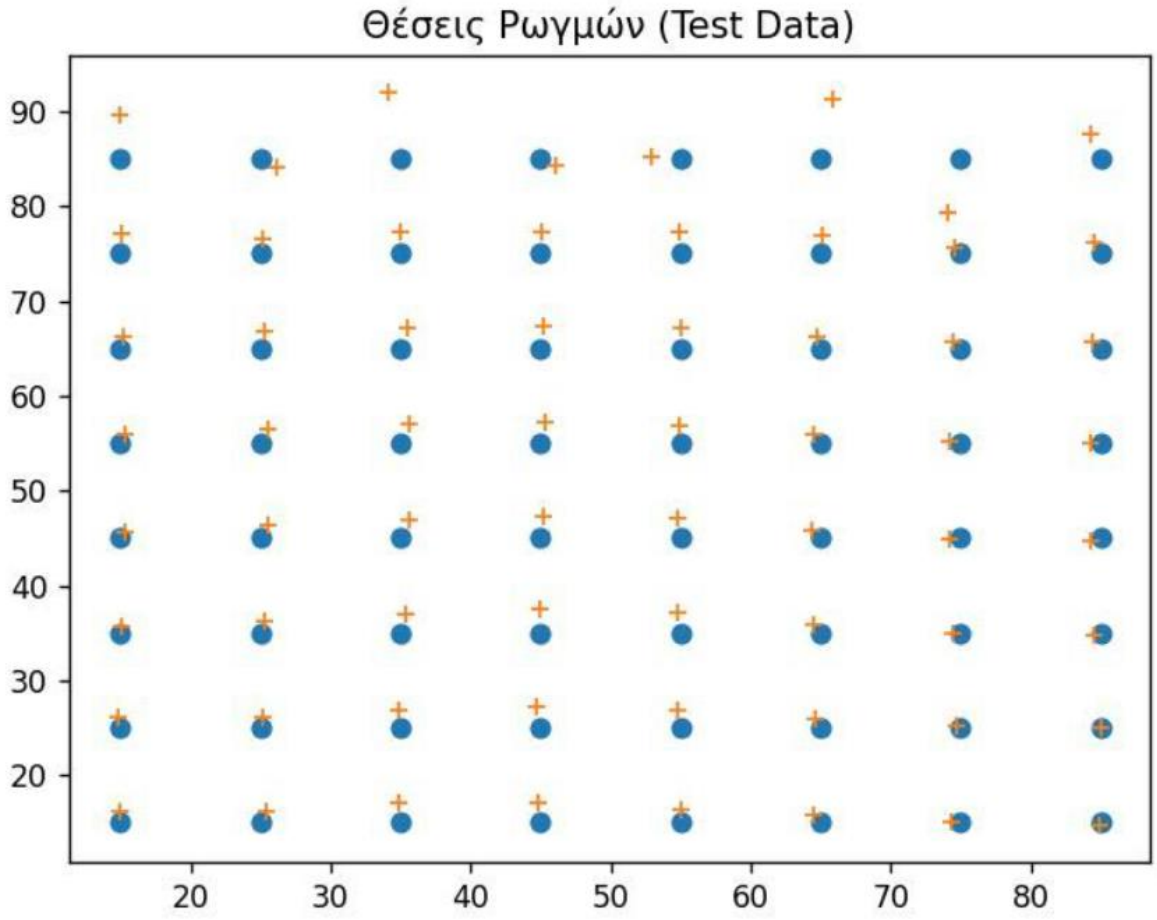


Εικόνα 32 – Matlab Y (test) Linear Regression

Y = 1*X – 1.4 R = 0.83354

6.3.2 Υλοποίηση Python

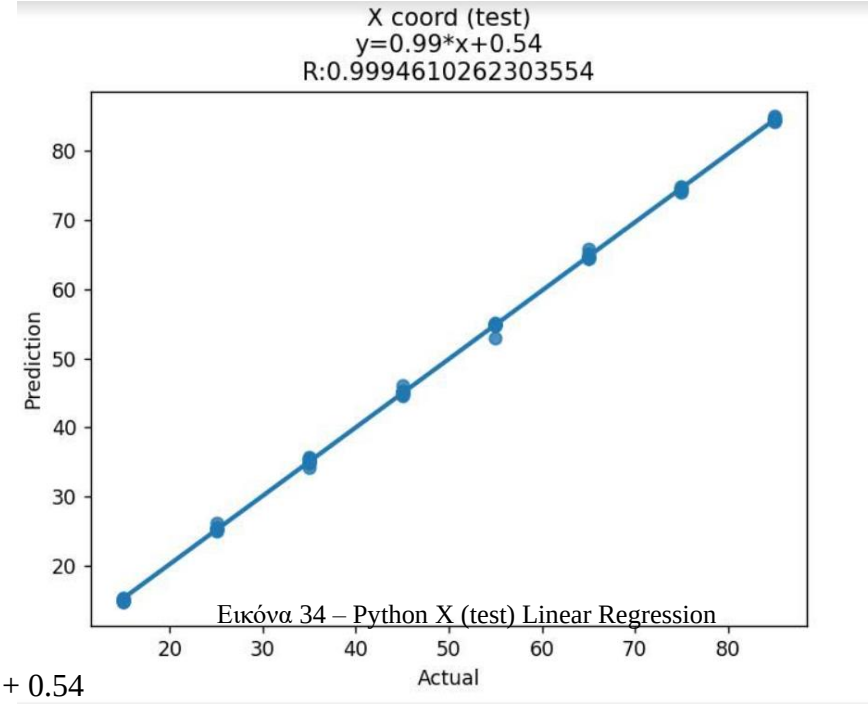
6.3.2.1 Θέσεις οπών



Εικόνα 33 – Python Θέσεις Οπών (test data)

6.3.2.2 Linear Regression Συντεταγμένων Κέντρου Ρωγμών

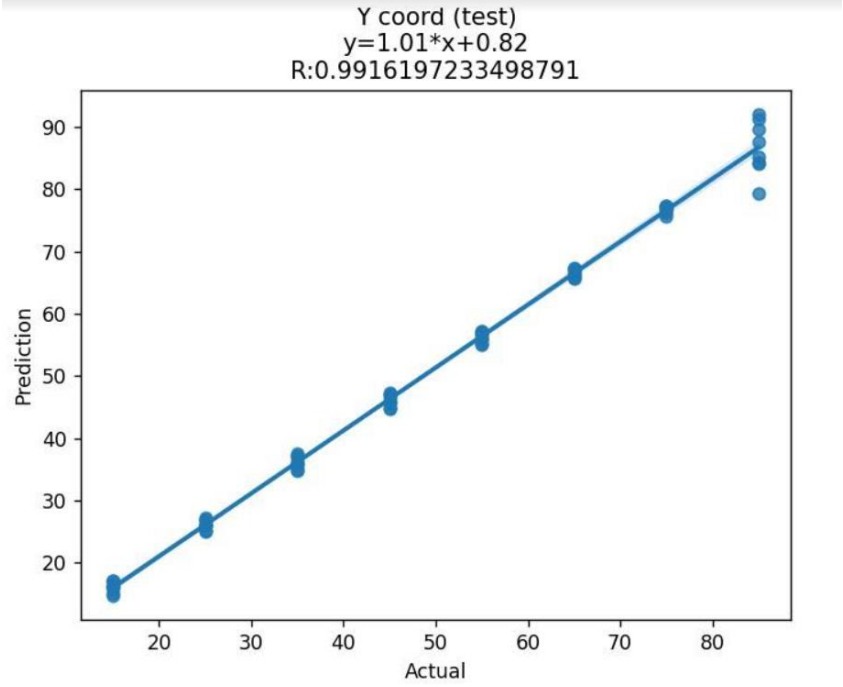
Συντεταγμένη X



Εικόνα 34 – Python X (test) Linear Regression

Y = 0.99*X + 0.54 R = 0.99946

Συντεταγμένη Y



Εικόνα 35 – Python Y (test) Linear Regression

Y = 1.01*X + 0.82 R = 0.99161

6.3.3 Παρατηρήσεις

Όπως φαίνεται ξεκάθαρα, οι διαφορές ανάμεσα στις τιμές Regression για τις συντεταγμένες X, Y είναι μικρές αλλά σημαντικές, με μεγαλύτερη διαφορά να παρατηρείται για τη συντεταγμένη Y. Το αξιοσημείωτο του γεγονότος αυτού φαίνεται στην Εικόνα 33 όπου παρατηρούμε πόσο κοντά βρίσκονται οι προβλέψεις του νευρωνικού δικτύου σε σχέση με τις τιμές-στόχους.

Κεφάλαιο 7

7.1 Επίλογος

Στην παρούσα εργασία διερευνήθηκε η αποτελεσματικότητα της χρήσης νευρωνικών δικτύων οπίσθιας διάδοσης για ένα αντίστροφο πρόβλημα αναγνώρισης ρωγμών από μετρήσεις μετακινήσεων στην εξωτερική πλευρά μιας στατικά φορτισμένης πλάκας. Επιπλέον, παρατέθηκε εγχειρίδιο, όπου αναλύεται η διαφοροποίηση του νέου κώδικα σε Python σε σχέση με τον προηγούμενο κώδικα σε Matlab. Αυτή η διαφοροποίηση έχει να κάνει τόσο με τις αλλαγές στα εργαλεία που χρησιμοποιήθηκαν όσο και σε στοιχεία που αφορούν το ίδιο το μοντέλο του νευρωνικού δικτύου. Τέλος, τα δεδομένα για την εκπαίδευση και τον έλεγχο, έχουν παραχθεί από μια υπολογιστική τεχνική(Boundary Element Method).

7.2 Συμπεράσματα

Όπως φαίνεται και από τα αποτελέσματα που παρουσιάστηκαν στο προηγούμενο κεφάλαιο η μέθοδος που προτείνεται για την επίλυση αντίστροφων προβλημάτων τέτοιου είδους, κρίνεται απολύτως επαρκής. Επιπροσθέτως, είναι ορθό να σημειωθεί πως μολονότι η Matlab αποτελεί μια ισχυρή πλατφόρμα, εν τούτοις η πρόοδος που συντελείται σε ότι αφορά άλλες γλώσσες προγραμματισμού και σε πακέτα που αυτές περιλαμβάνουν, δε γίνεται να αφήσει την έρευνα ανεπηρέαστη. Κατ’ αυτόν τον τρόπο διευκολύνεται τόσο η διαδικασία μοντελοποίησης ενός νευρωνικού δικτύου όσο και ο τρόπος εξαγωγής των εκάστοτε αποτελεσμάτων

7.3 Προτάσεις για Μελλοντική Έρευνα

Η εφαρμογή άλλων ειδών μοντέλων νευρωνικών δικτύων και σχημάτων μείωσης δεδομένων, που θα επέτρεπαν τη μελέτη πιο περίπλοκων επιπτώσεων, αφήνεται ανοιχτή για περαιτέρω έρευνες. Επιπλέον, η χρήση άλλων κατάλληλων δεδομένων του προβλήματος (π.χ. υστέρηση φάσης), τα οποία μπορούν να παραχθούν με ανάλογες υπολογιστικές τεχνικές, μπορεί να έχει ενδιαφέρον για την επίλυση σχετικών αντίστροφων προβλημάτων στη μηχανική.

Παράρτημα

Παρατίθεται ο κώδικας της επίλυσης του προβλήματος σε γλώσσα προγραμματισμού Python

```
import time
import levenberg_marquardt as lm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pylab import rcParams
from pprint import pprint
import warnings
import tensorflow as tf
from tensorflow import keras
import seaborn as sns
from sklearn.metrics import r2_score
from ann_visualizer.visualize import ann_viz
from graphviz import Source

#not normalized test targets
y_test_after = pd.read_csv('//not_normalized_test_targets')
y_test_after = np.asarray(y_test_after,np.float32)
y_test_after = y_test_after.transpose()

#not normalized train targets
myCheck = pd.read_csv('//not_normalized_train_targets')
myCheck = np.asarray(myCheck,np.float32)
myCheck = myCheck.transpose()

#training inputs(normalized)
X_train = pd.read_csv('//train_inputs') # 81,81
X_train = np.asarray(X_train,np.float32)
X_train = X_train.transpose()

#training targets(normalized)
y_train = pd.read_csv('//train_targets') #0,81
y_train = np.asarray(y_train,np.float32)
y_train = y_train.transpose()

#test inputs(normalized)
X_test = pd.read_csv('//test_inputs)
X_test = np.asarray(X_test,np.float32)
X_test = X_test.transpose()

#test targets(normalized)
y_test = pd.read_csv('//test_targets')
y_test = np.asarray(y_test,np.float32)
y_test = y_test.transpose()

#####################################################
##########custom_functions############################
#####################################################
```

```
def lineEquation(regPlotObj):

    #get x-y data
    theXs = regPlotObj.get_lines()[0].get_xdata()
    theYs = regPlotObj.get_lines()[0].get_ydata()

    # Define some random known points
    x = [theXs[3],theXs[13]]
    y = [theYs[3],theYs[13]]

    # Calculate and round the coefficients.
    coefficients = np.polyfit(x, y, 1)
    myA = "%.2f" % round(coefficients[0], 2)
    myB = "%.2f" % round(coefficients[1], 2)

    #console print
    print('a =', coefficients[0])
    print('b =', coefficients[1])

    #check in order to display equation promptly
    if(float(myB)>=0):
        theText = 'y='+str(myA) + '*x+'+str(myB)
    else:
        theText = 'y=' + str(myB) + '*x' + str(myB)

    return theText

def theRofY(pureData,predictedData):
    #calculate R for Xs
    R_square = r2_score(pureData[0:, 1], predictedData[0:, 1])
    print('Coefficient of Determination', R_square)
    theText = "R:"+str(R_square)
    return theText

def theRofX(pureData,predictedData):
    #calculate R for Ys
    R_square = r2_score(pureData[0:, 0], predictedData[0:, 0])
    print('Coefficient of Determination', R_square)
    theText = "R:"+str(R_square)
    return theText

#################################################################

#extra steps for data proccessing
train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
train_dataset = train_dataset.shuffle(82,)
train_dataset = train_dataset.batch(512).cache()
train_dataset = train_dataset.prefetch(tf.data.experimental.AUTOTUNE)

#model definition
model = keras.Sequential()

model.add(keras.layers.Dense(82,activation='tanh', input_shape=(82,)))
model.add(keras.layers.Dense(27, activation='tanh'))
model.add(keras.layers.Dense(9, activation='tanh'))
model.add(keras.layers.Dense(2))
```

```
#defining haltcallback
class haltCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_loss') <= 0.00001 ):#10e-06
            print("\n\n\nReached 10e-06 loss value so cancelling
training!\n\n\n")
            self.model.stop_training = True


trainingStopCallback = haltCallback()


model.summary()
ann_viz(model, title="My artificial neural network")

#define model wrapper in order to imply LM algorithm
model_wrapper = lm.ModelWrapper(
    tf.keras.models.clone_model(model))

#model compilation usin LM algorithm
model_wrapper.compile(
    optimizer=tf.keras.optimizers.SGD(learning_rate=1),
    loss=lm.MeanSquaredError())

print("\n_____________________________________________________________________")
print("Train using Levenberg-Marquardt")
#start counter
t2_start = time.perf_counter()

#training the ann model
hist = model_wrapper.fit(train_dataset,
epochs=5000,callbacks=[trainingStopCallback])

#stop counter
t2_stop = time.perf_counter()
print("Elapsed time: ", t2_stop - t2_start)
print("\n_____________________________________________________________________")
print("Plot results")


#ANN predictions for test inputs
y_pred=model_wrapper.predict(X_test)

#De-Normalize predictions for test inputs
y_pred2 = (y_pred+0.9)/1.8*(np.max(y_test_after.max(axis=1))-
np.min(y_test_after.min(axis=1)))+np.min(y_test_after.min(axis=1))

#Crack Position on Plate (Test)
plt.plot(y_test_after[0:,0],y_test_after[0:,1],'o')
plt.plot(y_pred2[0:,0],y_pred2[0:,1],'+')
plt.title('Θέσεις Ρωγμών (Test Data)')
plt.show()

#ANN predictions for train data
y_before = model_wrapper.predict(X_train)
```

```
#De-normalize predictions for train data
y_before2 = (y_before+0.9)/1.8*(np.max(myCheck.max(axis=1))-
np.min(myCheck.min(axis=1)))+np.min(myCheck.min(axis=1))

#Crack Position on Plate (Train)
plt.plot(myCheck[0:,0],myCheck[0:,1],'o')
plt.plot(y_before2[0:,0],y_before2[0:,1],'+')
plt.title('Θέσεις Ρωγμών (Train Data)')
plt.show()

#Loss Diagramm
plt.plot(hist.history['loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training loss', 'Validation loss'], loc='upper right')
plt.show()

#create Pandas Dataframes
#test data
df_compareXs = pd.DataFrame({'Actual' : y_test_after[0:,0], 'Prediction':
y_pred2[0:,0]})
df_compareYs = pd.DataFrame({'Actual' : y_test_after[0:,1], 'Prediction':
y_pred2[0:,1]})

#train data
train_df_compareXs = pd.DataFrame({'Actual' : myCheck[0:,0], 'Prediction':
y_before2[0:,0]})
train_df_compareYs = pd.DataFrame({'Actual' : myCheck[0:,1], 'Prediction':
y_before2[0:,1]})

#Linear Regression Diagrams

#X test
c=sns.regplot(x = "Actual", y = "Prediction", data=df_compareXs)
print('(test data)->X->')
theEq=lineEquation(c)
theR=theRofX(y_test_after,y_pred2)
plt.title("X coord (test)\n"+theEq+"\n"+theR)
plt.show()

#Y test
d=sns.regplot(x = "Actual", y = "Prediction", data=df_compareYs)
print('(test data)->Y->')
theEq=lineEquation(d)
theR=theRofY(y_test_after,y_pred2)
plt.title("Y coord (test)\n"+theEq+"\n"+theR)
plt.show()

#Xtrain
e=sns.regplot(x = "Actual", y = "Prediction", data=train_df_compareXs)
print('(train data)->X->')
theEq=lineEquation(e)
theR=theRofX(myCheck,y_before2)
plt.title("X coord (train)\n"+theEq+"\n"+theR)
plt.show()
```

```
#Ytrain
g= sns.regplot(x = "Actual", y = "Prediction", data=train_df_compareYs)
print('(test data)->Y->')
theEq=lineEquation(g)
theR=theRofY(myCheck,y_before2)
plt.title("Y coord (train)\n"+theEq+"\n"+theR)
plt.show()
```

Βιβλιογραφία

[1] G.E. Stavroulakis, H. Antes ,Neural crack identification in steady state elastodynamics , Computer Methods in Applied Mechanics and Engineering. Volume 165, Issues 1–4, 2 November 1998.

[2] https://en.wikipedia.org/wiki/Boundary_element_method , (Προσπελάστηκε 7/7/2022)

[3] https://el.wikipedia.org/wiki/Τεχνητή_νοημοσύνη ,(Προσπελάστηκε 7/7/2022)

[4] Zachos, E., Pagourtzis, A., & Souliou, T. (2015).Τεχνητή Νοημοσύνη [Chapter]. In Zachos, E., Pagourtzis, A., & Souliou, T. 2015. Θεμελίωση επιστήμης υπολογιστών[Undergraduate textbook]. Kallipos, Open Academic Editions., σελίδα 1

[5] Georgouli, A. (2015). Μηχανική Μάθηση [Chapter]. In Georgouli, A. 2015. Τεχνητή νοημοσύνη [Undergraduate textbook]. Kallipos, Open Academic Editions. chapter 1 ,σελίδες 1-3

[6] Mitchell, T.M. (1997). Machine Learning. Maidenhead, H.B.:McGraw-Hill International Editions.

[7] https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network ,(Προσπελάστηκε 7/7/2022)

[8] https://www.analyticssteps.com/blogs/deep-learning-overview-practical-examples-popular-algorithms (Προσπελάστηκε 7/7/2022)

[9] Βαλουξής Δημήτρης, Δράκος Γιώργος, ΜΕΛΕΤΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ DEEP LEARNING ΤΕΧΝΙΚΩΝ ΣΤΟΝ ΤΟΜΕΑ ΤΗΣ ΑΥΤΟΚΙΝΗΣΗΣ , Διπλωματική εργασία, Πανεπιστήμιο Θεσσαλίας. Πολυτεχνική Σχολή. Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, 2018, σελίδες 19-21, 26-29, 36

[10] https://en.wikipedia.org/wiki/Artificial_neural_network (Προσπελάστηκε 7/7/2022)

[11] https://en.wikipedia.org/wiki/Supervised_learning (Προσπελάστηκε 7/7/2022)

[12] Maad M. Mijwel , Artificial Neural NetworksAdvantages and Disadvantages , Computer science, college of science,University of BaghdadBaghdad, Iraq, January 2018 , σελίδες 1-2

[13] https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a , (Προσπελάστηκε 7/7/2022)

[14] https://machine-learning.paperspace.com/wiki/weights-and-biases (Προσπελάστηκε 7/7/2022)

[15] P.SIBI, S.ALLWYN JONES, P.SIDDARTH, ANALYSIS OF DIFFERENT ACTIVATION FUNCTIONS USING BACK PROPAGATION NEURAL NETWORKS, SASTRA University,

Kumbakonam, India, Journal of Theoretical and Applied Information Technology, January 2013, σελίδες 2-4

[16] https://keras.io/api/layers/activations/ (Προσπελάστηκε 7/7/2022)

[17] Peter F. Christoffersen, Kris Jacobs , The Importance of the Loss Function in Option Valuation in EFA 2003 Glasgow Meetings Presentation Papers, August 2002 , σελίδα 4

[18] Ilonen, J., Kamarainen, JK. & Lampinen, J. Differential Evolution Training Algorithm for Feed-Forward Neural Networks.Neural Processing Letters, σελίδες: 93–105 (2003)

[19] Lei Jimmy, Ba Brendan Frey, Adaptive dropout for training deep neural networks, Department of Electrical and Computer Engineering University of Toronto , σελίδες 1-2

[20] https://www.applause.com/blog/training-data-validation-data-vs-test-data (Προσπελάστηκε 7/7/2022)

[21] Sudeep Tanwar , Qasim Bhatia, Pruthvi Patel, Dr. Aparna Kumari , Pradeep Kumar Singh, Wei-Chiang Hong, Machine Learning Adoption in Blockchain-Based Smart Applications: The Challenges, and a Way Forward, IEEE Access 2020, March 2020, σελίδα 9

[22] https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/ (Προσπελάστηκε 7/7/2022)

[23] Eibe Frank , Mark A. Hall , Christopher Pal, Data Mining: Practical Machine Learning Tools and Techniques (Morgan Kaufmann Series in Data Management Systems, 2016, σελίδα 61

[24] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html Προσπελάστηκε 7/7/2022)

[25] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (Προσπελάστηκε 7/7/2022)

[26] https://levity.ai/blog/neural-networks-cnn-ann-rnn (Προσπελάστηκε 7/7/2022)

[27] https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/ (Προσπελάστηκε 7/7/2022)

[28] https://www.tensorflow.org/api_docs/python/tf/keras/Model (Προσπελάστηκε 7/7/2022)

[29] https://www.mathworks.com/discovery/what-is-matlab.html (Προσπελάστηκε 7/7/2022)

[30] https://www.python.org/doc/essays/blurb/ (Προσπελάστηκε 7/7/2022)

[31] Taylor Colliau, Grace Rogers, Z. Hughes, Ceyhun Ozgur, MatLab vs. Python vs. R, Valparaiso University, 2017 σελίδες 3, 5-9

[32] https://www.tensorflow.org/resources/learn-ml (Προσπελάστηκε 7/7/2022)

[33] https://keras.io/about/ (Προσπελάστηκε 7/7/2022)

[34] https://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_Mfile#:~:text=An%20m%2Dfile%2C%20or%20script,m (Προσπελάστηκε 7/7/2022)

[35] https://en.wikipedia.org/wiki/Comma-separated_values (Προσπελάστηκε 7/7/2022)

[36] Peshawa Jamal Muhammad Ali*, and Rezhna Hassan Faraj, Data Normalization and Standardization: A Technical Report, The Machine Learning Lab. at Koya University Koya, Erbil, Iraq,2014, σελίδες 1-3

[37] https://www.tensorflow.org/api_docs/python/tf/data/Dataset (Προσπελάστηκε 7/7/2022)

[38]http://matlab.izmiran.ru/help/toolbox/nnet/newff.html#:~:text=newff%20(Neural%20Network%20Toolbox (Προσπελάστηκε 7/7/2022)

[39] https://www.mathworks.com/help/deeplearning/ref/tansig.html (Προσπελάστηκε 7/7/2022)

[40] https://www.mathworks.com/help/deeplearning/ref/purelin.html (Προσπελάστηκε 7/7/2022)

[41] https://machinelearningmastery.com/difference-between-backpropagation-and-stochastic-gradient-descent/ (Προσπελάστηκε 7/7/2022)

[42] Ian Goodfellow, Yoshua Bengio , Deep Learning (Adaptive Computation and Machine Learning series),2016, σελίδες 195, 204-205, 222

[43] John Taylor , Wenyi Wang , Biswajit Bala, Tomasz Bednarz, Optimizing the optimizer for data driven deep neural networks and physics informed neural networks, 2022, σελίδες 4-5,10

[44] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathematics 2 (2), 164–168

[45] Marquardt, D. W. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. Journal of the Society for Industrial and Applied Mathematics 11 (2), 431–441

[46] https://en.wikipedia.org/wiki/Hessian_matrix (Προσπελάστηκε 7/7/2022)

[47] Sam Roweis, Levenberg-Marquardt Optimization, https://cs.nyu.edu/~roweis/notes/lm.pdf (Προσπελάστηκε 7/7/2022)

[48] https://au.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-trainingfunction (Προσπελάστηκε 7/7/2022)

[49] Wenyi Wang, John Taylor ,Biswajit Bala, Exploiting the Power of Levenberg-Marquardt Optimizer with Anomaly Detection in Time Series, Australian National University, School of Computing, 2021, σελίδες 1-2

[50] Dejan BrkiT, Carko SojbašiT, Intelligent Flow Friction Estimation in Hindawi Publishing Corporation Computational Intelligence and Neuroscience ,2016, σελίδες 4-5

[51] https://machinelearningmastery.com/keras-functional-api-deep-learning (Προσπελάστηκε 7/7/2022)

[52] https://keras.io/api/layers/core_layers/dense/ (Προσπελάστηκε 7/7/2022)

[53] https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (Προσπελάστηκε 7/7/2022)

[54] KOLMOGOROV, A.N., 1957, On the representational of continuous functions of many variables by superpositions of continuous functional of one variable and addition. Doklady Akademii Nauk USSR, 114, pp. 953–956.

[55] HECHT-NIELSEN, R., 1987, Kolmogorov's mapping neural network existence theorem. In IEEE First Annual International Conference on Neural Networks, 3, pp. 11–13

[56] KURKOVA, V., 1992, Kolmogorov's theorem and multilayer neural networks. Neural Networks, 5, pp. 501–506

[57] HUANG, G.-B. and BABRI, H., 1997, General approximation theorem on feedforward networks. In International Conference on Information, Communications and Signal Processing, ICICS '97, Singapore, 9–12 September, pp. 698– 702

[58] (HUANG, G.-B., 2003, Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Transactions on Neural Networks, 14, pp. 274–281

[59] D. STATHAKIS, How many hidden layers and nodes?, International Journal of Remote Sensing, April 2009, σελίδες 2-3

[60] https://keras.io/api/layers/core_layers/ (Προσπελάστηκε 7/7/2022)

[61] https://github.com/fabiodimarco/tf-levenberg-marquardt (Προσπελάστηκε 7/7/2022)

[62] https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanSquaredError (Προσπελάστηκε 7/7/2022)

[63] https://keras.io/api/losses/ (Προσπελάστηκε 7/7/2022)

[64] https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-

optimizers/ (Προσπελάστηκε 7/7/2022)

[65] https://keras.io/api/optimizers/sgd/ (Προσπελάστηκε 7/7/2022)

[66] https://en.wikipedia.org/wiki/Learning_rate (Προσπελάστηκε 7/7/2022)

[67] https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/ (Προσπελάστηκε 7/7/2022)

[68] http://matlab.izmiran.ru/help/toolbox/nnet/train.html (Προσπελάστηκε 7/7/2022)

[69] https://keras.io/api/models/model_training_apis/ (Προσπελάστηκε 7/7/2022)

[70] https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/ (Προσπελάστηκε 7/7/2022)

[71] https://www.mathworks.com/help/matlab/ref/plot.html (Προσπελάστηκε 7/7/2022)

[72] http://matlab.izmiran.ru/help/toolbox/nnet/postreg.html (Προσπελάστηκε 7/7/2022)

[73] https://matplotlib.org/ (Προσπελάστηκε 7/7/2022)

[74] Tukey, J. W. (1977). Exploratory data analysis. Addison-Wesley. ISBN: 978-0201076165

[75] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

[76] Michael L. Waskom, seaborn: statistical data visualization, the Journal of Open Source Software, 2021 , σελίδα 1

[80] https://pandas.pydata.org/docs/getting_started/index.html#getting-started (Προσπελάστηκε 7/7/2022)

[81] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html (Προσπελάστηκε 7/7/2022)

[82] https://github.com/mwaskom/seaborn/issues/655#issuecomment-125073496 (Προσπελάστηκε 7/7/2022)

[83] https://towardsdatascience.com/visualizing-artificial-neural-networks-anns-with-just-one-line-of-code-b4233607209e (Προσπελάστηκε 7/7/2022)

[85] https://analyticsindiamag.com/hands-on-guide-to-graphviz-python-tool-to-define-and-visualize-graphs/ (Προσπελάστηκε 7/7/2022)

[86] https://en.wikipedia.org/wiki/Unsupervised_learning (Προσπελάστηκε 7/7/2022)

[87] https://en.wikipedia.org/wiki/Reinforcement_learning (Προσπελάστηκε 7/7/2022)

[88] https://en.wikipedia.org/wiki/Mean_squared_error (Προσπελάστηκε 7/7/2022)

[89] G.E. Stavroulakis (2001) Inverse and Crack Identification Problems in Engineering Mechanics. Springer.

[90] G.E. Stavroulakis, G.A. Drosopoulos, A.D. Muradova (2022) Data-driven, data-based and artificial intelligence methods in structural mechanics. CESARE'22 Conference Proceedings. https://www.just.edu.jo/cesare22/documents/papers/145.pdf